Politeknik Caltex Riau

LAPORAN PROYEK AKHIR

RANCANG BANGUN *TRAINER KIT EDGE INDUSTRIAL INTERNET OF THINGS* BERBASIS REVOLUTION PI DENGAN KOMUNIKASI WEBSOCKET DAN DASHBOARD NI G WEB

Syahrul Sya'ban

NIM. 2220307031

Pembimbing

Heri Subagiyo, S.T.,M.T.

PROGRAM STUDI

TEKNOLOGI REKAYASA SISTEM ELEKTRONIKA

POLITEKNIK CALTEX RIAU

2026

LEMBAR PENGESAHAN

RANCANG BANGUN *TRAINER KIT EDGE INDUSTRIAL INTERNET OF THINGS* BERBASIS REVOLUTION PI DENGAN KOMUNIKASI WEBSOCKET DAN DASHBOARD NI G WEB

SYAHRUL SYA'BAN

NIM. 2220307031

Laporan Proyek Akhir ini disusun untuk memenuhi salah satu persyaratan memperoleh gelar **Sarjana Terapan Teknik (S.Tr.T)** di Politeknik Caltex Riau

Pekanbaru, 19 Agustus 2026

Disetujui oleh:

Pembimbing 1	:	Heri Subagiyo S.T., M.T. NIP. 007508	(Pembimbing)
Penguji 1	:	Yusmar Palapa Wijaya S.Si., M.T. NIP. 027608	(Penguji 1)
Penguji 2	:	Retno Tri Wahyuni S.T., M.T. NIP. 068314	(Penguji 2)

Mengetahui,

Ketua Program Studi Teknologi Rekayasa Sistem Elektronika

Yusmar Palapa Wijaya S.Si., M.T.
NIP. 027608

PERNYATAAN PENGGUNAAN KECERDASAN BUATAN

Dengan ini saya menyatakan bahwa proyek akhir yang berjudul:

Rancang Bangun *Trainer Kit Edge Industrial Internet Of Things* Berbasis Revolution Pi dengan Komunikasi Websocket dan Dashboard NI G Web

Menggunakan teknologi Kecerdasan Buatan (*Artificial Intelligence*) dalam penyusunan dokumen ini sebatas pada hal-hal berikut:

1) Membantu penyusunan dan penyempurnaan struktur penulisan serta penyajian laporan agar sistematis dan sesuai dengan kaidah penulisan ilmiah.
2) Membantu memberikan masukan dalam penyusunan, pengembangan, dan penyempurnaan dokumen Proyek Akhir.
3) Memberikan saran penyajian informasi dan perbaikan penulisan tanpa menggantikan proses analisis maupun pengambilan keputusan dalam proyek akhir ini.

Penggunaan tersebut **tidak menyangkut substansi utama proyek akhir**, seperti analisis, penyusunan ide pokok, perumusan konsep ilmiah, atau penentuan hasil proyek akhir. Seluruh substansi utama dalam dokumen ini merupakan hasil pemikiran dan karya saya sendiri. Apabila di kemudian hari ditemukan pelanggaran terhadap pernyataan ini, saya bersedia menerima konsekuensi sesuai ketentuan yang berlaku.

Pekanbaru, 29 Juli 2026

Yang menyatakan,

Syahrul Sya'ban

NIM. 2220307031

PERNYATAAN ORISINALITAS

Dengan ini saya menyatakan bahwa proyek akhir yang berjudul:

Rancang Bangun *Trainer Kit Edge Industrial Internet Of Things* Berbasis Revolution Pi dengan Komunikasi Websocket dan Dashboard NI G Web

Adalah benar hasil karya saya, dan tidak mengandung karya ilmiah atau tulisan yang pernah diajukan di suatu Perguruan Tinggi

Setiap kata yang dituliskan tidak mengandung plagiat, pernah ditulis maupun diterbitkan orang lain kecuali secara tertulis diacu dalam dokumen ini dan disebutkan pada daftar pustaka. Saya siap menanggung seluruh akibat apabila terbukti melakukan plagiat.

Pekanbaru, 29 Juli 2026

Yang menyatakan,

Syahrul Sya'ban

NIM. 2220307031

KESEPAKATAN PUBLIKASI

Demi pengembangan ilmu pengetahuan, dengan ini saya menyatakan:

1. Memberikan persetujuan kepada Politeknik Caltex Riau untuk menyimpan, mengolah dalam bentuk pangkalan data, merawat, mengalih media/formatkan dan mempublikasikan **Proyek Akhir** ini selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.
2. Tidak melakukan alih media/format dan publikasi dalam bentuk makalah ilmiah dari bagian atau keseluruhan **Proyek Akhir** ini ke suatu publikasi ilmiah, pada seminar ataupun jurnal, skala nasional maupun internasional, kecuali ada persetujuan dari saya dan Dosen Pembimbing Utama, dan mencantumkan nama saya, Dosen Pembimbing Utama dan nama-nama lain (jika ada) yang berkontribusi pada makalah.

Pekanbaru, 29 Juli 2026

Yang menyatakan,

Syahrul Sya'ban

NIM. 2220307031

KATA PENGANTAR

Segala puji syukur kehadirat Tuhan Yang Maha Esa yang telah melimpahkan rahmat dan karunia-Nya sehingga penulis **LAPORAN PROYEK AKHIR** dapat menyelesaikan proyek akhir berjudul **"Rancang Bangun *Trainer Kit Edge Industrial Internet Of Things* Berbasis Revolution Pi dengan Komunikasi WebSocket dan Dashboard NI G Web"**. yang disusun sebagai salah satu syarat untuk menyelesaikan jenjang pendidikan **Sarjana Terapan** pada Program Studi **Teknologi Rekayasa Sistem Elektronika** Politeknik Caltex Riau.

Penulis menyadari bahwa laporan proyek akhir ini masih jauh dari sempurna. Oleh karena itu, segala bentuk kritik dan saran yang bersifat membangun sangat penulis harapkan demi perbaikan di masa mendatang. Penulis berharap laporan ini dapat memberikan manfaat dan wawasan bagi para pembaca, serta menjadi bahan referensi untuk pengembangan proyek akhir selanjutnya.

Pekanbaru, 29 Juli 2026

Syahrul Sya'ban

UCAPAN TERIMA KASIH

1. Allah SWT atas segala rahmat, karunia, dan ridha-Nya sehingga penulis dapat menyelesaikan Proyek Akhir ini dengan baik.
2. Kedua orang tua penulis yang senantiasa memberikan doa, kasih sayang, dukungan moral maupun material, serta motivasi yang tiada henti selama proses penyusunan Proyek Akhir.
3. Kedua saudara penulis yang selalu memberikan doa, semangat, dan dukungan kepada penulis.
4. Bapak Dr. Dadang Syarif Sihabudin Sahid, S.Si., M.Sc. selaku Direktur Politeknik Caltex Riau yang telah mendedikasikan waktunya untuk meningkatkan perkembangan institusi Pendidikan Politeknik Caltex Riau
5. Bapak Yusmar Palapa Wijaya, S.Si., M.T. selaku Kepala Program Studi Teknologi Rekayasa Sistem Elektronika.
6. Bapak Heri Subagiyo, S.T., M.T. selaku dosen pembimbing yang telah meluangkan waktu, memberikan arahan, bimbingan, masukan, serta motivasi dengan penuh kesabaran selama proses penyusunan Proyek Akhir ini.
7. Seluruh Dosen Pengajar Program Studi **Teknologi Rekayasa Sistem Elektronika** Politeknik Caltex Riau yang telah memberikan ilmu, pengalaman, dan wawasan selama masa perkuliahan.
8. Teman-teman HIMIKA Angkatan 2022 Politeknik Caltex Riau atas kebersamaan, dukungan, dan semangat selama menjalani perkuliahan dan penyusunan Proyek Akhir.
9. Diri sendiri yang telah berusaha, bertahan, dan terus berjuang menghadapi berbagai tantangan selama proses penyusunan Proyek Akhir ini hingga dapat diselesaikan dengan baik.
10. Persib Bandung yang secara tidak langsung telah menjadi sumber hiburan dan motivasi bagi penulis selama penyusunan Proyek Akhir ini.
11. Bang Amirul Huda & Lab 193 yang membantu penulis dalam pengerjaan PA.
12. Seluruh pihak yang telah membantu, baik secara langsung maupun tidak langsung, dalam proses penyusunan Proyek Akhir ini yang tidak dapat disebutkan satu per satu.

ABSTRAK

Perkembangan transformasi digital industri mendorong penerapan *Industrial Internet of Things* (IIoT) untuk mendukung akuisisi data, pemantauan, dan pengendalian proses secara *real-time*. Namun, media pembelajaran yang tersedia umumnya masih berfokus pada simulasi mikrokontroler sehingga belum mampu merepresentasikan implementasi arsitektur Edge IIoT yang digunakan di lingkungan industri. Penelitian ini bertujuan merancang, membangun, dan mengevaluasi kinerja *Trainer Kit Edge Industrial Internet of Things* berbasis Revolution Pi yang terintegrasi dengan komunikasi WebSocket dan Dashboard NI G Web sebagai media pembelajaran otomasi industri. Sistem menggunakan arsitektur *Controller–Gateway* dengan Revolution Pi sebagai perangkat *edge* yang mendukung mode *Local* dan *Remote* melalui komunikasi WebSocket dua arah. Pengujian dilakukan pada dua skenario industri, yaitu *Metal Part Conveyor Monitoring System*, dan *Automatic Mixing Tank Process Control System*. Evaluasi dilakukan berdasarkan parameter *control success rate* dan *telemetry*, *latency* kontrol, *throughput uplink*, stabilitas interval pengiriman data, dan *round trip time* (RTT). Hasil pengujian menunjukkan *control success rate* sebesar 94,51%, *telemetry success rate* sebesar 98,01%, *latency* kontrol kurang dari 2 ms, *throughput uplink* sebesar 3,54–5,66 kbps, interval pengiriman data sebesar 1012–1015 ms, serta RTT sebesar 119,83–156,14 ms. Hasil tersebut menunjukkan bahwa sistem memiliki performa komunikasi yang cepat, stabil, dan andal sehingga layak digunakan sebagai media pembelajaran dan pengembangan aplikasi Edge IIoT.

Kata kunci *: Edge Computing, G-Web Development Software, Industrial Internet of Things (IIoT), Revolution Pi, WebSocket*

ABSTRACT

The rapid advancement of industrial digital transformation has accelerated the adoption of the Industrial Internet of Things (IIoT) to support real-time data acquisition, monitoring, and process control. However, existing educational platforms are generally limited to microcontroller-based simulations and do not adequately represent Edge IIoT architectures used in industrial environments. This research aims to design, develop, and evaluate a Revolution Pi-based Edge Industrial Internet of Things Trainer Kit integrated with WebSocket communication and an NI G Web Dashboard as an industrial automation learning platform. The system employs a Controller–Gateway architecture, with Revolution Pi functioning as the edge device and supporting both Local and Remote operating modes through bidirectional WebSocket communication. The system was evaluated using two industrial scenarios, namely the Metal Part Conveyor Monitoring System and the Automatic Mixing Tank Process Control System. Performance evaluation was conducted based on control success rate, telemetry success rate, control latency, uplink throughput, data transmission interval stability, and round trip time (RTT). The results show a control success rate of 94.51%, a telemetry success rate of 98.01%, control latency below 2 ms, uplink throughput ranging from 3.54–5.66 kbps, data transmission intervals of 1012–1015 ms, and RTT values ranging from 119.83–156.14 ms. These results demonstrate that the proposed system provides fast, stable, and reliable communication performance, making it suitable as a learning and development platform for Edge IIoT applications.

***Keywords** : Edge Computing, G-Web Development Software, Industrial Internet of Things (IIoT), Revolution Pi, WebSocket*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Perkembangan transformasi digital pada sektor industri saat ini diarahkan menuju implementasi konsep Industri 4.0 yang berfokus pada integrasi sistem fisik dan digital melalui beberapa pilar utama, di antaranya *Industrial Internet of Things* (IIoT), *Collaborative Robot* (Cobots), dan *Sustainability*. Di antara ketiga pilar tersebut, IIoT menjadi teknologi yang berperan penting dalam mendukung proses akuisisi data, monitoring, dan kontrol secara *real-time* melalui integrasi sensor, aktuator, serta sistem komunikasi berbasis jaringan. Hal tersebut dikarenakan kemampuan IIoT dalam menyediakan informasi operasional secara cepat dan akurat mampu meningkatkan efisiensi, produktivitas, serta keandalan sistem pada berbagai sektor industri, seperti energi, transportasi, manufaktur, dan kesehatan (Tidrea et al., 2023).

Seiring dengan perkembangan IIoT, arsitektur sistem otomasi industri turut mengalami perubahan menuju konsep *Edge Computing*, yaitu mekanisme pemrosesan data yang dilakukan pada perangkat *edge* sebelum diteruskan ke sistem tingkat yang lebih tinggi. Pendekatan ini mampu mengurangi latensi komunikasi, mempercepat waktu respons, serta mengurangi beban lalu lintas jaringan sehingga lebih sesuai untuk implementasi sistem monitoring dan kontrol secara *real-time*. Selain itu, implementasi *Edge Computing* memerlukan mekanisme komunikasi yang mampu mempertahankan koneksi secara kontinu agar proses pertukaran data dapat berlangsung cepat dan stabil. Salah satu protokol yang banyak digunakan untuk memenuhi kebutuhan tersebut adalah WebSocket karena mendukung komunikasi dua arah (*bidirectional communication*) secara *full duplex* antara perangkat *edge* dan antarmuka pengguna.

Namun, perkembangan teknologi tersebut belum sepenuhnya diimbangi dengan ketersediaan media pembelajaran yang mampu merepresentasikan implementasi sistem IIoT pada lingkungan industri. Sebagian besar media pembelajaran masih berfokus pada simulasi dasar menggunakan mikrokontroler atau *single-board computer* sehingga belum mampu menggambarkan arsitektur

sistem industri yang melibatkan perangkat *edge*, komunikasi jaringan, monitoring, dan kontrol secara *real-time*. Kondisi tersebut menyebabkan peserta didik lebih banyak memahami konsep IIoT secara teoritis dibandingkan implementasi praktis menggunakan perangkat industri yang sesungguhnya (Sisinni et al., 2018). Hal ini juga berkontribusi terhadap munculnya kesenjangan kompetensi (*skill gap*) dalam menghadapi kebutuhan industri modern.

Oleh karena itu, diperlukan suatu media pembelajaran yang mampu mensimulasikan implementasi sistem *Edge IIoT* secara lebih komprehensif. Media tersebut tidak hanya berfungsi sebagai alat peraga, tetapi juga harus mampu mengintegrasikan perangkat keras industri, komunikasi jaringan, monitoring, dan kontrol dalam satu platform pembelajaran yang menyerupai implementasi sistem industri sebenarnya. Salah satu perangkat yang memenuhi kebutuhan tersebut adalah Revolution Pi, yaitu *Industrial Edge Computer* yang bersifat modular, menggunakan sistem operasi Linux industri, serta kompatibel dengan berbagai standar komunikasi dan perangkat otomasi industri, termasuk DIN EN 61131-2. Karakteristik tersebut memungkinkan Revolution Pi digunakan sebagai media pembelajaran yang lebih representatif dibandingkan perangkat pembelajaran berbasis mikrokontroler konvensional.

Berdasarkan kebutuhan tersebut, pada proyek akhir ini dikembangkan *Trainer Kit Edge Industrial Internet of Things (IIoT)* berbasis Revolution Pi dengan komunikasi WebSocket dan Dashboard NI G Web. Trainer kit ini mengintegrasikan Revolution Pi sebagai perangkat *edge*, berbagai sensor dan aktuator, komunikasi WebSocket, serta Dashboard NI G Web sebagai antarmuka monitoring dan kontrol berbasis web dalam satu sistem yang terintegrasi. Selain mendukung komunikasi dua arah (*bidirectional communication*) dengan latensi yang rendah, trainer kit ini juga dirancang untuk merepresentasikan berbagai skenario otomasi industri. Dengan demikian, trainer kit yang dikembangkan tidak hanya berfungsi sebagai media pembelajaran, tetapi juga sebagai sarana simulasi dan pengujian implementasi sistem Edge IIoT yang lebih aplikatif, terintegrasi, dan mendekati kondisi industri modern.

1.2 Perumusan Masalah

Berdasarkan latar belakang yang telah diuraikan serta kebutuhan implementasi *Industrial Internet of Things* (IIoT) di sektor industri. Maka dapat dirumuskan masalah yang menjadi fokus pada proyek akhir ini adalah sebagai berikut:

1. Bagaimana merancang *Trainer Kit Edge IIoT* Menggunakan Revolution Pi yang memenuhi karakteristik sistem industri.
2. Bagaimana mengimplementasikan mekanisme akuisisi data sensor dan kontrol aktuator pada Revolution Pi sebagai perangkat *edge*.
3. Bagaimana membangun komunikasi data dua arah antara Revolution Pi dan NI G Web menggunakan protokol WebSocket.
4. Bagaimana mengintegrasikan sistem monitoring dan kontrol berbasis web agar dapat menampilkan data serta mengirim perintah secara sinkron.
5. Bagaimana mengevaluasi performa komunikasi sistem berdasarkan parameter *latency, control success rate, telemetry success rate, throughput,* stabilitas interval pengiriman data dan *round trip time*.

1.3 Batasan Masalah

Agar ruang lingkup Proyek Akhir tidak terlalu luas, batasan masalah yang diterapkan adalah sebagai berikut:

1. Perancangan Trainer Kit *Edge IIoT* dibatasi pada penggunaan Revolution Pi sebagai perangkat *edge* utama.
2. Implementasi sistem dibatasi pada sensor PT100, sensor suhu dan kelembapan XY-MD02, sensor proximity, signal generator 4–20 mA, serta aktuator berupa LED, buzzer, motor DC, dan relay SSR
3. Komunikasi data dua arah antara RevPi dan NI G Web difokuskan pada penggunaan protokol WebSocket.
4. Antarmuka monitoring dan kontrol dikembangkan menggunakan NI G Web.
5. Implementasi skenario aplikasi dibatasi pada dua studi kasus industri, yaitu *Metal Part Conveyor Monitoring System dan Automatic Mixing Tank Process Control System.*
6. Evaluasi sistem dibatasi pada pengukuran parameter performa komunikasi, meliputi:

- *Latency* pengiriman dan penerimaan data,
- Tingkat keberhasilan eksekusi kontrol dan transmisi telemetry
- *Throughput* komunikasi,
- Stabilitas interval pengiriman data telemetry.
- *Round Trip Time* (RTT)

7. Sistem yang dikembangkan beroperasi pada jaringan lokal (*local area network*).

1.4 Tujuan dan Manfaat

1.4.1 Tujuan

Tujuan proyek akhir ini adalah merancang dan membangun *Trainer Kit Edge Industrial Internet of Things (IIoT)* berbasis Revolution Pi yang mengintegrasikan sensor, aktuator, komunikasi WebSocket, dan Dashboard NI G Web dalam satu platform terintegrasi untuk mendukung implementasi monitoring dan kontrol. *Trainer kit* yang dikembangkan digunakan untuk merepresentasikan arsitektur *Edge IIoT* pada lingkungan industri serta mendukung proses pembelajaran dan demonstrasi sistem otomasi berbasis jaringan. Selain itu, sistem yang diimplementasikan pada *trainer kit* dievaluasi berdasarkan parameter performa komunikasi yang meliputi *latency* komunikasi, *control success rate*, *telemetry success rate, throughput* komunikasi, stabilitas interval pengiriman data pada jaringan lokal dan *round trip time communication.*

1.4.2 Manfaat

Hasil dari Proyek Akhir ini diharapkan dapat memberikan manfaat sebagai berikut:

1. Menjadi referensi dalam pengembangan *trainer kit* berbasis *Edge Industrial Internet of Things* untuk mendukung kegiatan pembelajaran, penelitian, dan praktikum pada bidang otomasi industri.
2. Menyediakan platform implementasi sistem monitoring dan kontrol berbasis Revolution Pi, komunikasi WebSocket, dan Dashboard NI G Web yang dapat digunakan sebagai media simulasi, demonstrasi, serta pengujian implementasi sistem Edge IIoT.
3. Membantu peserta didik memahami implementasi teknologi Edge *Industrial Internet of Things (IIoT)* melalui integrasi sensor, aktuator, perangkat *edge*, komunikasi, serta sistem monitoring dan kontrol berbasis web.

BAB 2
TINJAUAN PUSTAKA

2.1 Landasan Teori

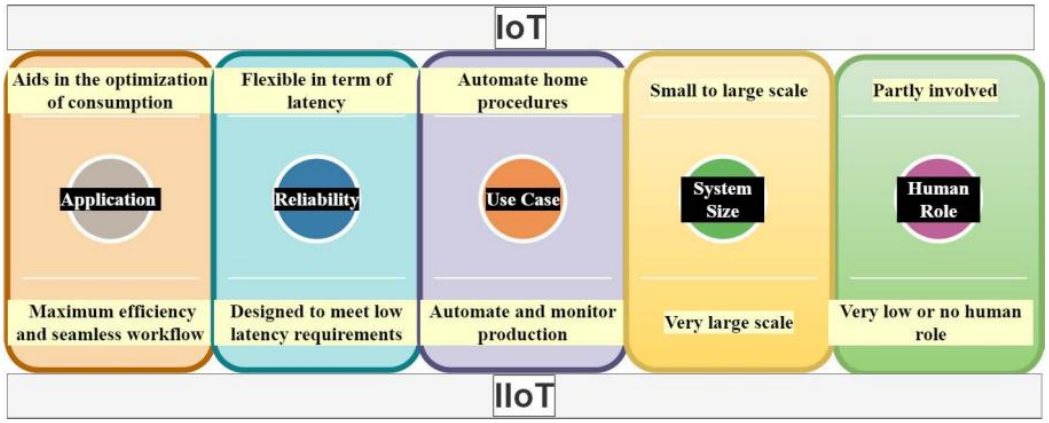
Pada subbab ini dijelaskan teori-teori yang mendasari perancangan dan implementasi sistem pada proyek akhir ini. Pembahasan meliputi konsep *Industrial Internet of Things (IIoT), edge computing, Revolution Pi*, komunikasi berbasis WebSocket, serta teknologi pendukung lainnya yang berkaitan dengan sistem monitoring dan kontrol.

2.1.1 Industrial Internet of Things (IIoT)

Industrial Internet of Things (IIoT) merupakan pengembangan lanjutan dari konsep *Internet of Things* (IoT) yang secara spesifik berfokus pada sektor manufaktur dan sistem otomasi industri (Younan et al., 2020). Perangkat IIoT mengintegrasikan fitur-fitur IoT seperti penginderaan (*sensing*), penggerakan (*actuating*), interkoneksi perangkat, serta pemrosesan data pada berbagai lapisan sistem produksi guna meningkatkan efektivitas dan dampak teknologi terhadap proses manufaktur (Mahmood, 2019). Melalui mekanisme akuisisi data secara kontinu, pemrosesan informasi berbasis komputasi, serta pengambilan keputusan dengan keterlibatan manusia yang minimal, IIoT mampu meningkatkan aksesibilitas sistem, performa operasional, skalabilitas infrastruktur, efisiensi waktu produksi, serta optimalisasi biaya (Zhang et al., 2020).

Berbeda dengan lingkungan produksi konvensional yang bergantung pada interaksi lokal antar komponen sistem, IIoT memungkinkan komunikasi lintas lokasi melalui jaringan terintegrasi, sehingga batasan geografis dalam proses monitoring dan kontrol dapat diminimalisir. Namun, implementasi IIoT tidak terlepas dari tantangan teknis yang berasal dari karakteristik dasar IoT itu sendiri. Hal itu dikarenakan perangkat IoT umumnya memiliki keterbatasan kapasitas memori, kebutuhan konsumsi energi yang rendah, keterbatasan pada komunikasi nirkabel, serta kemampuan komputasi yang terbatas. Faktor-faktor tersebut secara langsung memengaruhi efektivitas implementasi dan keberlanjutan infrastruktur IIoT dalam lingkungan industri manufaktur.

Dengan mempertimbangkan karakteristik, tujuan, serta kompleksitas implementasinya, dapat dipahami bahwa IIoT memiliki perbedaan mendasar dibandingkan dengan IoT konvensional. Perbedaan tersebut tidak hanya terletak pada konteks penggunaannya, tetapi juga pada kebutuhan keandalan sistem, skala integrasi perangkat, serta tuntutan operasional dalam lingkungan industri. Perbedaan tersebut dapat diilustrasikan melalui perbandingan IoT dan IIoT sebagaimana ditunjukkan pada Gambar 2.1 dan Tabel 2.1.



Gambar 2.1 Perbandingan antara IoT dengan IIoT[1]

Gambar 2.1 menunjukkan perbandingan antara konsep *Internet of Things (IoT)* dan *Industrial Internet of Things (IIoT*), yang memperlihatkan perbedaan mendasar dari aspek tujuan penggunaan, kebutuhan sistem, skala implementasi, serta tingkat keterlibatan manusia dalam operasionalnya.

1. **Dari Segi Tujuan Penggunaan *(Application vs Use Case)***

   IoT umumnya hanya berfokus pada optimalisasi konsumsi dan peningkatan kenyamanan dalam skala umum, seperti pada sistem *smart home* dan perangkat konsumen. Sebaliknya, IIoT berorientasi pada otomatisasi proses industri, monitoring kondisi mesin, serta pengendalian proses produksi yang bersifat kritis. Hal ini dikarenakan IIoT dirancang untuk mendukung keberlangsungan operasional industri yang menuntut stabilitas dan presisi tinggi.

[1] Alabadi, M., Habbal, A., & Wei, X. (2022). Industrial Internet of Things: Requirements, Architecture, Challenges, and Future Research Directions. In *IEEE Access* (Vol. 10, pp. 66374–66400). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ACCESS.2022.3185049

2. **Dari Segi Latensi dan Keandalan *(Reliability)***

   IoT relatif lebih toleran terhadap variasi latensi dan tidak selalu membutuhkan respons waktu yang ketat. Namun, pada IIoT, kebutuhan latensi rendah dan keandalan tinggi menjadi aspek fundamental. Hal ini dikarenakan sistem industri sering kali mengendalikan mesin produksi dan proses yang sensitif terhadap keterlambatan respons, sehingga gangguan komunikasi dapat berdampak pada keselamatan maupun kerugian operasional.

3. **Dari Segi Sistem dan Skala *(System Size and Integration)***

   Implementasi IoT umumnya berada pada skala kecil hingga menengah dengan jumlah perangkat yang terbatas. Sebaliknya, IIoT dapat diimplementasikan dalam skala sangat besar dengan integrasi ribuan sensor, aktuator, pengendali industri, serta sistem produksi dalam satu ekosistem terintegrasi. Sehingga aspek skalabilitas dan interoperabilitas menjadi karakteristik penting dalam sistem IIoT.

4. **Dari Segi Peran Manusia *(Human Role)***

   Pada sistem IoT, manusia masih memiliki peran signifikan dalam pengawasan dan pengoperasian. Sementara itu, pada IIoT, tingkat keterlibatan manusia lebih rendah karena sistem dirancang untuk berjalan secara otomatis melalui logika kontrol terintegrasi dan pengambilan keputusan berbasis data.

Berdasarkan uraian tersebut, perbedaan antara IoT dan IIoT tidak hanya dapat dipahami secara deskriptif, tetapi juga dirangkum secara sistematis agar setiap aspek pembeda dapat dianalisis secara komprehensif yang disajikan pada Tabel 2.1.

Tabel 2.1 Perbandingan antara IoT dengan IIoT

Perspektif	**IoT**	**IIoT**
Application (Usage)	*Optimize Consumption*	*Maximum Eficiency*
Reliability	*Flexible*	*Low Latency*
Use Cases	*Personal Use*	*Productions and Business*
System Size	*Small – to - Medium*	*Medium – to – Large*
Human Role (Automation)	*Limited*	*No Human Intervention*

Selain perbedaan pada tujuan dan skala implementasi, sistem IIoT juga memiliki karakteristik khusus pada infrastruktur jaringan komunikasinya. Berbeda dengan jaringan internet konvensional yang bekerja berdasarkan prinsip *best-effort*—di mana pengiriman data tidak selalu memiliki jaminan waktu tiba yang

pasti. Jaringan dalam lingkungan industri dirancang untuk memberikan latensi rendah, keandalan tinggi, serta ketersediaan sistem yang stabil. Hal itu dikarenakan proses produksi industri umumnya melibatkan sistem kontrol tertutup (*closed-loop control*) yang sangat sensitif terhadap keterlambatan maupun gangguan komunikasi (Sisinni et al., 2018). Keterlambatan dalam hitungan milidetik pun dapat memengaruhi kualitas produksi atau bahkan menimbulkan risiko keselamatan kerja.

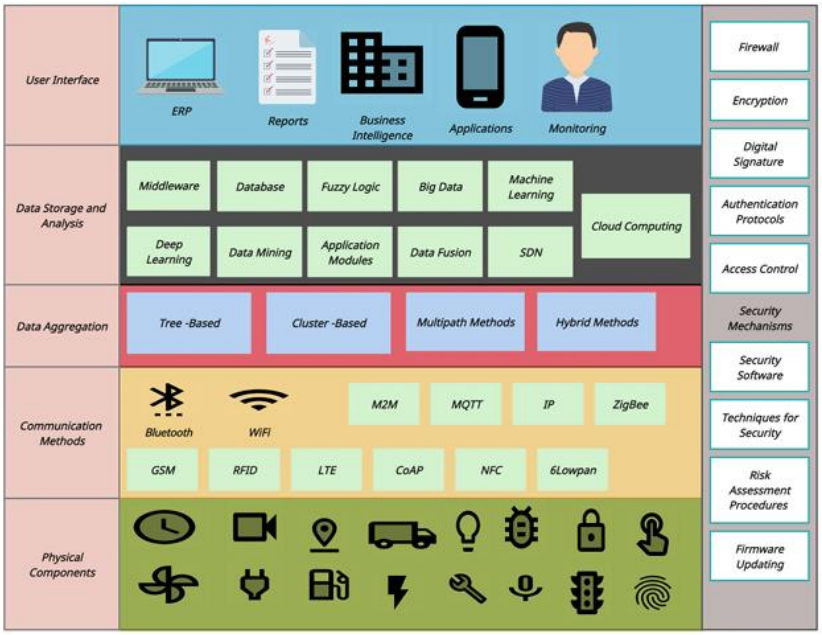
Dalam praktik implementasinya, jaringan IIoT tidak selalu dipisahkan secara fisik dari jaringan komputer perusahaan (enterprise). Namun demikian, sistem industri umumnya menerapkan pemisahan secara logis melalui segmentasi antara domain *Operational Technology* (OT), yang menangani mesin dan proses produksi, dengan domain *Information Technology* (IT), yang menangani pengolahan data dan sistem manajemen. Pendekatan ini biasanya diwujudkan dalam arsitektur berlapis (*three-tier architecture*) yang terdiri dari lapisan *edge*, *platform*, dan *enterprise*, sehingga komunikasi pada level produksi tetap terisolasi dan stabil, sementara data tetap dapat diintegrasikan ke sistem manajemen tingkat atas (Sisinni et al., 2018).

Prinsip ini juga sejalan dengan standar keamanan industri ISA/IEC 62443 yang merekomendasikan pembagian jaringan ke dalam *security zones* untuk mencegah gangguan antara jaringan produksi dan jaringan bisnis.

Dengan demikian, jaringan komunikasi dalam sistem IIoT industri tidak sekadar menggunakan internet umum secara langsung, melainkan dirancang dengan mekanisme segmentasi dan pengendalian lalu lintas data yang ketat. Pendekatan ini bertujuan untuk menjaga keandalan komunikasi, keamanan sistem, serta memastikan proses produksi dapat berjalan secara konsisten dan berkelanjutan.

2.1.2 Arsitektur Sistem IIoT

Arsitektur IIoT pada umumnya umumnya dirancang secara berlapis guna memastikan proses komunikasi, pengolahan data, serta pengambilan keputusan dapat berjalan secara efisien dan andal. Gambar dibawah ini menggambarkan arsitektur IIoT. Di mana setiap lapisan memiliki peran dan fungsi yang berbeda namun saling terhubung satu sama lain (Alabadi et al., 2022). Adapun komponen utama penyusun arsitektur IIoT adalah sebagai berikut.



Gambar 2.2 Arsitektur Sistem IIoT[2]

Gambar 2.2 merupakan arsitektur yang menekankan bahwa keberadaan komponen-komponen tertentu merupakan syarat utama agar tujuan integrasi *Industrial Internet of Things (IIoT)* dapat tercapai. Komponen-komponen tersebut meliputi:

1. **Komponen fisik (*Physical Components*)**

   Komponen ini mencakup seluruh objek fisik yang terdapat dalam sistem manufaktur. Elemen yang paling utama adalah mesin-mesin yang menjalankan proses produksi industri. Selain itu, terdapat pula sensor dan aktuator.

2. **Metode komunikasi (*Communication Methods*)**

   IIoT sangat bergantung pada protokol komunikasi nirkabel untuk mentransfer data antar elemen jaringan. Metode komunikasi dalam IIoT harus memenuhi berbagai spesifikasi, antara lain konsumsi daya yang rendah, kapasitas komunikasi yang tinggi, serta konektivitas yang stabil.

3. **Metode agregasi data (*Data Aggregation Methods*)**

   Agregasi data merupakan proses pengumpulan data mentah dari berbagai perangkat, seperti sensor atau objek fisik, yang kemudian diolah menjadi

[2] Alabadi, M., Habbal, A., & Wei, X. (2022). Industrial Internet of Things: Requirements, Architecture, Challenges, and Future Research Directions. In *IEEE Access* (Vol. 10, pp. 66374–66400). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ACCESS.2022.3185049

keluaran yang lebih ringkas dan terstruktur. Mekanisme ini bertujuan untuk mengurangi volume transmisi data, menyederhanakan kompleksitas informasi, mempermudah proses analisis, serta menghemat penggunaan bandwidth dan energi. Sebagai contoh, dibandingkan mengirimkan 100 data suhu setiap detik, sistem dapat mengirimkan nilai rata-rata dalam interval tertentu, nilai maksimum atau minimum, maupun status kondisi seperti "normal" atau "*overheat*" sebagai representasi data yang telah diagregasi.

4. **Penyimpanan data (*Data Storage*)**

   Pada arsitektur IIoT konvensional, *cloud* sering digunakan sebagai media utama penyimpanan dan analisis data karena memiliki kapasitas komputasi dan skalabilitas yang tinggi. Namun, ketergantungan penuh terhadap *cloud* dapat meningkatkan latensi komunikasi serta beban jaringan, terutama pada sistem industri yang memerlukan respons waktu yang cepat dan konsisten. Oleh karena itu, konsep *edge computing* berkembang sebagai pendekatan alternatif, di mana proses akuisisi, pemrosesan, dan penyimpanan data dilakukan lebih dekat dengan sumber data. Dengan memanfaatkan perangkat *edge*, sebagian proses analisis dan pengambilan keputusan dapat dilakukan secara lokal sebelum data dikirim ke sistem yang lebih tinggi. Pendekatan ini mampu mengurangi latensi, meningkatkan responsivitas sistem, serta meminimalisir ketergantungan terhadap infrastruktur *cloud* dalam operasional industri.

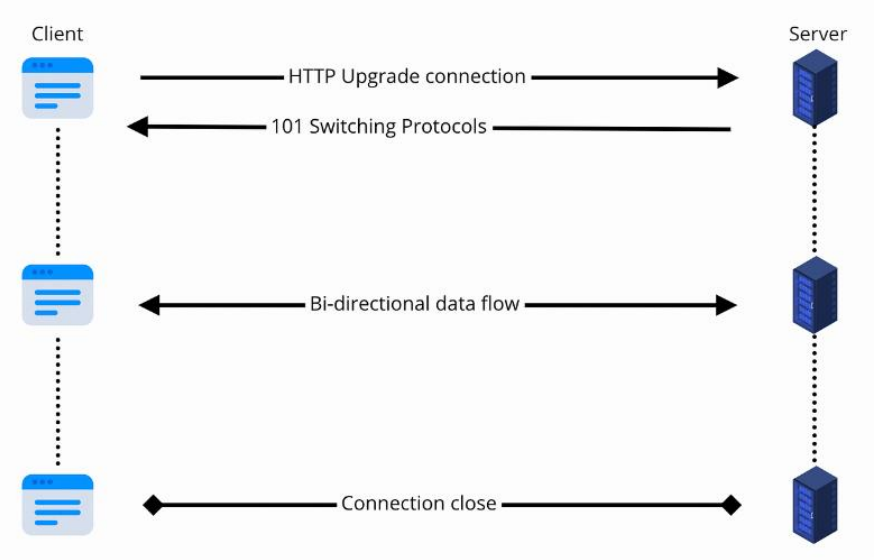
5. **Antarmuka pengguna (*User Interface*)**

   Antarmuka pengguna (*User Interface*) berada pada lapisan aplikasi dalam sistem IIoT dan berfungsi sebagai penghubung antara pengguna dan sistem. Melalui standarisasi data dan protokol komunikasi, antarmuka ini memungkinkan perangkat saling terintegrasi dengan baik. Dengan adanya antarmuka tersebut, proses monitoring dan kendali jarak jauh dapat dilakukan secara lebih mudah dan terstruktur dalam lingkungan industri.

2.1.3 Protokol Komunikasi WebSocket

WebSocket merupakan protokol komunikasi dua arah (*full-duplex*) yang memungkinkan pertukaran data secara real-time antara *server* dan *client* melalui koneksi TCP/IP. Berbeda dengan HTTP yang bersifat *request–response*, WebSocket memungkinkan koneksi yang bersifat persisten sehingga data dapat

dikirim dan diterima secara instan tanpa perlu melakukan permintaan ulang atau *refresh* halaman. Karakteristik ini menjadikan *WebSocket* sangat sesuai untuk aplikasi yang membutuhkan komunikasi *real-time* (Melnikov & Fette, 2020).



Gambar 2.3 Arsitektur Protokol Komunikasi WebSocket[3]

Gambar 2.3 merupakan arsitektur protokol komunikasi yang digunakan oleh websocket. Dimana, *WebSocket* menggunakan model komunikasi berbasis *event*, di mana *server* dan *client* dapat saling mengirim dan menerima pesan secara asinkron.

A. Cara Kerja *WebSocket*

Mekanisme kerja WebSocket diawali dengan proses *handshake* antara klien dan server. Pada tahap ini, klien mengirimkan permintaan koneksi menggunakan protokol HTTP dengan metode *HTTP Upgrade*. Jika server menyetujui permintaan tersebut, maka koneksi HTTP akan ditingkatkan (*upgrade*) menjadi koneksi WebSocket.

[3] Media, S. (2023, March 11). *Cross-site WebSocket hijacking | by Sibercat Media | Medium*. https://muh-hidayat7799.medium.com/ngesec-bebas-1-websocket-pentesting-fbda71f3167f

Gambar 2.4 Cara Kerja WebSocket[4]

Gambar 2.4 menjelaskan alur kerja websocket. Setelah proses *handshake* berhasil, terbentuk koneksi WebSocket yang bersifat persisten antara klien dan server. Berbeda dengan HTTP yang membuka koneksi baru untuk setiap permintaan, WebSocket mempertahankan satu koneksi terbuka selama sesi komunikasi berlangsung. Koneksi hanya akan ditutup apabila salah satu pihak mengakhiri sesi atau terjad *timeout*.

Pada kondisi koneksi aktif, komunikasi berlangsung secara dua arah (*full-duplex*), sehingga klien dan server dapat saling mengirim dan menerima pesan melalui satu saluran komunikasi yang sama tanpa harus menunggu permintaan terlebih dahulu. Hal ini memungkinkan pertukaran data secara kontinu dengan latensi rendah, sehingga sesuai untuk aplikasi monitoring dan kontrol berbasis jaringan.

B. Library Socket IO

Socket.IO merupakan salah satu *library* yang menyediakan mekanisme komunikasi real-time berbasis WebSocket dengan karakteristik komunikasi dua arah (*full-duplex*), dan berbasis *event*. Socket.IO dirancang untuk menyederhanakan implementasi komunikasi *real-time* antara klien dan server pada aplikasi web.

Penggunaan Socket.IO memberikan kemudahan dalam pengembangan aplikasi *real-time* karena menyediakan fitur tambahan, seperti manajemen koneksi, pengiriman pesan berbasis *event*, serta pengelompokan klien melalui *rooms* atau *namespaces*. Oleh karena itu, Socket.IO sangat sesuai digunakan

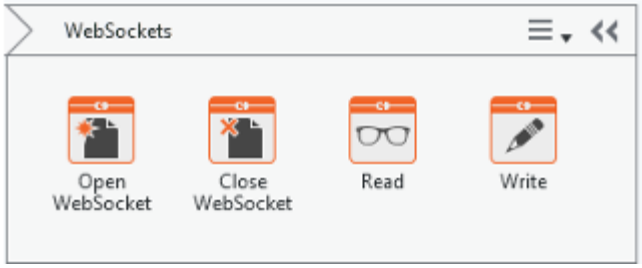
[4] Media, S. (2023, March 11). *Cross-site WebSocket hijacking | by Sibercat Media | Medium*. https://muh-hidayat7799.medium.com/ngesec-bebas-1-websocket-pentesting-fbda71f3167f

pada aplikasi web yang membutuhkan pembaruan data secara real-time, responsif, dan andal.

C. WebSocket Client APIs pada G Web

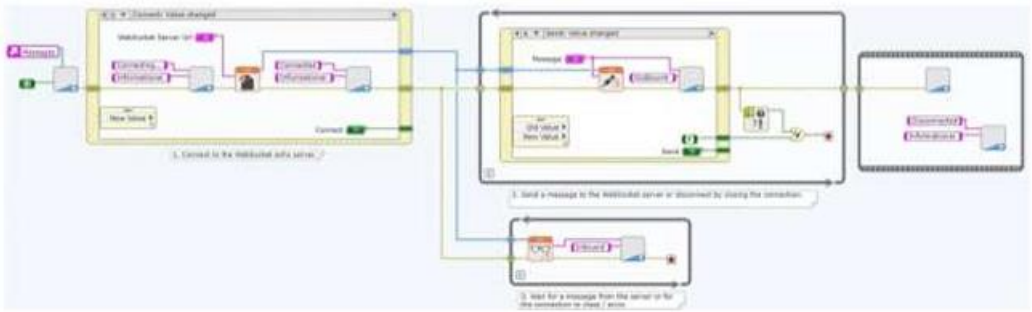
Metode ini dapat digunakan untuk berkomunikasi/bertukar data dengan WebSockets Server. WebSocket menawarkan koneksi TCP secara terus menerus antara klien dan server. WebSocket ideal untuk data streaming karena memiliki latensi yang rendah.

Fungsi-fungsi Websocket Client APIs dapat diambil pada pallete Data Communications>>Internet>>WebSockets. Berikut fungsi-fungsi pada WebSocket Client APIs:



Gambar 2.5 Fungsi Fungsi WebSocket Client APIs

Berikut contoh diagram dari penggunaan WebSocket Client APIs pada G Web:



Gambar 2.6 Diagram WebSocket pada G Web

2.1.4 NI G-Web Development Software

G Web Development Software adalah alat yang berguna untuk pengembang web yang ingin membuat situs web responsif tanpa keterampilan pemrograman web yang luas. Ini adalah alat visual yang memungkinkan pengguna untuk drag dan drop elemen ke halaman web, menghasilkan kode HTML dan CSS yang diperlukan.

NI G Web Development Software dirancang untuk mendukung pembuatan

aplikasi web yang bersifat interaktif dan *event-driven*, sehingga sangat sesuai digunakan pada aplikasi monitoring dan visualisasi data secara *real-time*. Selain itu, sifatnya yang berbasis web memungkinkan aplikasi diakses secara lintas platform tanpa memerlukan instalasi perangkat lunak tambahan pada sisi klien. Dengan kemampuan integrasi ke dalam ekosistem perangkat dan sistem National Instruments (National Instruments, 2024).

Gambar 2.7 Logo *G-Web Development Software*[5]

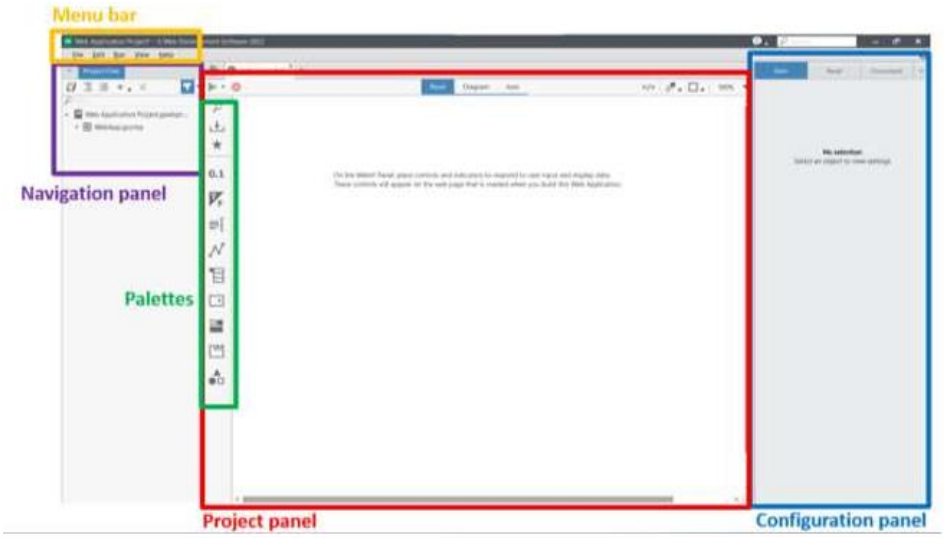
Salah satu keuntungan utama dari menggunakan G Web Development Software adalah versatilitasnya, karena kompatibel dengan berbagai perangkat dan browser. Pengembang dapat menyesuaikan situs web mereka dengan mudah, berkat kemampuan dinamis dari software. Selain itu, program ini terintegrasikan dengan baik dengan produk National Instruments lainnya seperti LabVIEW. Fitur ini memberikan proses yang lebih efisien dan rapi ketika menggunakan alat-alat ini bersama-sama.

Keuntungan utama lain dari menggunakan G Web Development Software adalah antarmuka intuitif yang ditawarkan. Software ini memudahkan pengguna yang tidak berpengalaman untuk membuat situs web yang fungsional dan indah. Pengguna dapat menggunakan software untuk menyesuaikan situs web dan menggunakan tutorial yang tersedia untuk mendapatkan hasil maksimal dari alat ini. Secara keseluruhan, G Web Development Software sangat cocok untuk setiap pengembang web yang ingin menyederhanakan proses pengembangan web mereka. Alat ini memungkinkan pengguna untuk membuat situs web responsif tanpa keterampilan pemrograman web yang luas yang dibutuhkan sambil menawarkan solusi yang hemat biaya bagi pengguna.

[5] National Instruments. (2024). *G Web Development Software*.

A. Bagian Bagian WebVI pada G Web Development Software

Berikut merupakan bagian bagian yang ada pada WebVI Gweb Development Software



Gambar 2.8 Antarmuka Pada G Web

Berikut penjelasan per bagian yang ada pada Gambar 2.8:

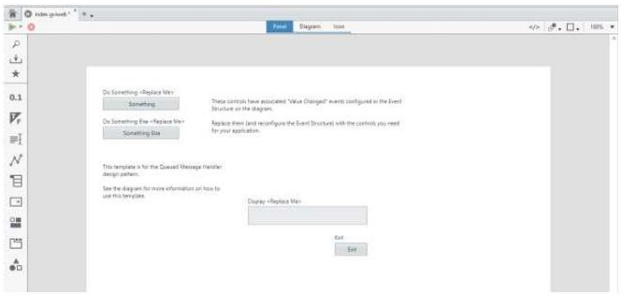
- **Menu bar** – Berisi menu-menu operasi untuk melakukan pengaturan dan perubahan pada project secara khusus atau G Web Development Software secara umum
- **Navigation panel** – Berisikan file-file dan komponen-komponen pendukung yang digunakan pada project
- **Palettes** – Berisi fungsi atau control yang dapat digunakan dalam aplikasi yang dibangun.
- **Project panel** – Tempat dimana tampilan dari project yang dibangun dapat dilihat (tab panel), serta logic yang digunakan di dalamnya (tab block diagram).
- **Configuration panel** – Berisi menu-menu yang dapat digunakan untuk mengatur project dan palette yang digunakan di dalam Project panel.

Adapun penjelasan lebih detailnya sebagai berikut:

1) Front Panel

Front Panel adalah bagian LabVIEW yang berlatar belakang abu-abu yang berfungsi meletakkan kontrol beserta indikator, dan menjadi antarmuka

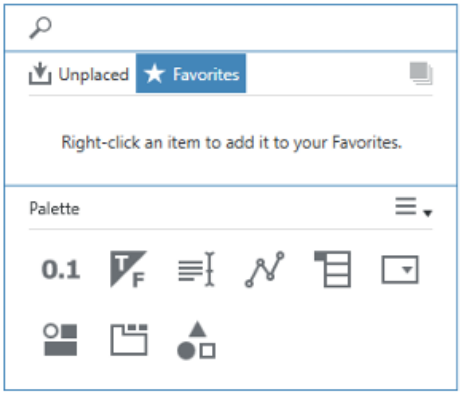
pengguna dari WebVI. Front Panel. Kontrol adalah elemen grafis pada user interface (UI) yang memungkinkan pengguna untuk berinteraksi dengan aplikasi web. Contohnya seperti tombol, input teks, dan dropdown menu. Pada gambar di bawah terlihat pada bagian yang diberi lingkaran biru. Sedangkan untuk Indikator adalah elemen UI yang menampilkan informasi kepada pengguna. Contohnya seperti progress bar, loading indicator, dan notification messages.



Gambar 2.9 Front Panel

2) Controls Palette

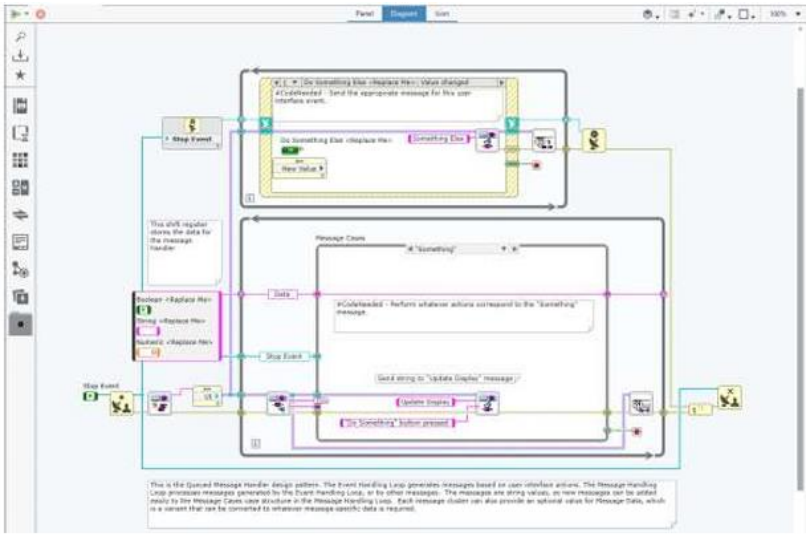
Controls Palette berisi kontrol dan indikator yang digunakan untuk membuat antarmuka di Front Panel. Controls Palette dapat diakses dari jendela Front Panel pada bagian palette di kiri atau dengan mengeklik kanan pada ruang kosong di jendela Front Panel. Tampilan dari Controls Palette dapat lihat pada gambar berikut:



Gambar 2.10 Controls Palette

3) Block Diagram

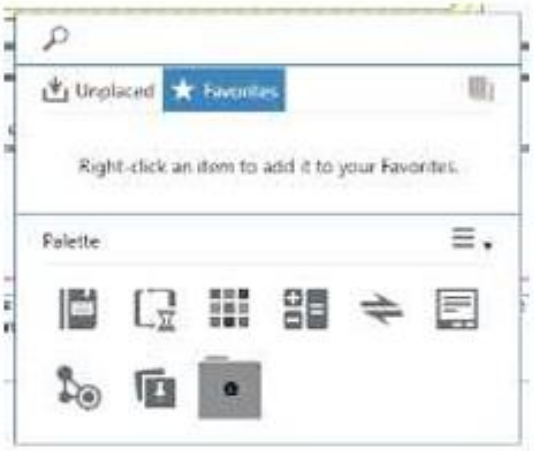
Block Diagram adalah bagian jendela G Web, berisi source code grafis yang dibuat, dan berfungsi sebagai instruksi untuk Front Panel. Tampilan dari Block Diagram dapat lihat pada gambar berikut:



Gambar 2.11 Block Diagram

4) Functions Palette

Functions Palette berisi gVI dan fungsi yang digunakan untuk membangun Block Diagram. Functions palette dapat diakses pada palette di bagian kiri panel atau dengan mengeklik kanan pada ruang kosong di jendela Block Diagram. Tampilan dari Functions Palette dapat lihat pada berikut:



Gambar 2.12 Functions Palette

2.1.5 *Revolution Pi*

Revolution Pi (RevPi) merupakan sebuah komputer industri modular dan terbuka (*open-source*) yang dikembangkan oleh Kunbus GmbH, Jerman. Produk ini berbasis *Raspberry Pi Compute Module* dan telah disesuaikan untuk memenuhi standar industri seperti EN61131-2, menjadikannya alternatif yang ekonomis dan fleksibel dibandingkan PLC konvensional. RevPi dirancang untuk dapat dipasang pada DIN-rail seperti perangkat industri lainnya dan mendukung sistem operasi

berbasis *Linux* (Raspbian dengan kernel RT patch) untuk menjamin stabilitas serta kinerja *real-time* (Revolution Pi, 2016).

Sebagai sistem terbuka, Revolution Pi memungkinkan pengguna untuk melakukan pengembangan mandiri dengan berbagai bahasa pemrograman seperti Python, C, Node-RED, maupun CODESYS. Fleksibilitas ini memungkinkan RevPi digunakan pada berbagai aplikasi, mulai dari otomasi industri, pengumpulan data (*data acquisition*), integrasi *cloud* dan *edge computing*, hingga pengembangan sistem berbasis *Internet of Things (IoT)* dan *Industry 4.0.* Pada Gambar 2.13 akan ditampilkan bentuk fisik Revolution Pi Connect SE beserta bagian-bagian utamanya.

Gambar 2.13 Revolution Pi Connect SE[6]

Pada Tabel 2.2 dan Tabel 2.3 dapat dilihat komponen pendukung serta spesifikasi dari revolution pi connect se.

Tabel 2.2 Komponen Pendukung Revolution Pi Connect SE

Position	***Component***	***Application***
1	*X2 connector*	*Relay output, Digital Input*
2	*6 x status LED*	*RevPi LEDs*
3	*2 x RJ45 Ethernet*	*Ethernet Interfaces RJ45, Establishing a Network Connection*
4	*Micro USB*	*Saving and Reinstalling the Image*
5	*2 x USB A*	*USB Interfaces*
6	*RS485 Socket*	*RS485 Serial Interface*

[6] Revolution Pi. (2016). *RevPi Connect S/SE | Industrial Raspberry Pi*. https://revolutionpi.com/en/docs/revpi-connect-se

Position	*Component*	*Application*
7	*X4 Connector*	*Connecting the Power Supply. Watchdog*
8	*2 x locking clip*	*Mounting the Device on a DIN Rail*
9	*Ventilation Slots*	*Mounting the Device on a DIN Rail*
10	*PiBridge*	*Connecting Expansion Modules*
11	*ConBridge*	*Connecting Expansion Modules*
12	*Micro HDMI*	*Setting up Desktop Mode*

Adapun pada Tabel 2.3 disajikan ringkasan spesifikasi dan standarisasi yang dimiliki oleh Revolution Pi, yang meliputi aspek spesifikasi fisik dan standar industri, spesifikasi daya, serta spesifikasi komputasi. Penyajian dalam bentuk tabel ini bertujuan untuk memberikan gambaran terstruktur mengenai karakteristik teknis perangkat yang mendukung implementasinya sebagai perangkat *edg*e dalam sistem. IIoT.

Tabel 2.3 Spesifikasi Revolution Pi Connect SE

Item Type	***Description***
Housing Type	DIN rail housing (for DIN rail version EN 50022)
Protection Class	*IP20 / NEMA Class 1*
Operating Temperature	-25 … +55 °C
Power Supply	12 … 24 V DC -15 % / +20 %, reverse-polarity protected
Maximum power consumption	*20 W (incl. 1 A total USB output current)*
Max. relative humidity (at 40 °C)	*up to 93 % (non-condensing)*
Processor	Broadcom BCM2711, quad-core Arm Cortex-A72
Clock rate	*1.5 GHz*
RAM	*1 GB LPDDR4*
Flash memory	*(8/16/32 GB)*

Spesifikasi tersebut menjadi dasar pertimbangan dalam pemilihan Revolution Pi sebagai platform utama pada trainer kit Edge IIoT yang akan dikembangkan ini. Selain memenuhi standar perangkat industri, Revolution Pi berperan sebagai perangkat *edge* dalam arsitektur *edge computing*, di mana proses akuisisi data, pemrosesan awal, serta eksekusi logika kontrol dilakukan secara lokal pada sisi *edge* sebelum data diteruskan ke lapisan aplikasi.

Dalam implementasi yang dirancang, RevPi menerima data dari sensor, melakukan pemrosesan sesuai logika yang telah ditentukan, serta mengendalikan aktuator secara langsung berdasarkan hasil pengolahan tersebut.

Selain itu, dukungan antarmuka komunikasi seperti Ethernet serta sistem bus modular (PiBridge dan ConBridge) memungkinkan RevPi berfungsi sebagai node integrasi dalam jaringan IIoT. Dengan kemampuan tersebut, RevPi tidak hanya bertindak sebagai pengendali lokal, tetapi juga sebagai penghubung antara perangkat lapangan dan sistem monitoring berbasis web.

Karakteristik ini memperkuat justifikasi pemilihan Revolution Pi sebagai platform utama pada trainer kit Edge IIoT yang akan dikembangkan nantinya, karena perangkat tersebut mampu merepresentasikan fungsi *edge computing* sekaligus memenuhi kebutuhan integrasi sistem industri secara nyata.

2.1.6 Software Python

Python merupakan bahasa pemrograman interpretatif yang bersifat multiguna dengan filosofi perancangan yang menekankan keterbacaan dan kesederhanaan sintaks. Bahasa ini menggabungkan kapabilitas komputasi yang luas dengan struktur kode yang jelas serta didukung oleh pustaka standar yang komprehensif. Selain itu, Python memiliki komunitas pengembang yang besar sehingga pengembangan dan pemeliharaan sistem dapat dilakukan secara berkelanjutan (Syahrudin & Kurniawan, 2018)**.**

Python mendukung berbagai paradigma pemrograman, seperti pemrograman berorientasi objek, imperatif, dan fungsional. Sebagai bahasa pemrograman dinamis, Python dilengkapi dengan manajemen memori otomatis yang memudahkan proses pengembangan perangkat lunak. Meskipun sering digunakan sebagai bahasa skrip, pemanfaatannya telah berkembang luas pada berbagai konteks, termasuk pengolahan data, pengembangan sistem berbasis jaringan, hingga aplikasi industri.

Saat ini kode python dapat dijalankan di berbagai platform sistem operasi, beberapa diantaranya adalah:

1. Linux/Unix
2. Windows
3. Mac OS X
4. Java Virtual Machine
5. Symbian (untuk produk-produk Nokia), dll.

Selain itu, Python didistribusikan dengan lisensi yang bersifat terbuka dan dapat digunakan secara bebas, termasuk untuk kepentingan komersial, tanpa bertentangan dengan prinsip Open Source maupun General Public License (python, 2019).

Gambar 2.14 Logo Python[7]

2.1.7 Sensor RTD PT1000

Sensor RTD PT1000 merupakan sensor dengan nilai resistansi nominal 1000 Ω pada suhu 0 °C. Sensor ini bekerja berdasarkan perubahan nilai resistansi elemen platinum yang bersifat linier dan stabil terhadap perubahan suhu, sehingga menghasilkan pengukuran suhu yang akurat, presisi, dan repeatable.

Modul ini mendukung konfigurasi 2-wire, 3-wire, dan 4-wire, serta dilengkapi kompensasi kesalahan kabel dan deteksi fault, sehingga sangat sesuai untuk aplikasi industri. Sensor RTD PT1000 memiliki rentang pengukuran suhu yang luas (hingga ±200 °C atau lebih, tergantung sensor) dan tingkat akurasi yang lebih tinggi dibandingkan sensor suhu berbasis termistor atau sensor digital konvensional (Heraeus Sensor Technology, 2016).

Gambar 2.15 Sensor RTD PT1000[8]

[7] python, org. (2019). *The Python Logo | Python Software Foundation*. https://www.python.org/community/logos/

[8] Heraeus Sensor Technology. (2016). *Platinum Resistance Temperature Detector SMD 0603 (V) Heraeus Sensor Technology USA*. www.hst-us.com

2.1.8 Sensor Proximity

Sensor induktif proximity LJ12A3-4-Z/BX merupakan sensor non-kontak yang digunakan untuk mendeteksi keberadaan objek logam pada jarak maksimum 4 mm. Sensor ini menggunakan konfigurasi keluaran NPN *Normally Open* (NO) dengan tegangan kerja 6–36 VDC, sehingga sesuai untuk aplikasi otomasi dan sistem kontrol industri (Datasheet.com, 2019).

Gambar 2.16 Sensor Proximity[9]

2.1.9 Sensor XY-MD02

Sensor XY-MD02 merupakan modul sensor suhu dan kelembapan berbasis RS485 (Modbus RTU) dirancang untuk aplikasi industry. Sensor ini menggunakan sensor digital SHT20 berpresisi tinggi, sehingga mampu menghasilkan pengukuran suhu dan kelembapan yang stabil, akurat, serta memiliki repeatability yang baik. XY-MD02 mendukung protokol komunikasi Modbus RTU standar industri, dengan jarak komunikasi hingga ±1000 meter. Sensor ini memiliki rentang pengukuran suhu −40 hingga 60 °C dengan akurasi ±0,5 °C, serta kelembapan 0–80 %RH dengan akurasi ±3 %RH, dan resolusi data 0,1 (IDB Smart, 2023).

Gambar 2.17 Sensor XY-MD02[10]

[9] Datasheet.com. (2019). *LJ12A3-4-Z ETC2 | Alldatasheet*.

[10] IDB Smart. (2023). *XY-MD02 1.Description*.

2.1.10 Signal Generator 4-20 mA

ProMax CSG01 4–20 mA Current Signal Generator merupakan perangkat simulator sinyal arus standar industri 4–20 mA yang dirancang untuk kebutuhan pengujian, kalibrasi, *commissioning,* dan *troubleshooting* sistem kontrol dan instrumentasi. Perangkat ini berfungsi untuk menggantikan sementara sensor lapangan (seperti *pressure transmitter, temperature transmitter, atau flow transmitter).*

CSG01 mampu menghasilkan sinyal arus 4 hingga 20 mA secara presisi, yang dapat diatur secara manual menggunakan *digital rotary encoder* berakurasi tinggi. Nilai arus keluaran ditampilkan secara *real-time* melalui LCD display dengan resolusi hingga 0,01 mA, sehingga memudahkan pengguna dalam melakukan penyesuaian dan pemantauan sinyal (Probots, 2023).

Gambar 2.18 *Signal Generator ProMax CSG01 4-20mA*[11]

2.1.11 Motor DC

Motor DC 775 merupakan aktuator yang berfungsi mengubah energi listrik menjadi energi mekanik berupa putaran poros. Motor ini bekerja pada tegangan 12–24 VDC, memiliki karakteristik torsi tinggi dan kecepatan putar hingga 12.000 RPM, serta dilengkapi *double ball bearing* dan kipas pendingin internal untuk mendukung kinerja yang stabil pada berbagai aplikasi otomasi dan industri (eBay, 2023).

Gambar 2.19 Motor DC[12]

[11] Probots. (2023). *ProMax CSG01 4-20ma Current Signal Generator*. www.probots.co.in
[12] eBay. (2023). *775 / 795 DC Motor 12V-24V 12000RPM*.

2.1.12 Push Button

Schneider Electric Harmony XA2 Push Button Flush 22 mm 1NC Merah (XA2EA42) merupakan perangkat input digital yang digunakan pada sistem kontrol industri. Push button ini memiliki diameter standar 22 mm dengan desain flush mounting, sehingga mudah dipasang pada panel kontrol dan memberikan tampilan yang rapi. Berdasarkan jenis kontaknya, push button industri umumnya dibedakan menjadi *Normally Open* (NO) dan *Normally Closed* (NC) (Schneider Electric Indonesia, 2025).

Pada tipe *Normally Open* (NO), rangkaian berada dalam kondisi terbuka pada keadaan normal dan akan tertutup saat tombol ditekan. Sebaliknya, pada tipe *Normally Closed* (NC) seperti pada XA2EA42, rangkaian berada dalam kondisi tertutup pada keadaan normal dan akan terbuka ketika tombol ditekan..

Gambar 2.20 *Push Button Schneider* XA2EA42[13]

2.1.13 Selector Switch

Selector switch Schneider Electric Easy Harmony XA2ED33 merupakan sakelar putar industri yang digunakan untuk memilih mode operasi atau kondisi kerja sistem. Switch ini memiliki tiga posisi tetap (*stay put*) dengan konfigurasi kontak 2 *Normally Open* (2NO), diameter pemasangan 22,5 mm, serta tingkat perlindungan IP65 sehingga sesuai digunakan pada panel kontrol industri (Schneider, 2026).

Gambar 2.21 *Selector Switch*[14]

[13] Schneider Electric Indonesia. (2025). *PES_Schneider Electric_Easy-Harmony-XA2_XA2EA42*.

[14] Schneider. (2026). *selector switch - Ø22 - standard handle - 3 positions - 2NO*.

2.1.14 Emergency Button

Selector switch Schneider Electric Easy Harmony XA2ED33 merupakan sakelar putar industri yang digunakan untuk memilih mode operasi atau kondisi kerja sistem. Switch ini memiliki tiga posisi tetap (*stay put*) dengan konfigurasi kontak 2 *Normally Open* (2NO), diameter pemasangan 22,5 mm, serta tingkat perlindungan IP65 sehingga sesuai digunakan pada panel kontrol industri(Omron, 2025).

Gambar 2.22 Emergency Button[15]

2.1.15 LED Lamp Indicator

Pilot Lamp LED 22 mm AD22-22DS (AD2222DS) merupakan lampu indikator panel yang digunakan untuk menampilkan status operasi sistem pada panel kontrol dan sistem otomasi industri. Lampu indikator ini tersedia dalam beberapa warna, yaitu merah, hijau, dan kuning, yang masing-masing umumnya merepresentasikan kondisi *alarm/stop, normal/run, dan warning/standby* (ele-b2b, 2021).

Gambar 2.23 LED[16]

2.1.16 Alarm Buzzer

Pilot Lamp Buzzer LED 22 mm merk EWIG (DC 24 V) merupakan perangkat indikator visual dan audio yang digunakan pada panel kontrol industri untuk memberikan peringatan atau status operasi sistem. Produk ini

[15] Omron. (2025). *Install in 22-dia. or 25-dia. Panel Cutout (When Using a Ring)*.

[16] ele-b2b. (2021). *Description Voltage Contour Indicator Type Color*. www.xindali.com

menggabungkan lampu LED sebagai indikator visual dan buzzer sebagai indikator suara dalam satu unit (Ewig Electronics, 2016).

Gambar 2.24 *Alarm Buzzer*[17]

2.2 Penelitian Terdahulu

Penelitian mengenai implementasi *Edge Industrial Internet of Things* (IIoT) terus berkembang seiring meningkatnya kebutuhan sistem monitoring dan kontrol secara *real-time*. Berbagai penelitian memanfaatkan perangkat *Industrial Edge Computer* untuk melakukan pemrosesan data di sisi *edge* sehingga mampu mengurangi latensi komunikasi dan meningkatkan efisiensi sistem.

Salah satu penelitian dilakukan oleh Halenar et al. (2024), yang mengembangkan sistem *predictive maintenance* berbasis *Edge Computing* menggunakan Revolution Pi sebagai perangkat *edge*. Sistem tersebut melakukan pemrosesan data sensor secara lokal sehingga mampu mengurangi beban komunikasi menuju server pusat. Namun, penelitian ini masih berfokus pada fungsi monitoring kondisi mesin dan belum mengintegrasikan kontrol dua arah maupun antarmuka berbasis web.

Penelitian lain dilakukan oleh Kucera et al. (2022), yang mengembangkan sistem monitoring berbasis Edge IIoT menggunakan PLC industri dan protokol OPC UA. Pendekatan tersebut mampu menjaga interoperabilitas dan keandalan komunikasi industri. Namun, sistem yang dikembangkan masih menggunakan arsitektur komunikasi industri konvensional dan belum mendukung komunikasi berbasis web secara *real-time*.

Selanjutnya, Beňo et al. (2024), yang mengembangkan sistem otomasi

[17] Ewig Electronics. (2016). *Beli EWIG Pilot Lamp Buzzer Red 22mm 1pc | monotaro.id*. https://www.monotaro.id/s030438415.html

berbasis *Edge Computing* menggunakan pendekatan *containerization* pada *industrial PC*. Pendekatan tersebut memberikan fleksibilitas dan skalabilitas dalam proses *deployment*. Namun, kompleksitas sistem dan kebutuhan sumber daya komputasi menjadi lebih tinggi sehingga kurang sesuai untuk media pembelajaran..

Selain penelitian akademik, pengembangan media pembelajaran berbasis Revolution Pi juga telah tersedia dalam bentuk *training kit* komersial. Salah satunya adalah RevPi Next Plus Training Kit yang dikembangkan oleh (techsquare, 2023). *Training kit* tersebut mengintegrasikan Revolution Pi, modul I/O, sensor, aktuator, serta komunikasi industri untuk mendukung proses pembelajaran otomasi dan implementasi Industri 4.0. Fokus utamanya adalah pada konfigurasi perangkat, integrasi sistem, dan komunikasi menggunakan protokol industri standar sehingga peserta didik dapat mempelajari implementasi dasar sistem otomasi industri.

Produk komersial lainnya adalah RevPi Trainingsboard V1206Pi-S1 yang dikembangkan oleh (PLC4Training, 2023). *Training kit* ini menyediakan berbagai komponen praktikum berupa tombol, sakelar, indikator LED, *buzzer*, masukan analog, serta modul simulasi seperti *traffic light* dan *conveyor* yang digunakan untuk pembelajaran, pengujian, dan simulasi menggunakan Revolution Pi maupun CODESYS. Pendekatan tersebut sangat mendukung pembelajaran logika kontrol berbasis PLC, namun implementasinya masih berorientasi pada simulasi kontrol industri konvensional dan belum menitikberatkan pada integrasi komunikasi WebSocket maupun dashboard berbasis web sebagai media monitoring dan kontrol secara *real-time*.

Meskipun berbagai *training kit* tersebut telah mendukung proses pembelajaran otomasi industri, pengembangannya umumnya masih berorientasi pada skenario praktikum yang bersifat tetap (*fixed*). Selain itu, sebagian besar *trainer kit* belum menerapkan arsitektur perangkat lunak yang bersifat modular sehingga pengembangan skenario pembelajaran baru masih memerlukan perubahan program secara menyeluruh. Kondisi tersebut menyebabkan fleksibilitas *trainer kit* sebagai media pembelajaran dan penelitian menjadi terbatas.

Selain aspek perangkat keras, mekanisme komunikasi juga menjadi faktor penting dalam implementasi sistem Edge IIoT. (Pimentel & Nickerson, 2012). menjelaskan bahwa WebSocket menyediakan mekanisme komunikasi *full-duplex*

melalui koneksi TCP persisten sehingga mampu menurunkan latensi komunikasi dibandingkan metode *HTTP polling*. Mekanisme tersebut memungkinkan proses pertukaran data sensor dan perintah kontrol berlangsung secara kontinu tanpa melakukan proses *request-response* secara berulang. Hasil penelitian (Nuratch, 2018) juga menunjukkan bahwa WebSocket efektif diterapkan pada arsitektur *Device-to-Cloud* untuk mendukung proses monitoring dan kontrol dua arah secara *real-time* pada sistem IIoT.

Selain mekanisme komunikasi, sistem monitoring dan kontrol juga memerlukan antarmuka pengguna yang mampu menyajikan data secara interaktif serta mengirimkan perintah kontrol kepada perangkat *edge*. Salah satu platform yang mendukung kebutuhan tersebut adalah NI G Web Development Software. Platform ini memungkinkan pengembangan *dashboard* berbasis web yang dapat diintegrasikan dengan sistem pengukuran dan kontrol berbasis LabVIEW, sehingga proses monitoring dan pengendalian perangkat dapat dilakukan melalui *web browser* tanpa memerlukan aplikasi desktop.

Berdasarkan kajian penelitian terdahulu dapat diketahui bahwa berbagai penelitian telah berhasil mengimplementasikan konsep *Edge Computing*, Revolution Pi, maupun *training kit* berbasis otomasi industri. Namun, penelitian-penelitian tersebut umumnya masih berfokus pada fungsi monitoring atau kontrol secara terpisah, menggunakan komunikasi industri konvensional, serta belum mengintegrasikan komunikasi WebSocket, *dashboard* berbasis web, dan media pembelajaran yang fleksibel dalam satu sistem. Selain itu, implementasi *trainer kit* yang menerapkan arsitektur perangkat lunak berbasis Python secara modular sehingga memungkinkan pengembangan skenario praktikum baru juga masih sangat terbatas.

Oleh karena itu, pada proyek akhir ini dikembangkan *Trainer Kit Edge Industrial Internet of Things (IIoT)* Berbasis Revolution Pi dengan Komunikasi WebSocket dan Dashboard NI G Web sebagai media pembelajaran yang merepresentasikan implementasi sistem industri modern. Trainer kit ini mengintegrasikan Revolution Pi sebagai *Industrial Edge Computer*, berbagai sensor dan aktuator, komunikasi WebSocket, *dashboard* NI G Web, serta sistem basis data dalam satu platform yang mendukung proses monitoring, kontrol, dan

pencatatan data secara *real-time*. Selain menyediakan beberapa skenario praktikum, yaitu *Metal Part Conveyor Monitoring System* dan *Automatic Mixxing Tank Process Control*, trainer kit ini dirancang untuk memberikan pemahaman kepada peserta didik mengenai keseluruhan arsitektur sistem Edge IIoT, mulai dari proses akuisisi data sensor, pemrosesan data pada perangkat *edge*, komunikasi data secara *real-time*, visualisasi melalui *dashboard* berbasis web, hingga penyimpanan data ke dalam basis data (*database*). Data yang tersimpan selanjutnya dapat dimanfaatkan sebagai bahan analisis untuk mengevaluasi performa sistem, seperti pengukuran *latency*, *success rate*, *throughput*, stabilitas komunikasi dan *round trip time analysis*. Di samping itu, trainer kit ini menerapkan arsitektur perangkat lunak berbasis Python yang dirancang secara modular sehingga peserta didik dapat mempelajari pengembangan perangkat lunak industri, melakukan integrasi sensor dan aktuator, membangun mekanisme komunikasi, serta mengembangkan skenario praktikum baru dengan memanfaatkan *baseline* program yang telah disusun tanpa mengubah keseluruhan struktur sistem. Dengan demikian, trainer kit ini tidak hanya berfungsi sebagai media praktikum, tetapi juga sebagai platform pembelajaran yang mendukung penguasaan konsep, implementasi, analisis, dan pengembangan sistem *Edge Industrial Internet of Things* (IIoT) secara menyeluruh.

Dengan pendekatan tersebut, peserta didik tidak hanya mempelajari penggunaan perangkat keras dan perangkat lunak secara terpisah, tetapi juga memahami alur implementasi sistem Edge Industrial Internet of Things (IIoT) secara end-to-end, mulai dari akuisisi data, pemrosesan pada perangkat *edge*, komunikasi data, visualisasi, pengendalian, pencatatan data (*data logging*), hingga analisis performa komunikasi berdasarkan data yang diperoleh.

BAB 3
METODOLOGI PENELITIAN

Dalam pengerjaan proyek akhir ini, perancangan menjadi bagian utama yang menentukan hasil jadi dari keseluruhan proyek akhir. Pada proses ini mencakup beberapa aspek utama dalam perancangan, yaitu:

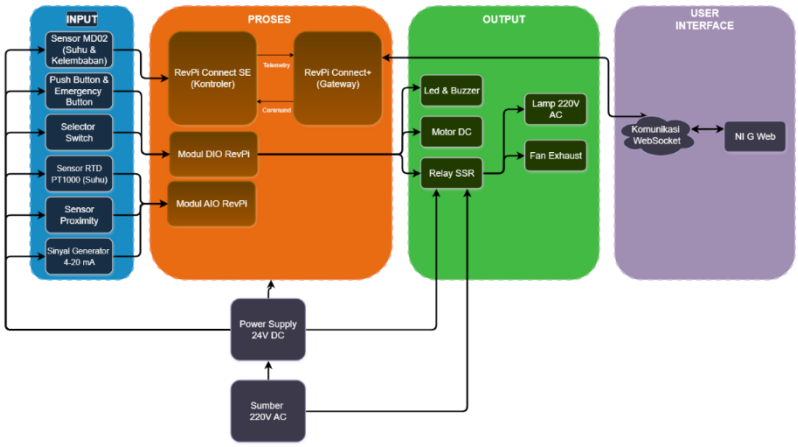
a) Perancangan Sistem, menentukan struktur dan alur kerja sistem secara keselurahan.
b) Perancangan *Hardware*, merancang komponen,desain prototipe dan rangkaian elektronik yang mendukung sistem.
c) Perancangan *Software*, Merancang program komunikasi WebSocket Melalui G-Web serta Python.

3.1 Perancangan Sistem

Dalam perancangan sistem ini dijelaskan secara menyeluruh melalui representasi blok diagram dan *flowchart* untuk menggambarkan struktur, alur kerja, dan hubungan antar komponen dalam sistem yang akan dikembangkan.

3.1.1 Blok Diagram Sistem

Dalam proses perancangan sistem, Blok diagram berfungsi sebagai representasi visual yang mempermudah pemahaman integrasi sistem, sehingga proses analisis, pengembangan, dan *troubleshooting* dapat dilakukan secara terstruktur. Berikut merupakan blok diagram sistem yang dirancang pada proyek akhir ini yang ditampilkan pada Gambar 3.1.



Gambar 3.1 Blok Diagram

Blok diagram pada Gambar 3.1 menunjukkan arsitektur *Trainer Kit Edge Industrial Internet of Things* (IIoT) yang terdiri atas lima bagian utama, yaitu *input*, *proses*, *output*, *user interface*, dan sumber daya. Sistem memperoleh catu daya 24 VDC untuk mengoperasikan Revolution Pi, modul I/O, sensor, dan aktuator DC, sedangkan sumber 220 VAC digunakan untuk beban AC seperti lampu dan *exhaust fan*.

Pada bagian input, sistem menerima data dari sensor MD02, sensor RTD PT1000, sensor *proximity*, *push button*, *emergency button*, *selector switch*, serta *signal generator* 4–20 mA. Seluruh data diproses oleh RevPi Connect SE sebagai *Controller* melalui modul RevPi DIO untuk sinyal digital dan RevPi AIO untuk sinyal analog.

Pada bagian proses, RevPi Connect SE melakukan akuisisi data, menjalankan logika kontrol, dan mengendalikan aktuator. Data *telemetry* kemudian dikirimkan ke RevPi Connect+ sebagai *Gateway*, sedangkan perintah kontrol dari *dashboard* diteruskan kembali ke *Controller* melalui komunikasi WebSocket berbasis JSON.

Pada bagian output, *Controller* mengendalikan LED, *buzzer*, motor DC, serta modul SSR. Modul SSR berfungsi mengendalikan beban AC berupa lampu 220 VAC dan *exhaust fan*.

Pada bagian user interface, NI G Web digunakan sebagai *dashboard* untuk melakukan monitoring dan kontrol sistem secara *real-time*. Komunikasi antara *dashboard* dan *Gateway* dilakukan menggunakan WebSocket sehingga pertukaran data dapat berlangsung secara dua arah dengan latensi yang rendah.

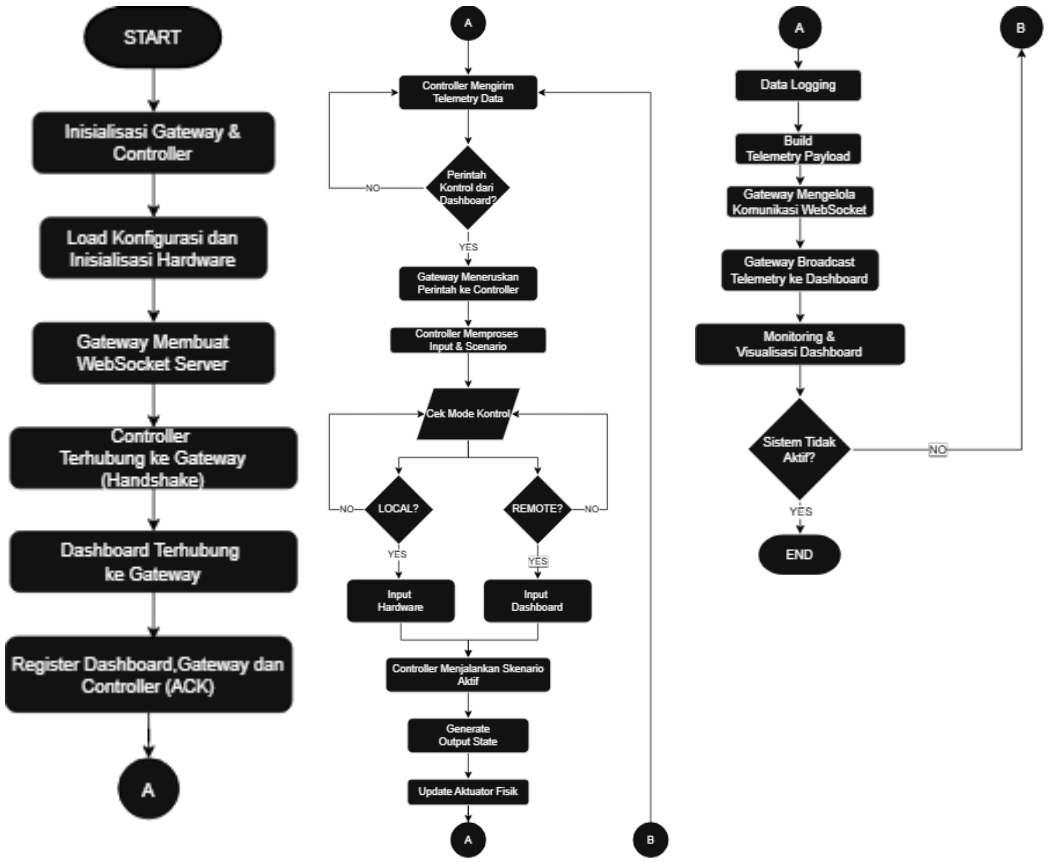
3.1.2 Flowchart (Diagram Alir)

Dalam perancangan sistem, Flowchart digunakan untuk menggambarkan alur kerja dan mekanisme operasi sistem secara visual. Karena sistem menerapkan arsitektur *Controller–Gateway* berbasis Revolution Pi, flowchart dibagi menjadi tiga bagian, yaitu *flowchart* sistem secara keseluruhan, *flowchart* Gateway, dan *flowchart* Controller.

A. *Flowchart* Sistem Secara Keseluruhan

Flowchart sistem secara keseluruhan digunakan untuk menggambarkan alur kerja sistem mulai dari proses inisialisasi perangkat, komunikasi antara Dashboard,

Gateway, dan Controller, pengolahan data, hingga proses monitoring dan pengendalian secara real-time. Flowchart ini ditunjukkan pada Gambar 3.2



Gambar 3.2 Flowchart Sistem Controller–Gateway

Pada Gambar 3.2 menunjukkan flowchart keseluruhan sistem *Trainer Kit Edge Industrial Internet of Things (IIoT).* Proses diawali dengan inisialisasi Gateway dan Controller, dilanjutkan dengan pemuatan konfigurasi sistem, inisialisasi perangkat keras, serta pembentukan WebSocket Server pada Gateway. Setelah itu, Controller dan Dashboard melakukan proses *handshake* dan registrasi sehingga komunikasi antarperangkat dapat berlangsung.

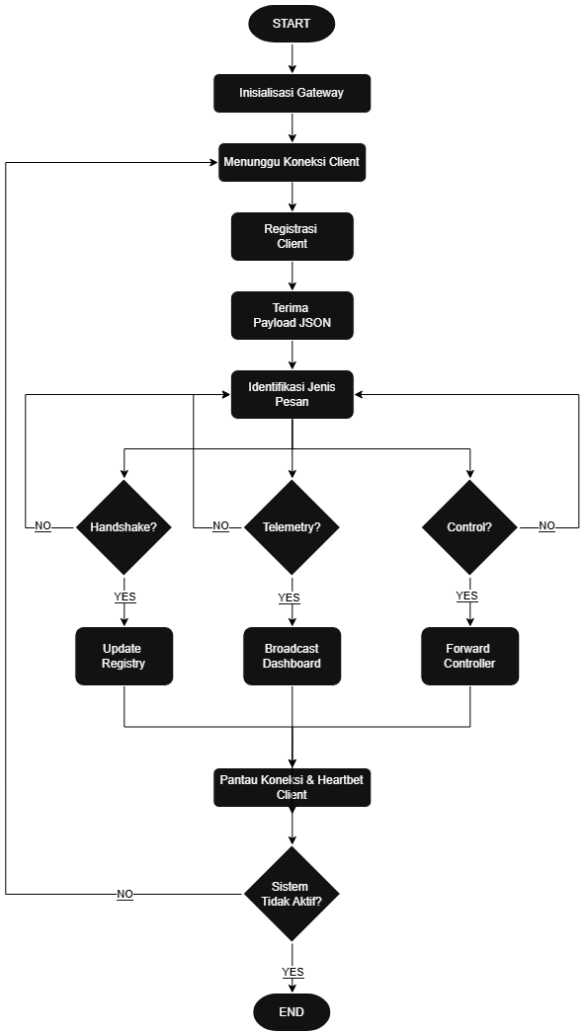
Selanjutnya, Controller mengirimkan data *telemetry* awal ke Gateway. Apabila Dashboard mengirimkan perintah kontrol, Gateway akan meneruskan perintah tersebut kepada Controller untuk diproses sesuai skenario yang sedang aktif. Controller kemudian menentukan mode kontrol, yaitu Local menggunakan masukan perangkat keras atau Remote menggunakan masukan dari Dashboard.

Berdasarkan mode yang dipilih, Controller menjalankan skenario, menghasilkan status keluaran, memperbarui aktuator fisik, melakukan *data*

logging, menyusun *telemetry payload*, kemudian mengirimkan data ke Gateway. Gateway selanjutnya meneruskan data *telemetry* ke Dashboard sehingga kondisi sistem dapat dimonitor secara *real-time*. Seluruh proses tersebut berlangsung secara berulang selama sistem masih aktif. Flowchart ini menggambarkan alur operasional sistem secara *end-to-end,* mulai dari proses inisialisasi, akuisisi data, pengendalian, pencatatan data, komunikasi, hingga visualisasi pada Dashboard.

B. Flowchart RevPi Gateway

Flowchart Gateway menjelaskan proses yang dijalankan oleh RevPi Connect+ sebagai WebSocket Server, meliputi proses inisialisasi, registrasi client, routing data telemetry, penerusan perintah kontrol dari Dashboard ke Controller, serta pengelolaan komunikasi antarperangkat. Flowchart Gateway ditunjukkan pada Gambar 3.3



Gambar 3.3 Flowchart Revpi Gateway

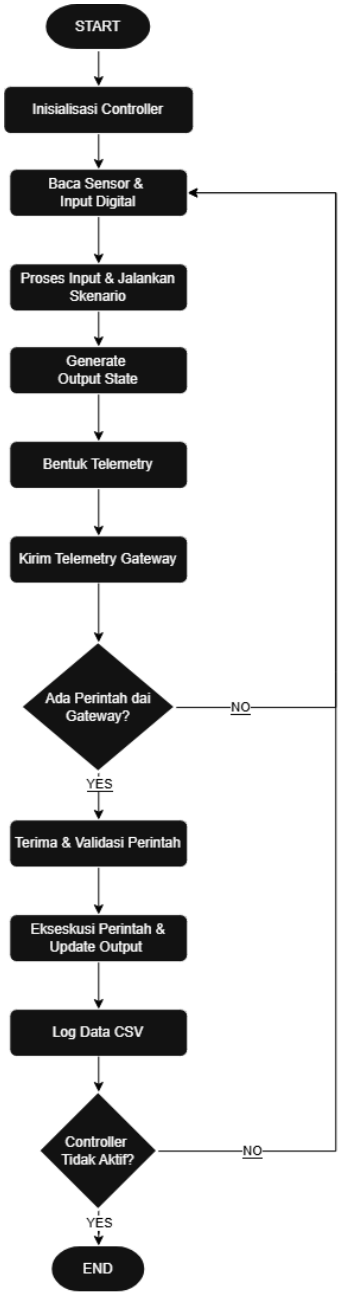
Pada Gambar 3.3 menunjukkan *flowchart* RevPi Gateway yang berfungsi sebagai penghubung komunikasi antara Controller dan Dashboard. Proses diawali dengan inisialisasi Gateway, kemudian Gateway membentuk WebSocket Server dan menunggu koneksi dari client. Setelah Controller atau Dashboard berhasil terhubung, Gateway melakukan registrasi client untuk menyimpan informasi koneksi yang aktif.

Selanjutnya, Gateway menerima *payload* JSON dari client dan mengidentifikasi jenis pesan yang diterima. Apabila pesan berupa *handshake*, Gateway memperbarui *registry* client. Jika pesan berupa *telemetry*, Gateway meneruskan data tersebut ke Dashboard untuk proses monitoring. Sementara itu, apabila pesan berupa *control*, Gateway meneruskan perintah ke Controller untuk diproses lebih lanjut.

Selama sistem berjalan, Gateway terus memantau koneksi setiap client melalui mekanisme *heartbeat* untuk memastikan komunikasi tetap aktif. Seluruh proses tersebut dilakukan secara berulang hingga sistem dihentikan.

C. *Flowchart Controller*

flowchart Controller menggambarkan proses yang dijalankan oleh RevPi Connect SE sebagai Controller, mulai dari pembacaan data sensor dan input digital, pemrosesan logika berdasarkan skenario yang aktif, pengendalian aktuator, pembentukan data telemetry, hingga pengiriman telemetry ke Gateway melalui komunikasi WebSocket. Flowchart Controller ditunjukkan pada Gambar 3.4



Gambar 3.4 *Flowchart* Revpi Controller

Pada Gambar 3.4 menunjukkan *flowchart* RevPi Controller yang berfungsi sebagai pusat akuisisi data, pengolahan logika kontrol, dan pengendalian aktuator pada sistem. Proses diawali dengan inisialisasi Controller, kemudian dilakukan akuisisi data sensor dan pembacaan masukan digital. Data tersebut diproses sesuai skenario yang sedang aktif untuk menghasilkan status keluaran (*output state*). Selanjutnya, Controller memperbarui kondisi aktuator, melakukan *data logging*, menyusun *telemetry payload*, dan mengirimkan data *telemetry* ke Gateway. Apabila

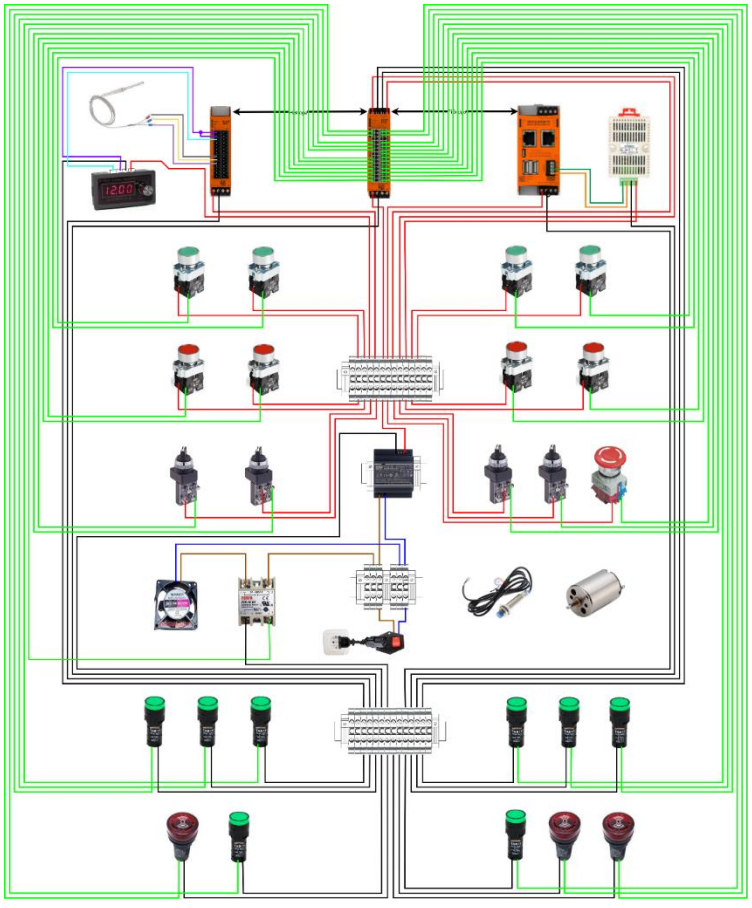
Gateway mengirimkan perintah kontrol, Controller akan menerima, memvalidasi, dan mengeksekusi perintah tersebut sebelum memperbarui kondisi keluaran sistem. Seluruh proses berlangsung secara berulang selama Controller masih aktif.

3.2 Perancangan *Hardware*

Perancangan *hardware* pada *trainer kit Edge IIoT* berbasis Revolution Pi disajikan melalui *wiring* diagram dan desain 3D Prototipe.

3.2.1 Wiring Diagram

Berikut ini merupakan gambaran dari *wiring diagram* yang akan digunakan untuk menghubungkan antar komponen, seperti sensor, modul I/O, Revolution Pi sebagai *edge controller*, serta catu daya sebagai acuan perakitan. Dengan adanya diagram ini, alur distribusi daya dan sinyal dapat dipahami secara jelas.



Gambar 3.5 *Wiring Diagram System*

Gambar 3.5 menunjukkan wiring diagram keseluruhan sistem pada *trainer kit* Edge IIoT berbasis Revolution Pi. Diagram tersebut memperlihatkan hubungan pengkabelan antara sumber catu daya, perangkat masukan (*input*), modul I/O, serta

perangkat keluaran (*output*) yang terintegrasi dalam satu sistem kontrol industri.

Sistem menggunakan Revolution Pi Connect yang terhubung dengan modul RevPi DIO dan RevPi AIO sebagai pusat akuisisi data dan pengendalian perangkat. Modul RevPi AIO digunakan untuk menghubungkan sensor RTD PT1000 beserta perangkat pengukuran suhu, sedangkan modul RevPi DIO digunakan untuk menangani seluruh sinyal digital yang berasal dari push button, selector switch, emergency stop button, dan sensor proximity, serta mengendalikan perangkat keluaran berupa lampu indikator, buzzer, Solid State Relay (SSR), kipas DC, dan motor DC. Seluruh pengkabelan didistribusikan melalui terminal block untuk mempermudah proses instalasi, pengujian, dan pemeliharaan sistem.

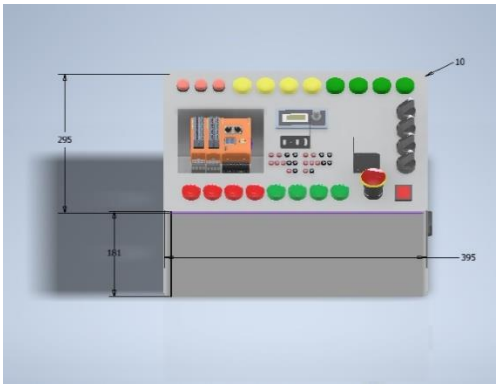
Pengkabelan dibedakan berdasarkan warna untuk mempermudah identifikasi jalur kelistrikan dan sinyal. Jalur catu daya 24 VDC menggunakan warna merah sebagai tegangan positif (+24 V) dan hitam sebagai tegangan negatif (0 V). Jalur sinyal digital input/output menggunakan warna hijau sebagai jalur komunikasi antara perangkat lapangan dengan modul RevPi. Sementara itu, jalur sensor RTD PT1000 menggunakan kombinasi warna sesuai konfigurasi sensor tiga kawat (*3-wire RTD*), sedangkan jalur catu daya AC digunakan sebagai sumber masukan menuju catu daya (*power supply*) sebelum dikonversi menjadi 24 VDC untuk menyuplai seluruh perangkat pada sistem.

Melalui wiring diagram tersebut, seluruh perangkat masukan dan keluaran terintegrasi dengan Revolution Pi sehingga proses akuisisi data, monitoring, dan pengendalian aktuator dapat dilakukan secara real-time berbasis Industrial Internet of Things (IIoT). Selain itu, penggunaan terminal block serta penerapan standar warna kabel bertujuan untuk meningkatkan kejelasan identifikasi jalur, meminimalkan kesalahan pengkabelan, serta mempermudah proses instalasi maupun *troubleshooting*.

3.2.2 3D Design Prototype

Berikut ditampilkan desain 3D prototipe sistem dalam sebuah box koper yang menunjukkan tampilan dari sisi dalam dan sisi atas. Visualisasi ini digunakan untuk memberikan gambaran mengenai tata letak komponen internal serta antarmuka pengguna pada *trainer kit Edge IIoT* berbasis Revolution Pi.

(a) Tampilan Dalam Koper Box



(b) Tampilan Atas Koper Box

Gambar 3.6 (a) Tampilan Dalam Koper Box dan (b) Tampilan Atas Koper Box

Gambar 3.6 (a) menunjukkan tampilan bagian dalam koper box yang memperlihatkan susunan komponen utama sistem. Pada bagian ini terlihat Revolution Pi beserta modul AIO dan DIO, catu daya (PSU), Dinrail, serta dua buah relay yang dipasang secara terstruktur pada dasar box. Penempatan komponen dilakukan untuk memudahkan proses pengkabelan, menjaga kerapian instalasi, serta mendukung kemudahan perawatan dan pengembangan sistem di kemudian hari.

Modul AIO dan DIO pada Revolution Pi menggunakan konektor tipe *pluggable spring clamp terminal block* yang umum digunakan pada sistem otomasi industri 24 VDC. Konektor ini bersifat *removable*, sehingga blok terminal dapat dilepas tanpa membongkar keseluruhan modul. Mekanisme penjepit pegas (*push-in spring clamp*) memungkinkan konduktor terpasang dengan tekanan kontak yang stabil tanpa pengencangan sekrup, sehingga meningkatkan keandalan koneksi dan mengurangi risiko kelonggaran akibat getaran atau penggunaan berulang. Konektor ini mendukung penggunaan kabel dengan penampang sekitar 0,2–1,5 mm² dan dilengkapi tombol pelepas untuk memudahkan proses pemasangan serta pelepasan kabel secara aman dan efisien.

Seluruh komponen dipasang pada rel DIN untuk memastikan struktur pemasangan yang kokoh dan terorganisir, sehingga sistem siap digunakan sebagai trainer kit Edge IIoT yang sesuai dengan praktik instalasi otomasi industri.

Gambar 3.6 (b) menunjukkan tampilan bagian atas koper box yang berfungsi sebagai panel antarmuka pengguna pada *trainer kit Edge IIoT*. Pada bagian ini ditampilkan komponen input dan output yang terhubung langsung dengan modul

AIO, DIO, serta komunikasi Modbus pada Revolution Pi.

Panel atas dilengkapi dengan push button, selector switch, emergency switch, sensor proximity, lampu indikator LED, alarm buzzer. Selain itu, tersedia jack banana yang digunakan sebagai titik koneksi eksternal, sehingga memudahkan pengguna dalam menghubungkan sensor atau sumber sinyal ke trainer kit Edge IIoT secara praktis dan terorganisir.

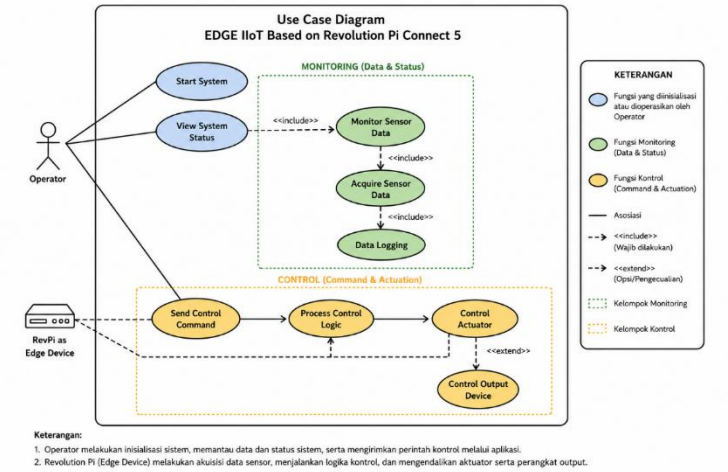
Tata letak komponen pada sisi atas dirancang agar mudah diakses dan dioperasikan dalam kegiatan praktikum, serta memungkinkan pengguna untuk melakukan pengujian berbagai skenario kontrol dan monitoring secara langsung melalui sistem *Edge IIoT* berbasis Revolution Pi.

3.3 Perancangan *Software*

Perancangan *software* bertujuan untuk merancang alur interaksi antara pengguna, sistem monitoring berbasis web, dan perangkat *edge* Revolution Pi. Perancangan ini mencakup pemodelan fungsi sistem, alur komunikasi data, serta tampilan antarmuka pengguna yang digunakan untuk monitoring dan kontrol sistem secara *real-time*.

3.3.1 Use Case Diagram

Use case diagram menggambarkan hubungan antara pengguna (operator) dengan sistem yang dikembangkan. Operator berperan dalam melakukan monitoring kondisi sensor, mengirim perintah kontrol ke aktuator, serta memantau status sistem melalui antarmuka *NI G Web* . Diagram ini menunjukkan fungsi utama sistem yang dapat diakses oleh pengguna secara langsung.



Gambar 3.7 *Use Case Diagram* Sistem

Use case diagram pada Gambar 3.7 menggambarkan interaksi antara operator dan sistem *Industrial Internet of Things (IIoT)* berbasis Revolution Pi Connect SE dalam proses monitoring dan kontrol. Diagram tersebut menunjukkan bahwa operator berperan sebagai pengguna utama yang melakukan pengoperasian sistem, pemantauan kondisi perangkat, serta pengiriman perintah kontrol melalui antarmuka aplikasi.

Proses sistem diawali melalui use case *Start System*, yaitu saat operator mengaktifkan sistem agar seluruh perangkat dan layanan komunikasi siap digunakan. Setelah sistem aktif, operator dapat menjalankan *View System Status* untuk melihat kondisi umum sistem dan status perangkat yang terhubung.

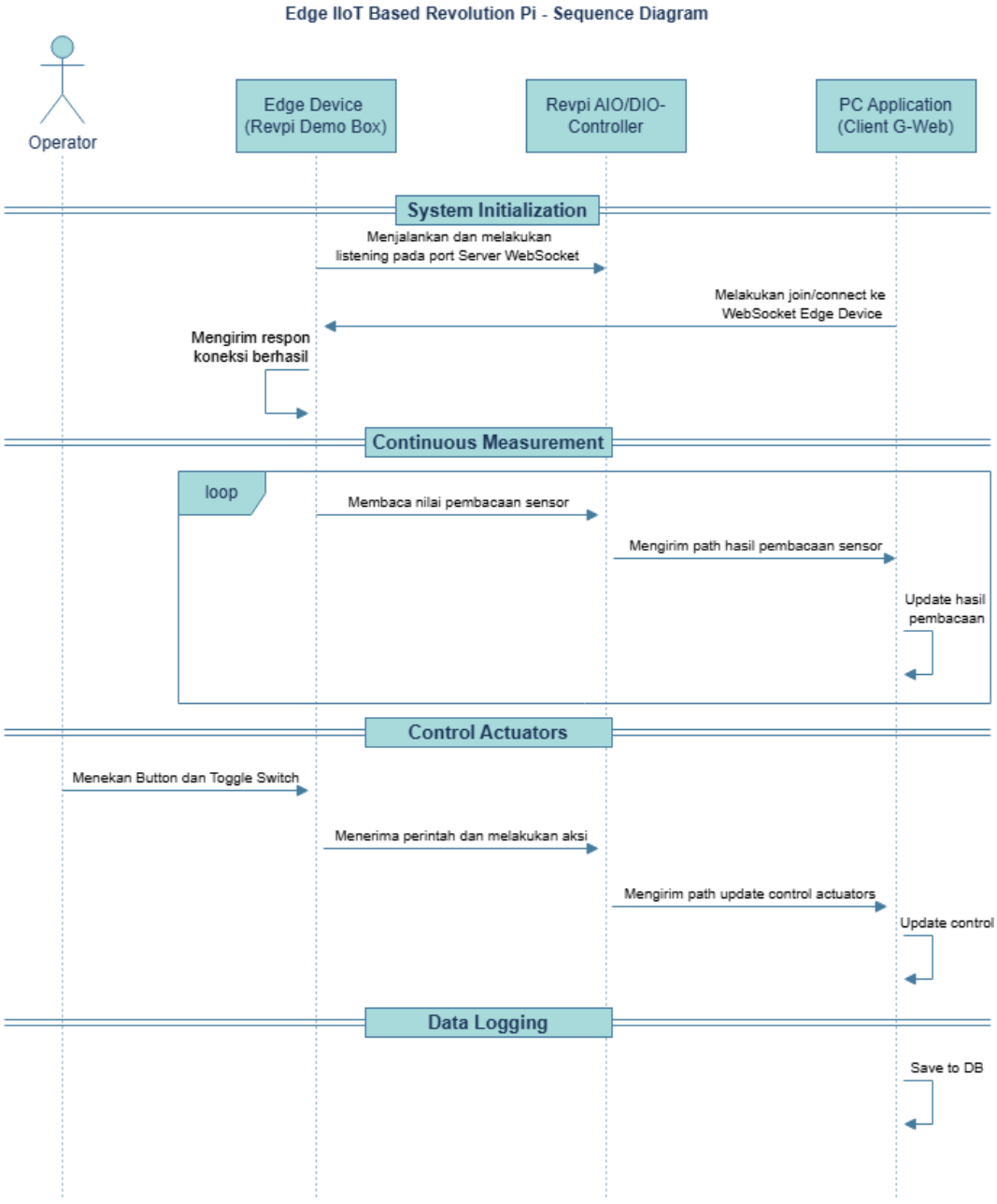
Pada bagian monitoring, sistem menjalankan proses *Monitor Sensor Data* untuk memantau data sensor secara real-time. Proses ini mencakup *Acquire Sensor Data*, yaitu pengambilan data dari sensor dan perangkat lapangan yang terhubung ke Revolution Pi melalui modul input/output. Seluruh data hasil akuisisi kemudian disimpan melalui proses *Data Logging* sebagai dokumentasi dan pencatatan historis data sistem.

Selain monitoring, operator juga dapat melakukan pengendalian perangkat melalui use case *Send Control Command*. Perintah yang dikirimkan operator selanjutnya diproses pada *Process Control Logic* yang dijalankan oleh Revolution Pi sebagai *edge device.* Berdasarkan hasil pemrosesan logika kontrol tersebut, sistem menjalankan *Control Actuator* untuk mengendalikan aktuator atau perangkat output sesuai kondisi dan perintah yang diberikan. Pada kondisi tertentu, proses kontrol juga dapat diteruskan menuju *Control Output Device* sebagai bagian dari pengendalian perangkat keluaran sistem.

Berdasarkan *use case diagram* tersebut, dapat diketahui bahwa Revolution Pi berperan sebagai pusat pemrosesan *edge computing* yang menangani akuisisi data sensor, pemrosesan logika kontrol, monitoring sistem, serta pengendalian aktuator secara terintegrasi, sedangkan operator berperan dalam pengawasan dan pengendalian sistem melalui aplikasi monitoring berbasis web.

3.3.2 Sequence Diagram

Sequence diagram digunakan untuk menggambarkan urutan proses komunikasi antara operator, *NI G Web ,* WebSocket server, dan Revolution Pi.



Gambar 3.8 *Sequence Diagram* Sistem

Sequence diagram pada Gambar 3.8 menggambarkan alur interaksi antara operator, Dashboard NI G Web sebagai client, aplikasi Python berbasis WebSocket Server yang berjalan pada Revolution Pi, serta modul RevPi AIO/DIO yang digunakan untuk akuisisi data dan pengendalian aktuator. Melalui mekanisme ini,

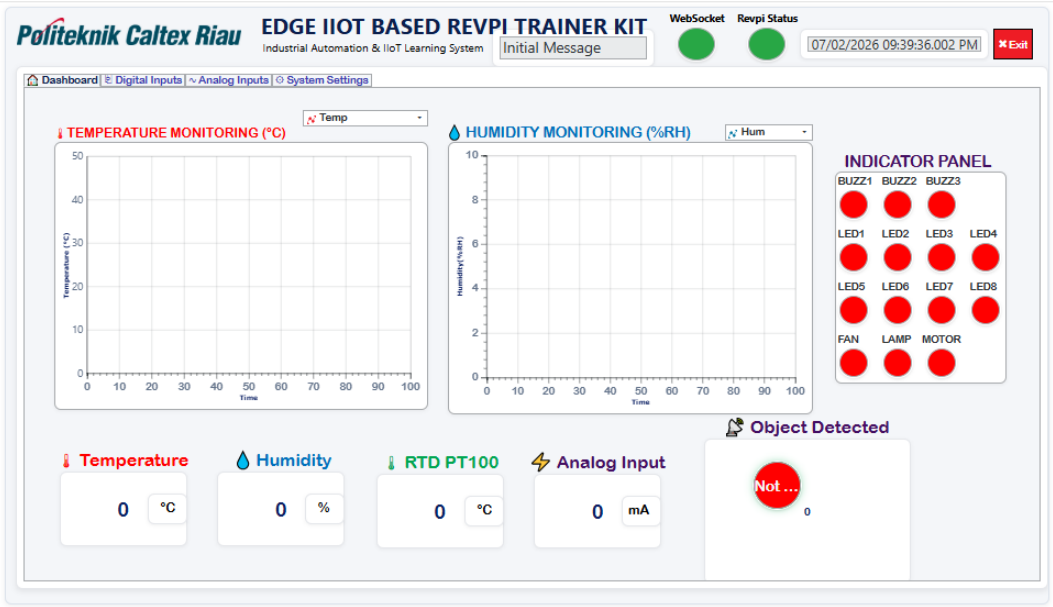
proses monitoring dan kontrol dapat dilakukan secara *real-time* menggunakan komunikasi dua arah berbasis WebSocket.

Proses diawali dengan *system initialization*, di mana RevPi menjalankan layanan WebSocket dan Client *NI G Web* melakukan koneksi hingga sistem siap beroperasi.

Setelah inisialisasi selesai, sistem memasuki mode *continuous measurement*, di mana RevPi secara periodik membaca data dari modul AIO/DIO, memproses data di sisi *edge,* serta mengirimkan data monitoring secara *real-time* ke *Client NI G Web* untuk ditampilkan. Selain itu, operator dapat mengirimkan perintah kontrol melalui Client *NI G Web* yang kemudian diproses oleh RevPi dan dieksekusi ke aktuator melalui modul DIO. Seluruh data hasil pengukuran dan status kontrol dicatat melalui proses *data logging* pada perangkat *edge* untuk mendukung pemantauan dan analisis sistem.

3.3.3 Tampilan *User Interface*

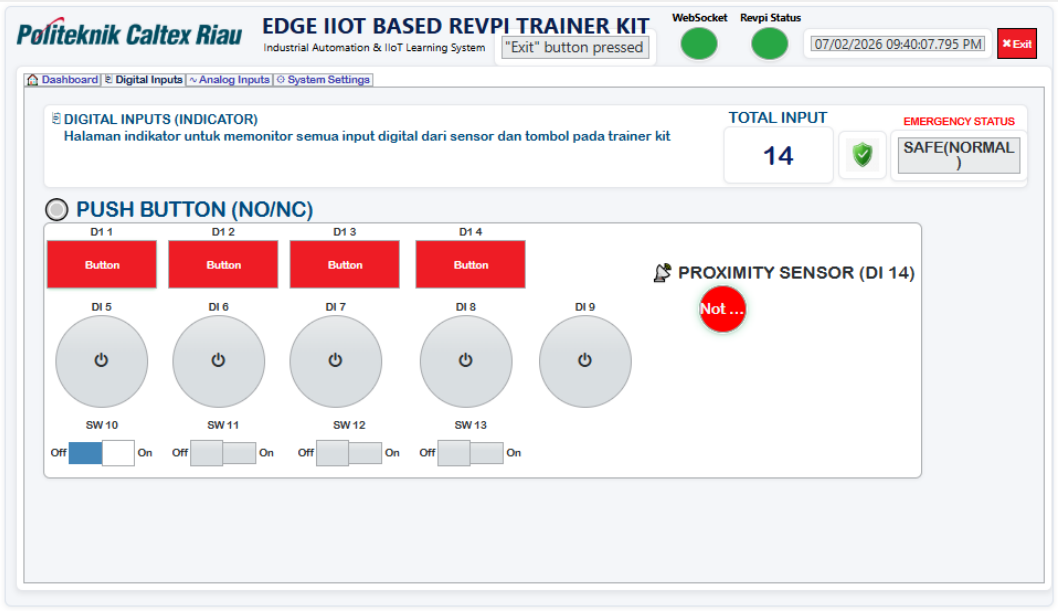
Tampilan antarmuka pengguna dirancang berbasis web menggunakan *NI G Web* untuk menampilkan data monitoring dan kontrol sistem. Antarmuka ini menampilkan informasi sensor, status aktuator, serta tombol kontrol yang memungkinkan operator berinteraksi langsung dengan sistem secara mudah dan respomsif.



Gambar 3.9 *Mockup User Interface (Dashboard Monitoring)*

Gambar 3.9 menunjukkan antarmuka (*interface*) halaman Dashboard yang berfungsi sebagai media monitoring kondisi sistem secara *real-time*. Pada bagian atas antarmuka ditampilkan informasi status komunikasi, meliputi indikator koneksi WebSocket, status Revolution Pi, pesan sistem (*Initial Message*), serta *timestamp* yang menunjukkan waktu pembaruan data terakhir.

Pada bagian utama dashboard terdapat dua grafik monitoring, yaitu Temperature Monitoring (°C) dan Humidity Monitoring (%RH), yang digunakan untuk menampilkan perubahan data sensor terhadap waktu secara *real-time*. Di sisi kanan terdapat Indicator Panel yang menampilkan status perangkat keluaran, meliputi Buzzer 1–3, LED 1–8, serta aktuator Fan, Lamp, dan Motor. Selain itu, dashboard juga menyediakan tampilan numerik untuk parameter Temperature, Humidity, RTD PT100, dan Analog Input (mA) sebagai informasi hasil pembacaan sensor secara aktual. Pada bagian kanan bawah juga disediakan indikator Object Detected yang digunakan untuk menampilkan status deteksi objek dari sensor proximity. Melalui antarmuka ini, pengguna dapat memantau kondisi sensor, status aktuator, serta komunikasi sistem secara terpadu dalam satu halaman dashboard.

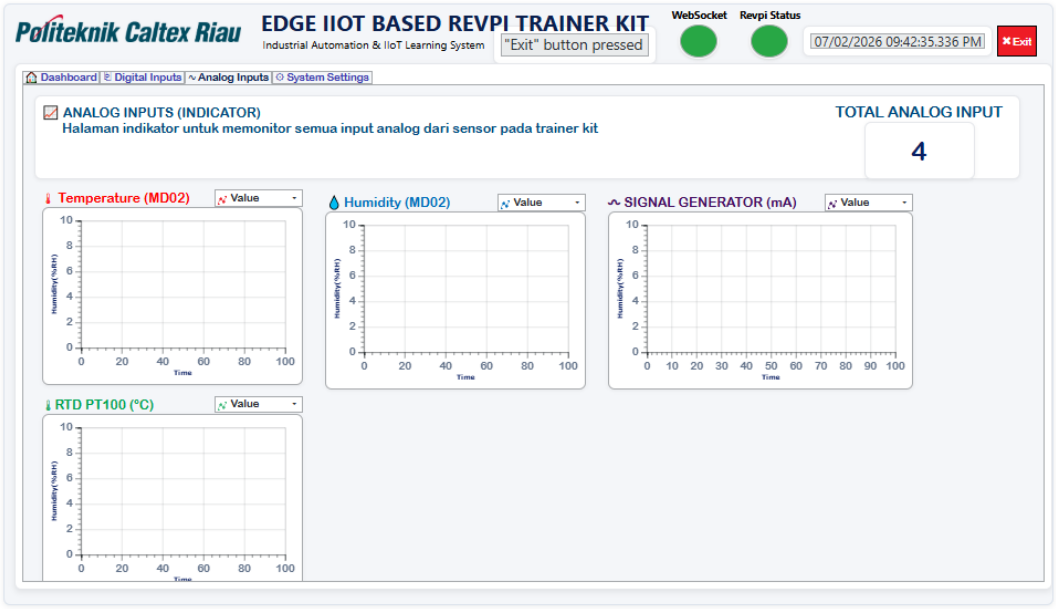


Gambar 3.10 *Mockup User Interface (Digital Inputs)*

Gambar 3.10 menunjukkan antarmuka Digital Inputs yang berfungsi untuk memantau status seluruh perangkat masukan digital pada *trainer kit* secara real-time. Pada bagian atas ditampilkan informasi Total Input yang menunjukkan jumlah masukan digital yang terdeteksi, serta Emergency Status yang menampilkan

kondisi tombol *emergency stop* apakah berada pada kondisi aman (*SAFE/NORMAL*) atau aktif.

Pada bagian utama antarmuka ditampilkan indikator Push Button (NO/NC) yang merepresentasikan status masing-masing tombol masukan digital, terdiri atas empat push button Normally Open (NO), lima indikator push button Normally Closed (NC), serta empat selector switch yang ditampilkan dalam bentuk *toggle switch*. Selain itu, pada sisi kanan tersedia indikator Proximity Sensor yang digunakan untuk menampilkan status deteksi objek secara *real-time*. Melalui antarmuka ini, pengguna dapat memantau kondisi seluruh perangkat input digital secara terintegrasi sehingga mempermudah proses pengujian, monitoring, dan validasi sinyal masukan pada sistem.



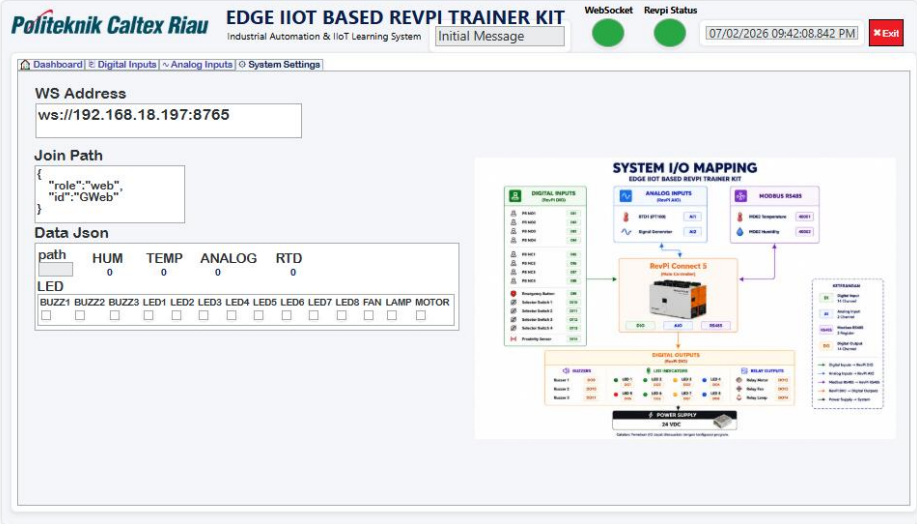
Gambar 3.11 *Mockup User Interface (Analog Inputs)*

Gambar 3.11 menunjukkan antarmuka Analog Inputs yang berfungsi untuk memantau seluruh data masukan analog dari sensor pada *trainer kit* secara real-time. Pada bagian atas ditampilkan informasi Total Analog Input yang menunjukkan jumlah kanal analog yang digunakan pada sistem.

Pada bagian utama antarmuka ditampilkan empat grafik monitoring yang merepresentasikan parameter analog, yaitu Temperature (MD02), Humidity (MD02), Signal Generator (mA), dan RTD PT100 (°C). Setiap grafik menampilkan perubahan nilai sensor terhadap waktu sehingga memudahkan pengguna dalam mengamati respons dan tren data secara kontinu. Selain itu, setiap parameter juga

dilengkapi dengan tampilan nilai (*Value*) sebagai informasi numerik hasil pembacaan sensor secara aktual.

Melalui antarmuka ini, pengguna dapat memantau seluruh masukan analog secara terintegrasi sehingga proses pengujian, monitoring, dan validasi data sensor dapat dilakukan dengan lebih mudah dan real-time.



Gambar 3.12 *Mockup User Interface (System Settings)*

Gambar 3.12 menunjukkan antarmuka System Settings yang digunakan untuk melakukan konfigurasi dan pemantauan komunikasi WebSocket pada sistem. Pada bagian WS Address ditampilkan alamat server WebSocket yang digunakan sebagai media komunikasi antara G Web Dashboard dan Revolution Pi. Selanjutnya, bagian Join Path menampilkan data *handshake* dalam format JSON yang berisi informasi *role* dan *id* sebagai identitas G Web Dashboard saat melakukan koneksi ke Gateway.

Selain itu, pada bagian Data JSON ditampilkan data telemetry yang diterima dari Revolution Pi, meliputi nilai Humidity (HUM), Temperature (TEMP), Analog Input (ANALOG), dan RTD, serta status perangkat keluaran seperti Buzzer 1–3, LED 1–8, Fan, Lamp, dan Motor dalam bentuk indikator *checkbox*. Di sisi kanan antarmuka juga ditampilkan System I/O Mapping yang memberikan gambaran konfigurasi perangkat input, output, analog, dan komunikasi yang terhubung pada *trainer kit*. Melalui halaman ini, pengguna dapat memantau konfigurasi komunikasi serta data yang dipertukarkan antara G Web Dashboard dan Revolution Pi secara real-time.

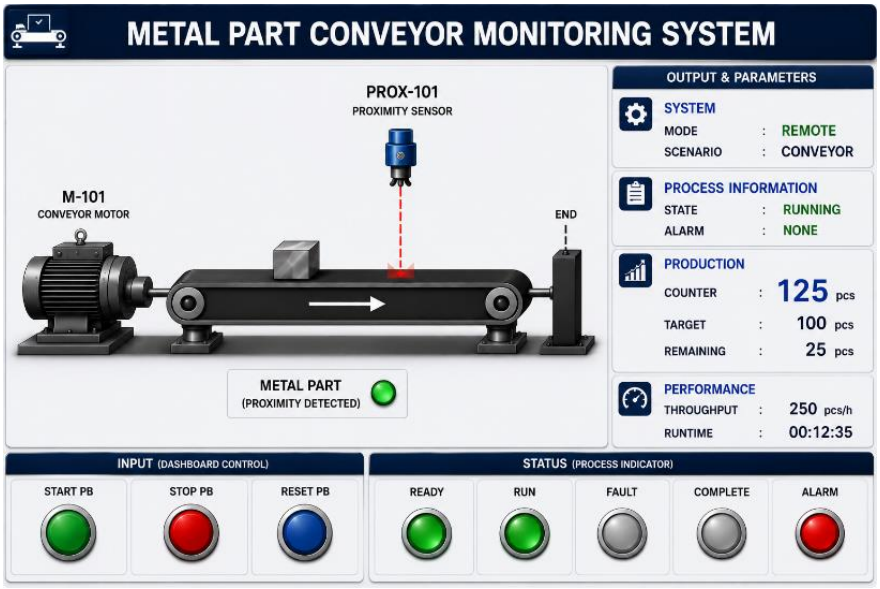
3.4 Skenario Implementasi Trainer Kit Edge IIoT

Trainer Kit Edge Industrial Internet of Things (IIoT) yang dikembangkan dirancang sebagai platform implementasi sistem monitoring dan kontrol *real-time* sekaligus sarana demonstrasi konsep dasar otomasi industri berbasis *edge computing*. Untuk menunjukkan kemampuan sistem, disusun beberapa skenario implementasi yang merepresentasikan fungsi umum industri, *Metal Part Conveyor Monitoring System* dan *Automatic Mixing Tank Process Control System,* Kedua skenario tersebut dipilih karena merepresentasikan tahapan kompleksitas otomasi industri secara berjenjang, mulai dari *material handling & production monitoring*, serta *process automation*, sekaligus mencerminkan penerapan nyata pada sektor manufaktur dan proses

Pada setiap skenario, Revolution Pi berfungsi sebagai perangkat *edge* untuk akuisisi data, pemrosesan logika kontrol, pengendalian aktuator, serta komunikasi data *real-time* menggunakan WebSocket. Informasi kondisi sistem dan status perangkat ditampilkan melalui Dashboard NI G Web sehingga pengguna dapat melakukan monitoring dan kontrol secara langsung melalui antarmuka berbasis web.

3.4.1 Metal Part Conveyor Monitoring System

Metal Part Conveyor Monitoring System mengimplementasikan sistem material handling berbasis conveyor yang dilengkapi fungsi deteksi objek logam, pencacatan produk, monitoring target produksi, event logging, dan telemetry real-time menggunakan Revolution Pi, WebSocket, serta Dashboard NI G Web. Sistem menggunakan pendekatan *Finite State Machine (FSM)* sehingga setiap transisi operasi berlangsung secara terstruktur dan aman. Objek logam yang melewati area deteksi diverifikasi menggunakan sensor proximity induktif, kemudian production counter ditambah satu. Nilai counter dibandingkan dengan target produksi (default 10 pcs); apabila target tercapai, conveyor dihentikan, indikator *Batch Complete* diaktifkan, telemetry dikirim, dan seluruh event dicatat ke dalam log.

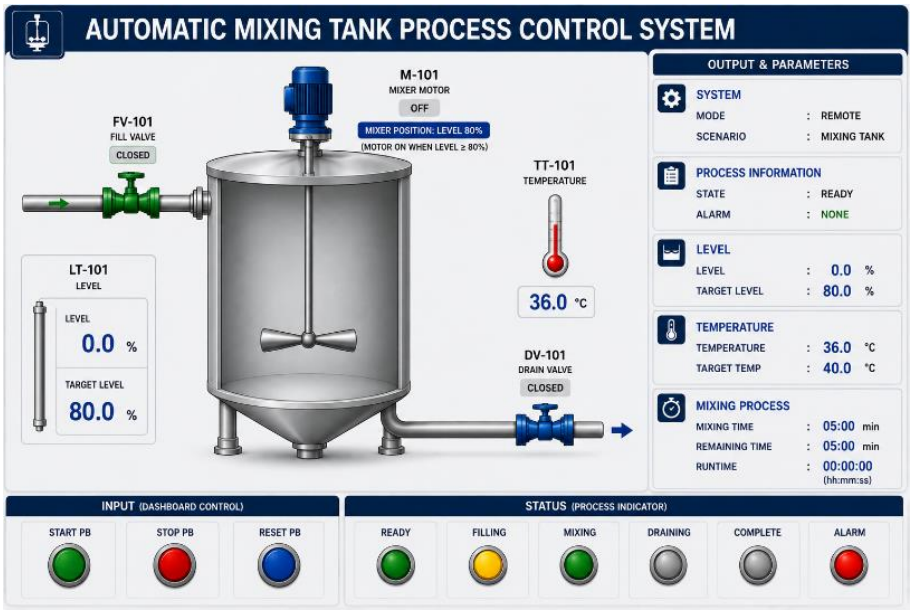


Gambar 3.13 *Case Metal Part Conveyor Monitoring System*

Gambar 3.13 menunjukkan implementasi conveyor monitoring pada *Trainer Kit Edge IIoT*. Status sensor proximity, production counter, dan kondisi aktuator (Conveyor Motor, Ready/Run/Batch Complete/Fault LED, Buzzer) dikirimkan melalui komunikasi WebSocket sehingga dapat dipantau secara real-time melalui Dashboard NI G Web.

3.4.2 Automatic Mixing Tank Process Control System

Automatic Mixing Tank Process Control System mengimplementasikan proses batch skala mini yang mengintegrasikan pengisian tangki, pencampuran, verifikasi temperatur, dan pengosongan tangki menggunakan Revolution Pi sebagai *Edge Controller*. Sistem memanfaatkan LT-101 yang disimulasikan menggunakan Signal Generator 4–20 mA sebagai pembacaan level tangki dan TT-101 berbasis RTD PT100 sebagai pengukuran temperatur proses. Seluruh tahapan dikendalikan menggunakan *Finite State Machine* dan dimonitor secara real-time melalui Dashboard NI G Web menggunakan komunikasi WebSocket.



Gambar 3.14 *Case Automatic Mixing Tank Process Control System*

Gambar 3.14 menunjukkan implementasi mixing tank process control pada Trainer Kit Edge IIoT. Data level tangki, temperatur proses, status valve, status mixer, serta timer proses dikirimkan melalui komunikasi WebSocket sehingga operator dapat memantau dan mengendalikan seluruh tahapan proses secara real-time melalui Dashboard NI G Web.

3.5 Metode Pengujian

Metode pengujian pada proyek akhir ini menggunakan pendekatan ***system testing secara eksperimental***, yaitu pengujian langsung terhadap *trainer kit Edge IIoT* berbasis Revolution Pi yang telah dirancang dan diintegrasikan. Pengujian dilakukan pada jaringan lokal sesuai dengan batasan penelitian dan difokuskan pada evaluasi fungsionalitas serta performa komunikasi sistem. Pengujian disusun berdasarkan rumusan masalah sebagai berikut:

1. Pengujian Akuisisi Data dan Kontrol Aktuator

Pengujian ini bertujuan untuk memverifikasi mekanisme akuisisi data sensor dan eksekusi kontrol pada sisi *edge*. Pengujian akan dilakukan dengan memberikan variasi nilai input pada sensor serta mengamati hasil pembacaan pada Revolution Pi dan dashboard *NI G Web.* Selain itu, perintah kontrol akan diberikan untuk memastikan aktuator merespons sesuai logika yang telah program.

2. Pengujian Komunikasi WebSocket

Evaluasi performa komunikasi WebSocket pada sistem *Edge* IIoT dilakukan dengan menganalisis parameter latency eksekusi kontrol dan tingkat keberhasilan transmisi data (*success rate*). Pengujian ini bertujuan untuk mengetahui waktu respons sistem serta reliabilitas komunikasi dua arah antara Revolution Pi sebagai *edge device* dan Dashboard NI G Web pada jaringan lokal.

Latency dalam proyek akhir ini didefinisikan sebagai selisih waktu antara penerimaan perintah kontrol dan pengiriman telemetry yang telah merefleksikan perubahan status aktuator. Pengukuran dilakukan menggunakan pencatatan timestamp pada sisi *edge* melalui sistem *logging program*, sehingga diperoleh nilai waktu respons aktual sistem.

Selain *latency*, dilakukan pengukuran *success rate* untuk mengetahui tingkat keberhasilan pengiriman dan eksekusi perintah tanpa error. Parameter *throughput*, stabilitas interval pengiriman dan *round trip time analysis* digunakan sebagai parameter pendukung untuk memberikan gambaran tambahan mengenai kapasitas transfer data dan konsistensi siklus monitoring.

1. *Latency* Eksekusi Kontrol

Latency dalam konteks proyek akhir ini didefinisikan sebagai waktu tunda (*delay*) yang terjadi sejak perintah kontrol dikirim hingga sistem memberikan respons yang terverifikasi pada sisi telemetry. Pengujian ini bertujuan untuk mengetahui waktu respons sistem serta reliabilitas komunikasi dua arah antara Revolution Pi sebagai *edge device* dan Dashboard NI G Web pada jaringan lokal.

Pengujian dilakukan pada jaringan lokal (*Local Area Network*) dengan skenario komunikasi dua arah antara Revolution Pi dan Dashboard NI G Web melalui protokol WebSocket. Metode pengujian yang digunakan adalah pencatatan *timestamp* berbasis logging internal pada sisi edge menggunakan bahasa pemrograman Python.

Setiap perintah kontrol yang diterima oleh Revolution Pi dicatat waktu penerimaannya (*t_receive*) menggunakan fungsi pencatatan waktu sistem. Setelah logika kontrol diproses dan aktuator diperbarui, sistem

mengirimkan telemetry terbaru dan mencatat waktu pengirimannya sebagai (*t_update*). Selisih kedua waktu tersebut dihitung sebagai nilai latency eksekusi kontrol.

Data hasil pencatatan timestamp disimpan dalam file log dan dianalisis secara kuantitatif untuk memperoleh nilai rata-rata latency, serta untuk menghitung tingkat keberhasilan eksekusi perintah (*success rate*). Analisis dilakukan menggunakan perhitungan matematis berdasarkan jumlah percobaan yang dilakukan pada kondisi operasi normal sistem.

$$\boldsymbol{Latency_{control,i} = t_{update,i-}\ t_{receive,i}} \quad 3.1$$

dimana:

- $Latency_{control,i}$ = latency pada perintah kontrol ke-i (ms)
- $t_{receive,i}$ = waktu penerimaan perintah kontrol ke-i
- $t_{update,i}$ = waktu pengiriman telemetr yang memuat status terbaru

Jika waktu dicatat dalam detik, maka dikonversi menjadi milidetik:

$$\boldsymbol{Latency_{control,i}(ms) = (t_{update,i-}\ t_{receive,i}) \times 1000} \quad 3.2$$

2. Success Rate Kontrol

Mengukur reliabilitas eksekusi perintah control

$$\boldsymbol{SR_{control}}\,(\%) = \frac{\boldsymbol{N_{executed}}}{\boldsymbol{N_{received}}} \times \boldsymbol{100\ \%} \quad 3.3$$

dimana:

- $\boldsymbol{N_{received}}$ = jumlah perintah kontrol yang diterima
- $\boldsymbol{N_{executed}}$ = jumlah perintah yang berhasil dieksekusi

$$\boldsymbol{SR_{control}} = \boldsymbol{100}(\%) \quad 3.4$$

3. Success Rate Telemetry

Mengukur keberhasilan pengiriman data periodik.

$$\boldsymbol{SR_{telemetry}}(\%) = \frac{\boldsymbol{N_{success}}}{\boldsymbol{N_{attempt}}} \times \boldsymbol{100}\% \quad 3.5$$

dimana:

- $\boldsymbol{N_{attempt}}$ = jumlah total siklus pengiriman telemetry
- $\boldsymbol{N_{success}}$ = jumlah pengiriman tanpa error (tanpa exception broadcast)

4. Throughput Uplink

Throughput dihitung berdasarkan ukuran payload yang dikirim setiap interval.

$$\boldsymbol{U_{Pkbps}(t) = \frac{8 \times bytes_{sent}(t)}{1000 \times \Delta t}} \quad 3.6$$

Karena interval pengiriman:

$$\boldsymbol{\Delta t = 1\ detik} \quad 3.7$$

Maka disederhanakan menjadi:

$$\boldsymbol{U_{Pkbps}(t) = \frac{8 \times bytes_{sent}(t)}{1000}} \quad 3.8$$

Untuk nilai rata-rata throughput:

$$\boldsymbol{Mean_{throughput} = \frac{1}{n}\sum_{i=i}^{n} U_{Pkbps}(i)} \quad 3.9$$

5. Stabilitas Interval Pengiriman Δt

Mengukur konsistensi siklus monitoring.

$$\boldsymbol{\Delta t_i = t_{send,i-}\ t_{send,i-1}} \quad 3.10$$

Rata-rata interval:

$$\boldsymbol{Mean_{\Delta t} = \frac{1}{n}\sum_{i=i}^{n} \Delta t_i} \quad 3.11$$

Standar deviasi:

$$\boldsymbol{\sigma\Delta t = \sqrt{\frac{1}{n}\sum_{i=i}^{n}(\Delta t_i - Mean_{\Delta t})^2}} \quad 3.12$$

Jika nilai standar deviasi kecil, maka sistem berjalan deterministik.

6. Round Trip Time Analysis

Pengujian *Round Trip Time (RTT)* dilakukan untuk mengetahui waktu respons komunikasi secara menyeluruh (*end-to-end communication delay*) pada sistem *Trainer Kit Edge Industrial Internet of Things (IIoT)* berbasis komunikasi WebSocket. Metode pengukuran ini mengadaptasi pendekatan yang digunakan oleh (Hlayel et al., 2025) dalam penelitian *Latency Analysis of WebSocket and Industrial Protocols in Real-Time Digital Twin Integration*, yang mendefinisikan RTT sebagai waktu tempuh pulang-pergi sejak suatu perintah dikirim oleh *client* hingga respons yang telah diproses diterima kembali oleh *client*.

Pengukuran RTT memberikan gambaran yang lebih komprehensif mengenai performa komunikasi WebSocket karena mempertimbangkan keseluruhan jalur komunikasi dua arah (*uplink* dan *downlink*). Oleh karena itu, parameter ini digunakan untuk mengevaluasi kualitas respons sistem yang

dirasakan secara langsung oleh pengguna (*user-perceived response time*) pada setiap skenario aplikasi yang diimplementasikan.

Nilai RTT untuk setiap pengujian dihitung menggunakan Persamaan (3.13).

$$RTT\_i\ (ms) = (t\_response,i - t\_command,i) \times 1000 \quad (3.13)$$

dengan:

- RTT_i = nilai Round Trip Time pada pengujian ke-i (ms)
- $t\_command,i$ = waktu saat perintah kontrol dikirim oleh dashboard
- $t\_response,$ = waktu saat telemetry atau respons diterima kembali oleh dashboard

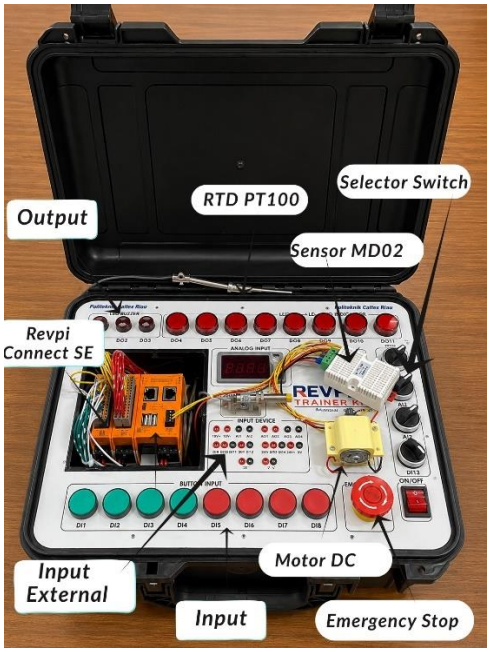
BAB 4
HASIL DAN ANALISIS

4.1 Hasil Implementasi *Trainer Kit Edge IIoT*

Pada bab ini disajikan hasil implementasi *Trainer Kit Edge Industrial Internet of Things (IIoT)* berbasis Revolution Pi yang telah dirancang pada tahap sebelumnya. Implementasi sistem dilakukan melalui integrasi perangkat keras, pengembangan perangkat lunak berbasis Python, serta pembangunan dashboard monitoring dan kontrol menggunakan NI G Web Development Software sehingga membentuk suatu *platform Edge IIoT* yang mampu melakukan akuisisi data, pemrosesan data pada perangkat *edge*, komunikasi dua arah berbasis WebSocket, serta monitoring dan kontrol secara *real-time.*

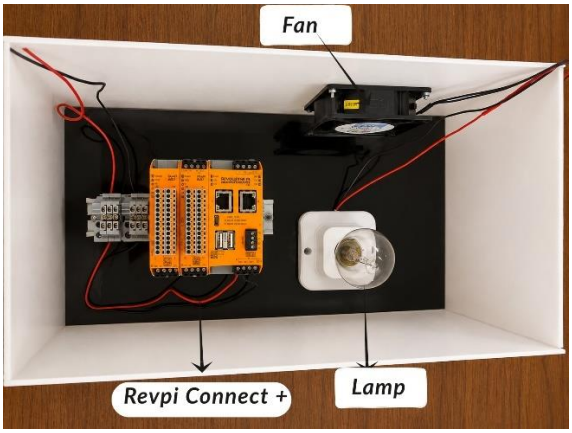
4.1.1 Implementasi Hardware Trainer Kit Edge IIoT

Pada tahap implementasi, *Hardware Trainer Kit Edge Industrial Internet of Things (IIoT)* berhasil direalisasikan sesuai dengan rancangan sistem yang telah dijelaskan pada BAB III. Implementasi hardware terdiri atas dua unit utama, yaitu RevPi Controller yang direpresentasikan dalam bentuk koper portabel (*portable trainer kit*) serta RevPi Gateway yang direpresentasikan dalam bentuk *chamber* simulasi proses. Kedua unit tersebut saling terhubung melalui jaringan Ethernet menggunakan komunikasi berbasis WebSocket sehingga membentuk arsitektur *Edge IIoT* yang mendukung proses monitoring dan kontrol secara *real-time*.



Gambar 4.1 Hardware *Trainer Kit Edge IIoT* (Revpi Controller)

RevPi Controller pada Gambar 4.1 mengintegrasikan berbagai perangkat input dan output dalam satu koper portabel sehingga seluruh proses praktikum dapat dilakukan tanpa memerlukan instalasi tambahan. Perangkat input terdiri atas push button, selector switch, emergency stop, dan terminal koneksi sensor, sedangkan perangkat output meliputi LED indikator, buzzer, serta indikator analog. RevPi Controller berfungsi sebagai perangkat *edge* yang melakukan akuisisi data, menjalankan logika kontrol, serta mengendalikan perangkat input dan output pada *trainer kit*. Desain koper portabel memberikan kemudahan dalam proses penggunaan, penyimpanan, dan mobilitas sehingga *trainer kit* dapat digunakan pada kegiatan praktikum maupun demonstrasi.



Gambar 4.2 Hardware *Trainer Kit Edge IIoT* (Revpi Gateway)

Sedangkan, pada Gambar 4.2 menunjukkan implementasi hardware RevPi Gateway yang ditempatkan di dalam *chamber* simulasi proses. *Chamber* digunakan sebagai tempat pemasangan Revolution Pi beserta komponen pendukung yang berfungsi sebagai pusat komunikasi sistem. Selain itu, chamber juga menjadi media simulasi proses dengan menyediakan ruang untuk pemasangan sensor sehingga proses akuisisi data dapat dilakukan sebelum dikirimkan ke dashboard melalui komunikasi berbasis WebSocket.

Gambar 4.3 Integrasi Hardware *Trainer Kit Edge IIoT* (Gateway-Controller)

Sehingga, Hasil implementasi pada Gambar 4.3 menunjukkan bahwa RevPi Controller dan RevPi Gateway telah berhasil diintegrasikan menjadi satu sistem *Trainer Kit Edge IIoT*. RevPi Controller berperan dalam proses akuisisi data dan pengendalian perangkat, sedangkan RevPi Gateway berfungsi sebagai penghubung komunikasi antara controller dan dashboard. Arsitektur tersebut juga mempermudah proses pengembangan, pemeliharaan, serta pengujian komunikasi karena fungsi akuisisi data dan fungsi gateway dipisahkan ke dalam dua perangkat yang berbeda.

4.1.2 Implementasi Software Python

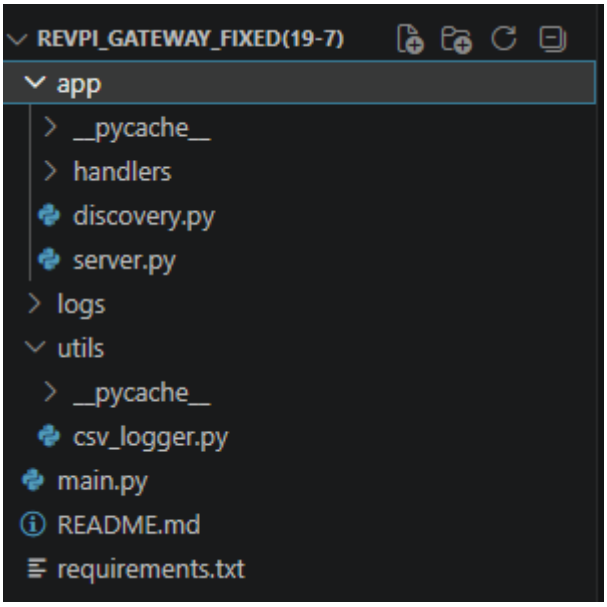
Implementasi perangkat lunak pada *Trainer Kit Edge Industrial Internet of Things (IIoT)* dikembangkan menggunakan bahasa pemrograman Python yang dijalankan pada Revolution Pi. Perangkat lunak dirancang menggunakan pendekatan modular agar setiap komponen sistem memiliki tanggung jawab yang spesifik, sehingga proses pengembangan, pemeliharaan, maupun pengujian dapat dilakukan secara lebih terstruktur. Berdasarkan arsitektur sistem yang telah dirancang pada Bab III, implementasi perangkat lunak dibagi menjadi dua aplikasi

utama, yaitu RevPi Gateway dan RevPi Controller. RevPi Gateway berfungsi sebagai pusat layanan komunikasi yang menghubungkan Dashboard NI G Web dengan perangkat *edge*, sedangkan RevPi Controller bertugas menjalankan proses akuisisi data sensor, pengendalian aktuator, serta eksekusi skenario operasi pada *trainer kit*. Selain kedua aplikasi tersebut, sistem juga dilengkapi dengan mekanisme *logging* yang digunakan untuk mendokumentasikan aktivitas sistem selama proses monitoring dan pengendalian berlangsung. Implementasi masing-masing komponen perangkat lunak dijelaskan sebagai berikut.

Seluruh komponen perangkat lunak tersebut diintegrasikan dalam satu *framework* yang berfungsi sebagai *base application pada Trainer Kit Edge Industrial Internet of Things (IIoT).* Framework ini menyediakan layanan dasar sistem, meliputi komunikasi WebSocket, akuisisi data sensor, pengendalian aktuator, pengelolaan mode operasi *Local* maupun *Remote*, Melalui mekanisme *dispatcher*, framework mampu memilih dan menjalankan logika proses sesuai skenario operasi yang diaktifkan tanpa mengubah arsitektur utama perangkat lunak. Dengan pendekatan tersebut, framework yang dikembangkan dapat digunakan kembali (*reusable*) untuk berbagai aplikasi industri. Pada proyek akhir ini, implementasi framework menghasilkan dua skenario aplikasi, yaitu *Metal Part Conveyor Monitoring dan System, Automatic Mixing Tank Process Control System.* Dokumentasi implementasi masing-masing skenario, meliputi struktur program, diagram proses, antarmuka pengguna, serta kode sumber perangkat lunak, disediakan pada repositori GitHub peneliti sebagai dokumentasi pendukung (GitHub - Banscript-Ssh/Websocket · GitHub, 2026).

A. Revpi sebagai Gateway

RevPi Gateway merupakan aplikasi yang berfungsi sebagai pusat layanan komunikasi pada sistem *Trainer Kit Edge IIoT*. Aplikasi ini dijalankan pada Revolution Pi dan bertanggung jawab mengelola layanan WebSocket, menerima koneksi dari perangkat yang terhubung, serta menyediakan mekanisme pertukaran data antara RevPi Controller dan Dashboard NI G Web. Agar fungsi-fungsi tersebut dapat diimplementasikan secara terstruktur, aplikasi Gateway dibangun menggunakan beberapa modul yang saling terintegrasi sebagaimana ditunjukkan pada Gambar 4.4**.**



Gambar 4.4 Struktur Direktori Folder Revpi Gateway

Struktur direktori RevPi Gateway ditunjukkan pada Gambar 4.4**.** Berdasarkan gambar tersebut, aplikasi Gateway terdiri atas beberapa direktori dan berkas program yang saling terintegrasi untuk membentuk layanan komunikasi pada sistem. Setiap direktori memiliki fungsi yang berbeda, mulai dari proses inisialisasi aplikasi, pengelolaan layanan WebSocket, pemrosesan pesan, hingga penyimpanan konfigurasi pendukung.

Meskipun keseluruhan modul saling berhubungan dalam menjalankan aplikasi Gateway, pembahasan pada proyek akhir ini difokuskan pada tiga berkas utama, yaitu main.py, server.py, dan messages.py. Ketiga berkas tersebut dipilih karena merupakan inti implementasi perangkat lunak yang bertanggung jawab terhadap proses inisialisasi aplikasi, pengelolaan layanan komunikasi, serta pemrosesan data yang dipertukarkan selama sistem beroperasi. Adapun implementasi masing-masing berkas dijelaskan sebagai berikut.

1. main.py

File main.py merupakan titik awal (*entry point*) aplikasi RevPi Gateway. Modul ini bertanggung jawab melakukan inisialisasi aplikasi, membangun objek layanan Gateway, mengaktifkan layanan WebSocket, serta menjaga agar aplikasi tetap berjalan selama sistem beroperasi. Selain itu, modul ini juga mengatur proses penghentian aplikasi secara aman (*graceful shutdown*) ketika sistem dihentikan.

Implementasi utama pada file main.py dijelaskan melalui beberapa tahapan sebagai berikut.

Script Program Import Library

```
import asyncio
import time
from app.server import WebSocketGatewayServer
```

Tahap pertama pada file main.py adalah mengimpor pustaka (*library*) yang dibutuhkan untuk menjalankan aplikasi RevPi Gateway. Library asyncio digunakan untuk mendukung mekanisme pemrograman asinkron (*asynchronous programming*) sehingga beberapa proses dapat dijalankan secara bersamaan tanpa saling menghambat. Library time digunakan untuk mendukung proses pencatatan waktu selama aplikasi berjalan, sedangkan kelas WebSocketGatewayServer diimpor dari file server.py sebagai komponen utama yang bertanggung jawab menyediakan layanan komunikasi WebSocket. Setelah seluruh pustaka berhasil dimuat, tahap berikutnya adalah melakukan inisialisasi layanan Gateway.

Script Program Inisialisasi Gateway Server

```
async def main():
    server = WebSocketGatewayServer(
        host="0.0.0.0",port=8765
    )
    await server.start()
```

Setelah seluruh pustaka berhasil dimuat, aplikasi memasuki tahap inisialisasi RevPi Gateway. Pada tahap ini dibuat sebuah objek WebSocketGatewayServer yang berfungsi sebagai layanan utama komunikasi pada sistem. Objek tersebut dikonfigurasi menggunakan alamat host 0.0.0.0 dan port 8765 sehingga Gateway dapat menerima koneksi dari seluruh perangkat yang berada pada jaringan yang sama. Selanjutnya, fungsi start() dipanggil untuk mengaktifkan layanan WebSocket sehingga Gateway berada pada kondisi siap menerima koneksi dari RevPi Controller maupun Dashboard NI G Web. Setelah layanan berhasil dijalankan, aplikasi memasuki kondisi aktif dan mempertahankan proses komunikasi selama sistem beroperasi.

Script Program Running Gateway Server

```
    try:
        while True:
            await asyncio.sleep(3600)
    finally:
        heartbeat_task.cancel()
```

```
        await server.stop()
```

Setelah layanan WebSocket berhasil diaktifkan, aplikasi memasuki kondisi *running* untuk mempertahankan layanan Gateway tetap aktif selama sistem beroperasi. Proses ini dilakukan menggunakan perulangan yang berjalan secara terus-menerus sehingga layanan komunikasi tidak berhenti selama tidak terdapat perintah penghentian aplikasi. Ketika proses dihentikan, Gateway membatalkan proses pemantauan client (*heartbeat task*), kemudian menutup seluruh koneksi WebSocket melalui fungsi stop(). Mekanisme tersebut menerapkan konsep *graceful shutdown*, yaitu proses penghentian aplikasi secara aman sehingga seluruh koneksi dapat ditutup dengan baik tanpa menyebabkan gangguan pada komunikasi yang sedang berlangsung.

2. server.py

Setelah proses inisialisasi aplikasi selesai dilakukan oleh main.py, tahap berikutnya adalah membangun layanan komunikasi yang menjadi fungsi utama RevPi Gateway. Layanan tersebut diimplementasikan pada file server.py yang berperan sebagai modul inti dalam mengelola komunikasi berbasis WebSocket. Modul ini bertanggung jawab menangani koneksi perangkat, melakukan registrasi client, mengatur proses distribusi data, serta mengelola siklus hidup layanan WebSocket selama sistem beroperasi.

Script Program Inisialisasi WebSocket Gateway

```
class WebSocketGatewayServer:
    def __init__(
        self,
        host="127.0.0.1",port=8765
    ):
        self.host = host
        self.port = port
        self.controllers = {}
        self.web_clients = set()
        self._clients = set()
        self.command_counter = 0
```

Tahap pertama pada file server.py adalah melakukan inisialisasi layanan WebSocket Gateway melalui kelas WebSocketGatewayServer. Pada tahap ini dilakukan konfigurasi alamat host dan port yang digunakan sebagai identitas layanan komunikasi. Selain itu, Gateway juga menyiapkan beberapa struktur data untuk menyimpan daftar RevPi Controller, Dashboard NI G Web, serta seluruh client yang sedang terhubung ke sistem. Variabel command_counter digunakan

sebagai pencacah (*counter*) untuk menghasilkan identitas (*correlation id*) yang unik pada setiap perintah kontrol sehingga proses pelacakan komunikasi dapat dilakukan dengan lebih mudah. Setelah konfigurasi dasar Gateway selesai dilakukan, tahap berikutnya adalah menangani setiap koneksi perangkat yang masuk ke layanan WebSocket.

Script Program Handler Koneksi Client

```
async def handler(
    self,websocket,path=None
):
    peer = websocket.remote_address
    self._clients.add(websocket)
    init_msg = await websocket.recv()
```

Setelah layanan WebSocket berhasil diinisialisasi, Gateway memasuki tahap penanganan koneksi client melalui fungsi handler(). Fungsi ini akan dipanggil secara otomatis setiap kali terdapat perangkat baru yang membangun koneksi dengan Gateway. Pada tahap ini Gateway menyimpan informasi alamat perangkat (*peer address*), kemudian menambahkan koneksi tersebut ke dalam daftar client aktif sebagai bagian dari proses pemantauan koneksi. Selanjutnya, Gateway menerima pesan awal (*initial message*) yang dikirimkan oleh perangkat. Pesan tersebut digunakan sebagai informasi identitas awal yang akan diproses pada mekanisme handshake sebelum komunikasi data dilakukan.

Script Program Registrasi Client (*Handshake)*

```
data = json.loads(init_msg)
role = data.get("role")
client_id = data.get("id")
if role == "controller":
    self.controllers[client_id] = websocket
elif role == "web":
    self.web_clients.add(websocket)
async def broadcast_to_web(self, payload):
    for ws in self.web_clients:
        await ws.send(payload)
```

Tahap berikutnya adalah melakukan registrasi perangkat melalui mekanisme handshake. Pada tahap ini Gateway memproses pesan awal yang diterima dengan mengambil informasi role dan id dari setiap perangkat yang terhubung. Informasi tersebut digunakan untuk mengidentifikasi jenis client sehingga Gateway dapat membedakan antara RevPi Controller dan Dashboard NI G Web. Apabila perangkat berperan sebagai Controller, koneksi akan disimpan ke

dalam daftar controller berdasarkan identitas perangkat. Sebaliknya, apabila perangkat berperan sebagai Dashboard, koneksi akan disimpan sebagai web client. Melalui proses registrasi ini, Gateway memperoleh informasi yang diperlukan untuk menentukan tujuan pengiriman data pada proses komunikasi berikutnya.

Script Program Routing Telemetry and Command

```
async def send_to_controller(
    self,controller_id,payload
):
    ws = self.controllers.get(controller_id)
    if ws:
        await ws.send(payload)
async def broadcast_to_web(
    self,payload
):
    for ws in self.web_clients:
        await ws.send(payload)
```

Setelah seluruh perangkat berhasil diregistrasi, Gateway memasuki tahap distribusi data sesuai dengan tujuan komunikasi. Fungsi send_to_controller() digunakan untuk meneruskan command yang dikirimkan oleh Dashboard menuju RevPi Controller berdasarkan identitas perangkat yang dituju. Sementara itu, fungsi broadcast_to_web() digunakan untuk mendistribusikan data telemetry, indikator, maupun informasi lainnya dari RevPi Controller kepada seluruh Dashboard NI G Web yang sedang aktif. Dengan mekanisme tersebut, RevPi Gateway berperan sebagai pusat distribusi data (*communication hub*) yang mengatur proses pertukaran informasi antarperangkat secara terpusat. Setelah seluruh mekanisme komunikasi berhasil diimplementasikan, tahap berikutnya adalah mengaktifkan layanan WebSocket agar seluruh fungsi tersebut dapat dijalankan selama sistem beroperasi.

Script Program Running dan Stop Gateway

```
async def start():
    self._server = await websockets.serve(
        self.handler,self.host,self.port
    )
async def stop():
    self._server.close()
    await self._server.wait_closed()
```

Tahap terakhir pada file server.py adalah mengaktifkan dan menghentikan layanan WebSocket. Fungsi start() digunakan untuk menjalankan WebSocket Server dengan memanfaatkan konfigurasi host, port, dan fungsi handler() yang telah diimplementasikan sebelumnya sehingga Gateway siap menerima koneksi

dari RevPi Controller maupun Dashboard NI G Web. Sebaliknya, fungsi stop() digunakan untuk menutup layanan WebSocket secara aman (*graceful shutdown*) ketika aplikasi dihentikan. Dengan demikian, seluruh koneksi yang sedang aktif dapat ditutup dengan baik sehingga proses komunikasi berakhir tanpa menimbulkan gangguan pada sistem.

3. messages.py

Setelah layanan komunikasi berhasil diimplementasikan pada server.py, setiap data yang diterima oleh RevPi Gateway perlu diproses sebelum diteruskan ke perangkat tujuan. Proses tersebut diimplementasikan pada file messages.py yang berperan sebagai modul pemrosesan pesan (*message processing module*). Modul ini bertugas mengidentifikasi sumber pesan, menentukan jenis data yang diterima, serta mengarahkan proses pengiriman pesan sesuai dengan mekanisme komunikasi yang telah dirancang. Implementasi utama pada file messages.py dijelaskan melalui beberapa tahapan sebagai berikut.

Script Program Identifikasi dan Pemrosesan Pesan

```
if websocket in server.web_clients:
    ...
elif websocket in server.controllers.values():
    ...
```

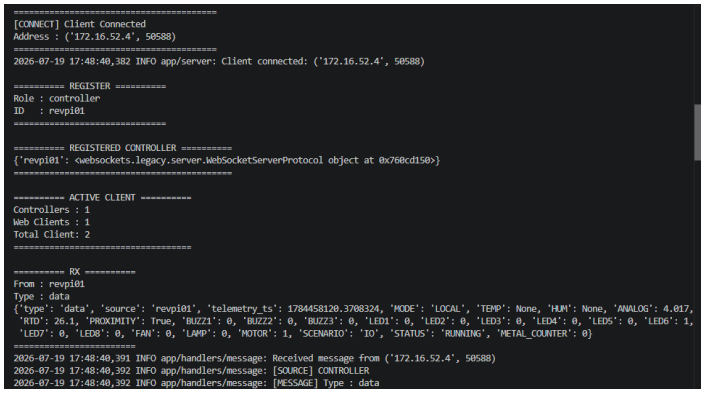
Tahap pertama pada file messages.py adalah mengidentifikasi sumber setiap pesan yang diterima oleh RevPi Gateway. Proses identifikasi dilakukan berdasarkan koneksi WebSocket yang telah diregistrasi sebelumnya melalui mekanisme *handshake*. Pada tahap ini Gateway memeriksa apakah pesan berasal dari Dashboard NI G Web atau dari RevPi Controller. Apabila pesan berasal dari Dashboard, data akan diproses sebagai command yang selanjutnya diteruskan menuju RevPi Controller. Sebaliknya, apabila pesan berasal dari RevPi Controller, Gateway akan mengidentifikasi jenis pesan berdasarkan nilai type yang dikirimkan, seperti telemetry, indicator, maupun acknowledgement, sebelum menentukan proses pengiriman berikutnya. Melalui mekanisme tersebut, setiap pesan dapat diproses sesuai dengan fungsi dan tujuan komunikasinya sehingga pertukaran data antarperangkat berlangsung secara terstruktur dan konsisten.

Script Program Distribusi Pesan

```
await server.broadcast_to_web(...)
await server.send_to_controller(...)
```

Tahap terakhir pada file messages.py adalah mendistribusikan pesan kepada perangkat tujuan sesuai dengan hasil identifikasi dan klasifikasi yang telah dilakukan sebelumnya. Pesan yang berasal dari Dashboard diteruskan menuju RevPi Controller melalui fungsi send_to_controller(), sedangkan data yang berasal dari RevPi Controller didistribusikan kepada Dashboard NI G Web menggunakan fungsi broadcast_to_web(). Dengan mekanisme tersebut, RevPi Gateway tidak hanya berfungsi sebagai penerima pesan, tetapi juga sebagai pengatur alur pertukaran data sehingga komunikasi antarperangkat dapat berlangsung secara terpusat dan sesuai dengan rancangan sistem.

Setelah seluruh modul perangkat lunak berhasil diimplementasikan, tahap berikutnya adalah menjalankan aplikasi RevPi Gateway untuk memastikan seluruh fungsi berjalan sesuai dengan rancangan sistem. Proses ini dilakukan dengan mengeksekusi file main.py pada perangkat Revolution Pi sehingga layanan WebSocket dapat diaktifkan dan mulai menunggu koneksi dari perangkat lain. Ketika aplikasi berhasil dijalankan, terminal akan menampilkan informasi status layanan, alamat host dan port yang digunakan, serta aktivitas koneksi yang dilakukan oleh RevPi Controller maupun Dashboard NI G Web. Tampilan hasil implementasi RevPi Gateway ditunjukkan pada Gambar 4.5.



Gambar 4.5 Running Program Revpi Gateway

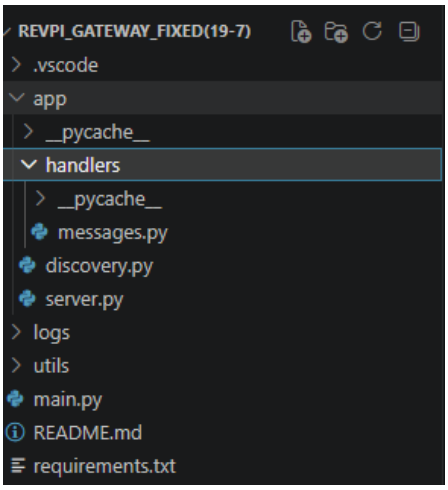
Berdasarkan Gambar 4.5, aplikasi RevPi Gateway berhasil dijalankan dengan baik yang ditandai oleh aktifnya layanan WebSocket pada alamat 0.0.0.0:8765. Selanjutnya, RevPi Controller berhasil membangun koneksi dan melakukan proses registrasi (*handshake*) menggunakan identitas *role* sebagai *controller* dan *id* sebagai *revpi01*. Hasil registrasi tersebut ditunjukkan pada bagian *Registered Controller* dan *Active Client*, yang menunjukkan bahwa RevPi Gateway

telah berhasil menerima koneksi serta mengenali perangkat yang terhubung sehingga siap digunakan sebagai pusat komunikasi dalam sistem Trainer Kit Edge Industrial Internet of Things (IIoT)

B. Revpi sebagai Controller

RevPi Controller merupakan salah satu aplikasi utama pada *Trainer Kit Edge Industrial Internet of Things (IIoT)* yang berfungsi sebagai perangkat *edge*. Aplikasi ini bertugas melakukan akuisisi data sensor, pengendalian aktuator, menjalankan skenario operasi, serta berkomunikasi dengan RevPi Gateway melalui protokol WebSocket. Dengan demikian, RevPi Controller menjadi komponen yang menghubungkan proses monitoring, pengendalian, dan komunikasi pada sistem.

Untuk mendukung fungsi tersebut, implementasi perangkat lunak pada RevPi Controller disusun menggunakan pendekatan modular sehingga setiap fungsi utama dipisahkan ke dalam modul yang memiliki tanggung jawab tertentu. Pendekatan ini mempermudah proses pengembangan, pemeliharaan, serta pengujian sistem. Struktur direktori RevPi Controller ditunjukkan pada Gambar 4.6



Gambar 4.6 Struktur Direktori Revpi Controller

Berdasarkan struktur direktori tersebut, implementasi RevPi Controller terdiri atas beberapa modul yang saling terintegrasi. Namun, pembahasan pada proyek akhir ini difokuskan pada lima berkas utama, yaitu main.py, input_manager.py, dispatcher.py, data_provider.py, dan client_revpi.py. Kelima berkas tersebut merupakan inti aplikasi karena bertanggung jawab terhadap proses inisialisasi sistem, pengelolaan sumber input, pelaksanaan skenario operasi,

akuisisi data sensor, pengendalian perangkat, serta komunikasi dengan RevPi Gateway. Adapun implementasi masing-masing berkas dijelaskan sebagai berikut.

1. main.py

File main.py merupakan titik awal (*entry point*) aplikasi RevPi Controller yang bertugas melakukan inisialisasi sistem serta menjalankan proses utama pada Controller. Modul ini mengintegrasikan komunikasi dengan RevPi Gateway dan pelaksanaan skenario operasi sehingga sistem dapat berjalan secara bersamaan. Implementasi utama pada file main.py dijelaskan sebagai berikut.

Script Program Running Controller

```
await asyncio.gather(
    run_client(...),
    run_dispatcher(mode)
)
```

Tahap pertama pada file main.py adalah menjalankan proses komunikasi dengan RevPi Gateway dan pelaksanaan skenario operasi secara bersamaan menggunakan mekanisme *asynchronous*. Dengan mekanisme tersebut, proses komunikasi dan pengendalian dapat berlangsung secara paralel tanpa saling mengganggu. Setelah proses utama berhasil dijalankan, tahap berikutnya adalah menentukan sumber input yang digunakan oleh sistem.

2. input_manager.py

Setelah proses utama berhasil dijalankan oleh main.py, tahap berikutnya adalah menentukan sumber input yang digunakan oleh RevPi Controller. Fungsi tersebut diimplementasikan pada file input_manager.py yang berperan sebagai modul pengelola sumber masukan. Implementasi utama pada file input_manager.py dijelaskan sebagai berikut.

Script Program Pemilihan Sumber Input

```
def get_input():
    hardware = read_all()
    if not hardware.get("SW10", False):
        return hardware
    return _dashboard_input
```

Tahap pertama pada file input_manager.py adalah menentukan sumber input yang digunakan oleh RevPi Controller. Apabila sistem berada pada mode LOCAL, data diperoleh dari perangkat keras, sedangkan pada mode REMOTE, sistem menggunakan data dari Dashboard NI G Web melalui RevPi Gateway.

Setelah sumber input berhasil ditentukan, tahap berikutnya adalah memilih skenario operasi yang akan dijalankan.

3. dispatcher.py

Setelah sumber input berhasil ditentukan, tahap berikutnya adalah memilih skenario operasi sesuai dengan mode yang dijalankan. Fungsi tersebut diimplementasikan pada file dispatcher.py yang bertugas mengelola pemilihan skenario operasi pada sistem. Implementasi utama pada file dispatcher.py dijelaskan sebagai berikut.

Script Program Pemilihan Skenario

```
SCENARIOS = {
    "io": io_test_case,
    "logic": logic_gate_case,
    "traffic": traffic_light_case,
    "process": mini_process_case,
}
scenario = SCENARIOS.get(mode)
```

Tahap pertama pada file dispatcher.py adalah memilih skenario operasi berdasarkan mode yang ditentukan saat aplikasi dijalankan. Setiap mode akan memanggil modul skenario yang berbeda sesuai dengan fungsi trainer kit. Setelah skenario berhasil dipilih, tahap berikutnya adalah melakukan akuisisi data sensor serta pengendalian perangkat.

4. data_provider.py

Setelah skenario operasi berhasil dipilih, tahap berikutnya adalah melakukan akuisisi data sensor dan pengendalian perangkat yang digunakan selama sistem beroperasi. Fungsi tersebut diimplementasikan pada file data_provider.py yang berperan sebagai penghubung antara aplikasi RevPi Controller dengan perangkat input maupun output. Implementasi utama pada file data_provider.py dijelaskan sebagai berikut.

Script Program Akuisisi Data Sensor

```
def read_all():
    return {
        "ANALOG": mA,
        "RTD": rtd,
        "TEMP": temp,
        "HUM": hum
    }
```

Tahap pertama pada file data_provider.py adalah melakukan pembacaan data dari perangkat input dan sensor yang terhubung ke Revolution Pi. Data yang diperoleh selanjutnya digunakan sebagai dasar dalam menjalankan skenario operasi serta dikirimkan sebagai informasi telemetry. Setelah data berhasil diperoleh, tahap berikutnya adalah menerapkan kondisi output sesuai hasil pengolahan sistem.

Script Program Penerapan Perintah Kontrol

```
def apply_control_command(cmd):
    if "LED1" in cmd:
    if "BUZZ1" in cmd:
    write_outputs()
```

Tahap berikutnya adalah menerapkan kondisi perangkat output berdasarkan hasil pemrosesan sistem. Potongan kode tersebut digunakan untuk mengendalikan aktuator seperti LED dan buzzer sehingga kondisi perangkat fisik sesuai dengan hasil skenario operasi maupun perintah yang diterima. Setelah kondisi output berhasil diterapkan, tahap berikutnya adalah membangun komunikasi dengan RevPi Gateway untuk mengirimkan data telemetry dan menerima perintah kontrol.

5. client_revpi.py

Setelah proses akuisisi data dan pengendalian perangkat berhasil dilakukan, tahap berikutnya adalah membangun komunikasi antara RevPi Controller dan RevPi Gateway. Fungsi tersebut diimplementasikan pada file client_revpi.py yang bertugas mengelola pertukaran data melalui protokol WebSocket. Implementasi utama pada file client_revpi.py dijelaskan sebagai berikut.

Script Program Handshake ke RevPi Gateway

```
await websocket.send(handshake)
```

Tahap pertama pada file client_revpi.py adalah mengirimkan pesan *handshake* ke RevPi Gateway. Proses ini bertujuan untuk melakukan registrasi identitas RevPi Controller sehingga Gateway dapat mengenali perangkat yang terhubung. Setelah proses registrasi berhasil dilakukan, tahap berikutnya adalah membentuk dan mengirimkan data telemetry.

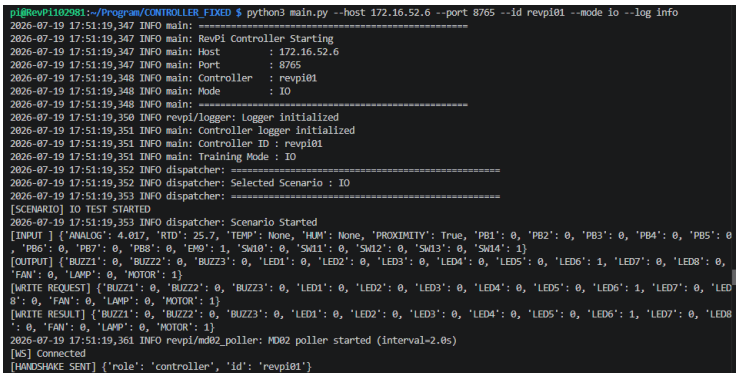
Script Program Pengiriman Telemetry

```
payload = build_payload(...)
await websocket.send(json.dumps(payload))
```

Tahap berikutnya adalah membentuk dan mengirimkan data telemetry ke RevPi Gateway secara berkala. Data yang dikirimkan meliputi informasi sensor,

status perangkat, serta kondisi sistem yang digunakan untuk proses monitoring pada Dashboard NI G Web. Dengan mekanisme tersebut, kondisi RevPi Controller dapat dipantau secara *real-time* oleh pengguna.

Setelah seluruh modul pada RevPi Controller berhasil diimplementasikan, tahap berikutnya adalah menjalankan aplikasi untuk memastikan setiap fungsi dapat bekerja sesuai dengan rancangan. Proses ini bertujuan untuk memverifikasi bahwa Controller mampu melakukan komunikasi dengan RevPi Gateway, menjalankan skenario operasi, serta mengirimkan data telemetry selama sistem beroperasi. Hasil implementasi RevPi Controller ditunjukkan pada Gambar 4.7.



Gambar 4.7 Running Program Revpi Controller

Berdasarkan Gambar 4.7, aplikasi RevPi Controller berhasil dijalankan dengan baik yang ditandai dengan berhasilnya proses inisialisasi sistem, pemilihan skenario operasi IO, serta pembacaan data input dan output pada perangkat Revolution Pi. Selain itu, layanan pembacaan sensor berhasil diaktifkan dan proses *handshake* berhasil dikirimkan ke RevPi Gateway sebagai langkah awal pembentukan komunikasi WebSocket. Hasil tersebut menunjukkan bahwa RevPi Controller telah berfungsi sesuai dengan rancangan dan siap digunakan sebagai perangkat *edge* pada sistem Trainer Kit Edge Industrial Internet of Things (IIoT).

C. Logging System (CSV & Database)

logging digunakan untuk mencatat aktivitas sistem selama RevPi Gateway dan RevPi Controller beroperasi. Data yang dicatat meliputi informasi proses aplikasi, komunikasi WebSocket, pembacaan sensor, serta kondisi perangkat sehingga dapat digunakan untuk proses pemantauan, analisis, dan pengujian sistem. Implementasi logging dijelaskan sebagai berikut.

Script Program Logging

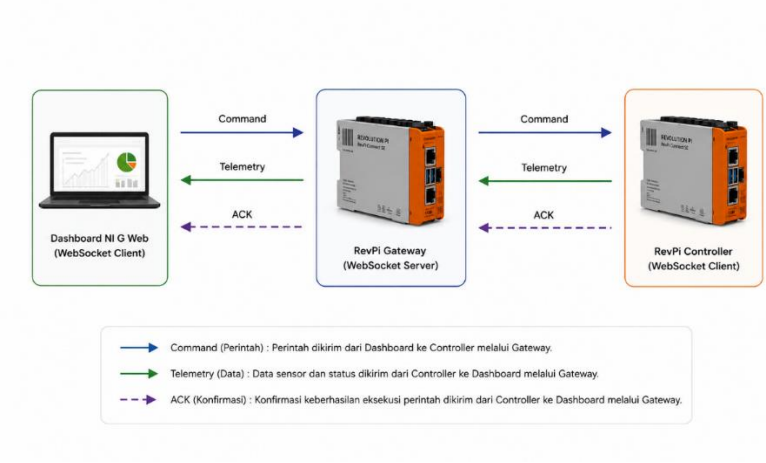
```
def measurements(data):
    row = {
        "ANALOG": data.get("ANALOG"),
        "RTD": data.get("RTD"),
        "TEMP": data.get("TEMP"),
        "HUM": data.get("HUM")
    }
    _insert_db(...)
    _insert_csv(...)
```

Tahap pertama pada file logging.py adalah mencatat data yang dihasilkan selama sistem beroperasi. Potongan kode tersebut digunakan untuk menyimpan data *telemetry*, perubahan status aktuator (*event*), serta konfirmasi eksekusi perintah (*Acknowledgement/ACK*) ke dalam SQLite Database dan file CSV. Setelah proses pencatatan berhasil dilakukan, data *logging* dapat digunakan sebagai dokumentasi dan bahan analisis sistem.

Berdasarkan hasil implementasi yang telah dilakukan, seluruh modul perangkat lunak pada RevPi Gateway, RevPi Controller, dan mekanisme *logging* telah berhasil diintegrasikan sesuai dengan fungsinya masing-masing. Selanjutnya, pembahasan difokuskan pada implementasi komunikasi WebSocket yang digunakan untuk mendukung pertukaran data antara RevPi Gateway, RevPi Controller, dan Dashboard NI G Web secara *real-time*.

4.1.3 Implementasi Komunikasi WebSocket

Implementasi komunikasi WebSocket pada proyek akhir ini dirancang untuk menghubungkan dua perangkat Revolution Pi yang memiliki peran berbeda, yaitu RevPi Gateway sebagai pusat komunikasi dan RevPi Controller sebagai pengendali trainer kit. Selain itu, Dashboard NI G Web juga terhubung ke RevPi Gateway sebagai antarmuka monitoring dan pengendalian sistem. Pada implementasinya, seluruh proses pertukaran data tidak dilakukan secara langsung antara Dashboard dan Controller, melainkan melalui RevPi Gateway yang bertugas menerima koneksi, mengelola identitas perangkat, serta meneruskan setiap pesan sesuai dengan tujuan komunikasi. Mekanisme komunikasi yang telah diimplementasikan ditunjukkan pada Gambar 4.8.



Gambar 4.8 Diagram Sistem Komunikasi

Berdasarkan Gambar 4.8, RevPi Controller terlebih dahulu membangun koneksi WebSocket dengan RevPi Gateway untuk mengirimkan data *telemetry* yang diperoleh dari pembacaan sensor dan status aktuator. Selanjutnya, Dashboard NI G Web membangun koneksi ke RevPi Gateway untuk menerima data *telemetry* sekaligus mengirimkan *command* pengendalian. RevPi Gateway kemudian meneruskan *command* tersebut kepada RevPi Controller, sedangkan Controller mengirimkan pesan *acknowledgement* (ACK) sebagai konfirmasi bahwa perintah telah berhasil diproses. Dengan mekanisme tersebut, seluruh komunikasi pada trainer kit dapat berlangsung secara terpusat dan *real-time*. Implementasi komunikasi tersebut dijelaskan melalui proses monitoring dan pengendalian sistem sebagai berikut.

A. Implementasi Monitoring Sistem

Implementasi monitoring dilakukan untuk mengirimkan data sensor dari RevPi Controller ke Dashboard NI G Web melalui RevPi Gateway menggunakan komunikasi WebSocket. Data yang dikirim berupa *telemetry* hasil pembacaan sensor dan status perangkat secara *real-time*.

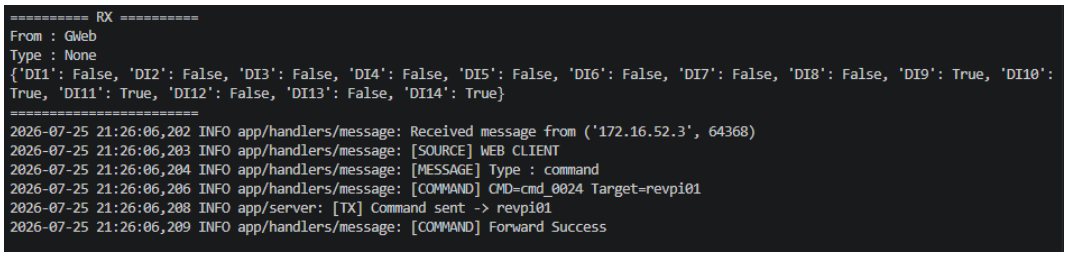
```
========== RX ==========
From : revpi01
Type : data
{'type': 'data', 'source': 'revpi01', 'telemetry_ts': 1784989444.6696286, 'MODE': 'LOCAL', 'TEMP': 26.6, 'HUM': 42.1, 'ANALOG': 16.515
, 'RTD': 25.6, 'PROXIMITY': False, 'BUZZ1': False, 'BUZZ2': False, 'BUZZ3': False, 'LED1': False, 'LED2': False, 'LED3': False, 'LED4'
: False, 'LED5': False, 'LED6': True, 'LED7': False, 'LED8': True, 'FAN': False, 'LAMP': False, 'MOTOR': False, 'SCENARIO': 'IO', 'STA
TUS': 'RUNNING', 'METAL_COUNTER': 0}
========================
2026-07-25 21:24:04,665 INFO app/handlers/message: Received message from ('172.16.52.4', 56396)
2026-07-25 21:24:04,667 INFO app/handlers/message: [SOURCE] CONTROLLER
2026-07-25 21:24:04,667 INFO app/handlers/message: [MESSAGE] Type : data
2026-07-25 21:24:04,668 INFO app/handlers/message: [DATA] revpi01
```

Gambar 4.9 Implementasi Monitoring Sistem

Berdasarkan Gambar 4.9, RevPi Controller berhasil mengirimkan data *telemetry* ke RevPi Gateway melalui koneksi WebSocket. Selanjutnya, data diteruskan ke Dashboard NI G Web sehingga informasi sensor dan status perangkat dapat ditampilkan secara *real-time*. Hasil tersebut menunjukkan bahwa proses monitoring telah berjalan sesuai dengan rancangan sistem.

B. Implementasi Kontrol Sistem

Implementasi kontrol dilakukan untuk mengirimkan *command* dari Dashboard NI G Web menuju RevPi Controller melalui RevPi Gateway. Setelah *command* diterima, RevPi Controller mengeksekusi perintah dan mengirimkan pesan *acknowledgement* (ACK) sebagai konfirmasi.



Gambar 4.10 Implementasi Kontrol Sistem

Berdasarkan Gambar 4.10, *command* yang dikirimkan dari Dashboard berhasil diteruskan oleh RevPi Gateway kepada RevPi Controller. Selanjutnya, Controller mengeksekusi perintah dan mengirimkan pesan *ACK* sebagai konfirmasi keberhasilan eksekusi. Hasil implementasi menunjukkan bahwa proses pengendalian telah berjalan secara *real-time* sesuai dengan rancangan sistem.

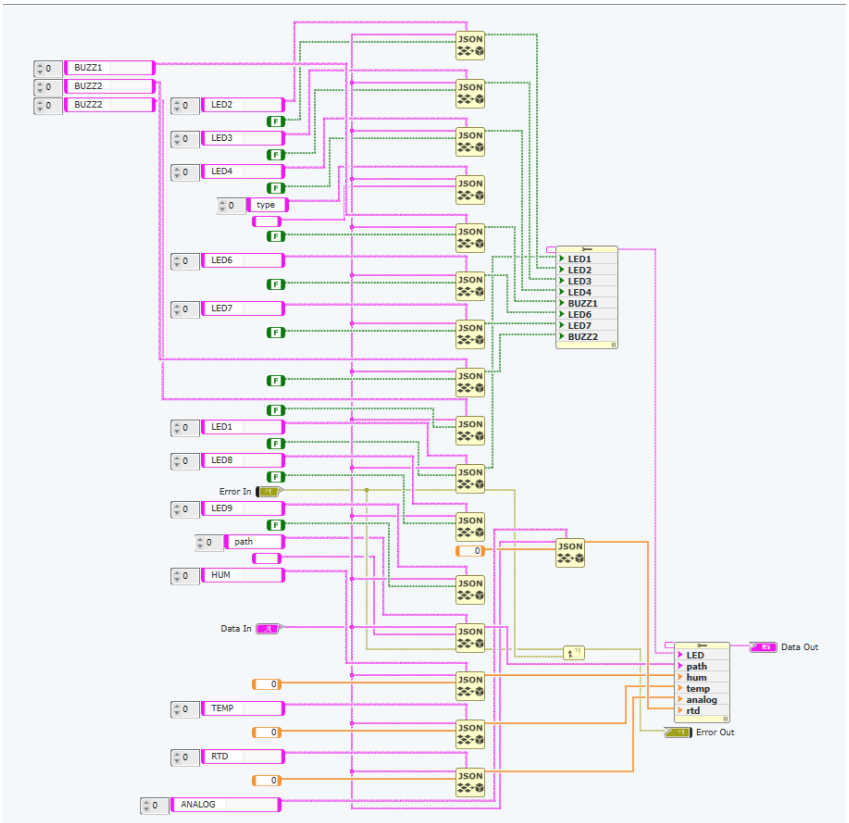
Berdasarkan hasil implementasi, komunikasi WebSocket telah berhasil menghubungkan RevPi Controller, RevPi Gateway, dan Dashboard NI G Web. Proses monitoring dan pengendalian dapat berlangsung secara *real-time* sehingga sistem beroperasi sesuai dengan rancangan yang diusulkan.

4.1.4 Implementasi Dashboard NI G Web Development Software

Dashboard monitoring dan kontrol diimplementasikan menggunakan NI G Web Development Software sebagai antarmuka pengguna berbasis web. Dashboard berfungsi menampilkan data *telemetry*, status perangkat, serta mengirimkan perintah kontrol ke RevPi Controller melalui RevPi Gateway menggunakan komunikasi WebSocket. Implementasi dashboard terdiri atas dua blok diagram utama, yaitu Parse JSON dan Main Program.

A. Blok Diagram Parse Json

Block diagram Parse Json digunakan untuk memproses data telemetry yang diterima dari RevPi Gateway dalam format JSON sebelum ditampilkan pada dashboard.

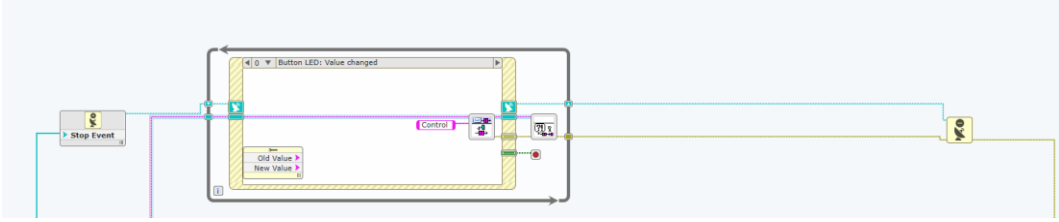


Gambar 4.11 Blok Diagram Parsing Json

Berdasarkan Gambar 4.11**,** data *telemetry* yang diterima diuraikan menjadi beberapa parameter, meliputi temperatur (TEMP), kelembapan (HUM), RTD, analog input, serta status LED,buzzer, fan.lamp dan motor. Setiap parameter kemudian dipetakan ke indikator yang sesuai sehingga informasi monitoring dapat ditampilkan secara *real-time* pada dashboard.

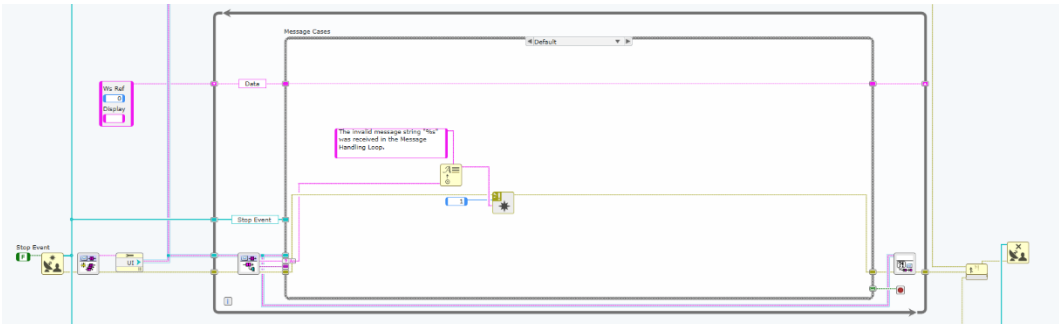
B. Blok Diagram Main Program

Blok diagram Main Program diimplementasikan sebagai pusat pengendali seluruh fungsi pada Dashboard NI G Web. Diagram ini mengintegrasikan proses komunikasi, monitoring, dan kontrol yang dijalankan pada dashboard sehingga pertukaran data antara pengguna dan sistem dapat berlangsung secara terkoordinasi.



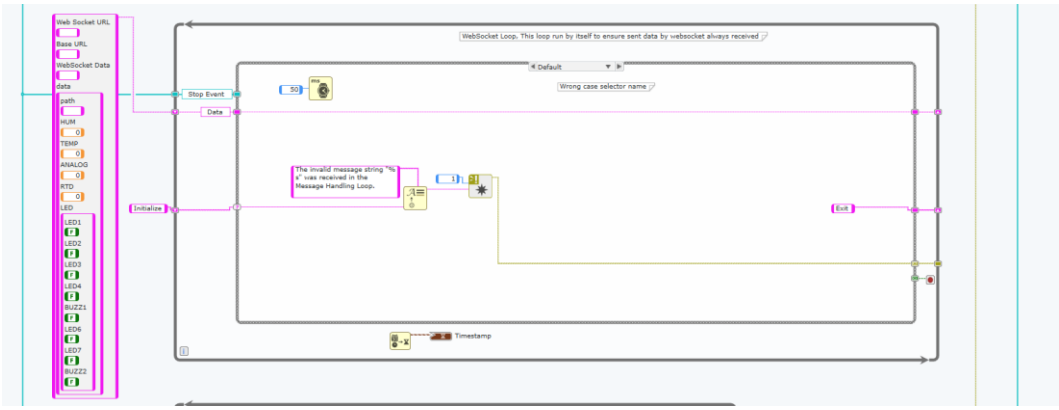
Gambar 4.12 Loop Event Handler

Gambar 4.12 menunjukkan **Loop Event Handler** yang berfungsi menangani setiap interaksi pengguna pada dashboard. Event yang berasal dari tombol maupun perubahan nilai kontrol dideteksi, kemudian diteruskan ke proses yang sesuai sehingga setiap aksi pengguna dapat diproses secara *real-time*.



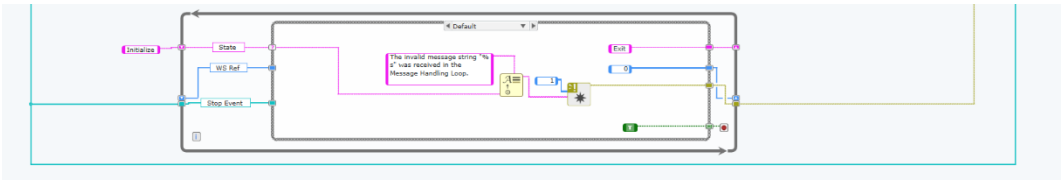
Gambar 4.13 Loop Control

Gambar 4.13 menunjukkan **Loop Control** yang bertanggung jawab mengirimkan perintah kontrol ke RevPi Gateway melalui komunikasi WebSocket. Perintah yang berasal dari dashboard dikemas dalam format JSON sebelum diteruskan ke RevPi Controller untuk mengendalikan aktuator.



Gambar 4.14 Loop Monitoring

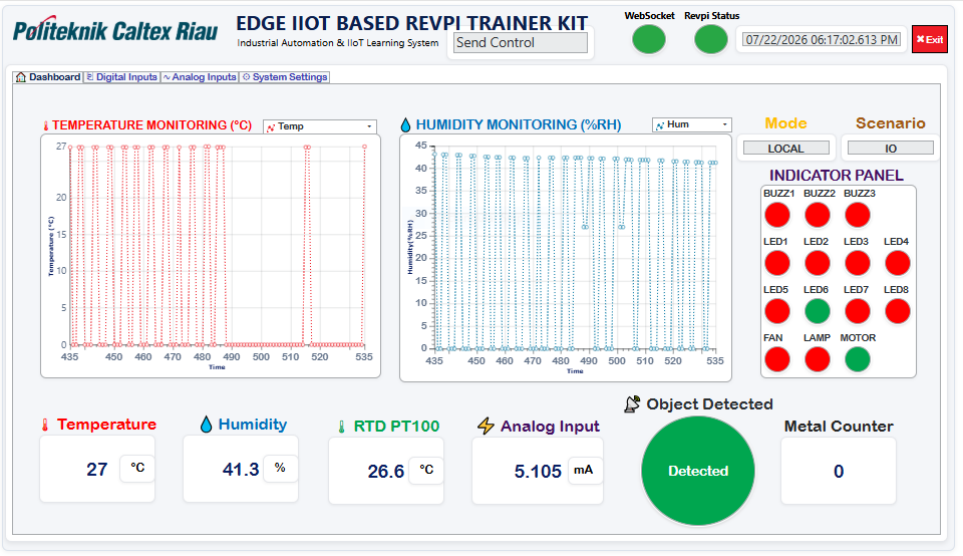
Gambar 4.14 menunjukkan **Loop Monitoring** yang berfungsi memperbarui tampilan dashboard berdasarkan data telemetry yang diterima dari RevPi Gateway. Melalui mekanisme ini, nilai sensor dan status aktuator dapat ditampilkan secara *real-time*.



Gambar 4.15 Loop WebSocket

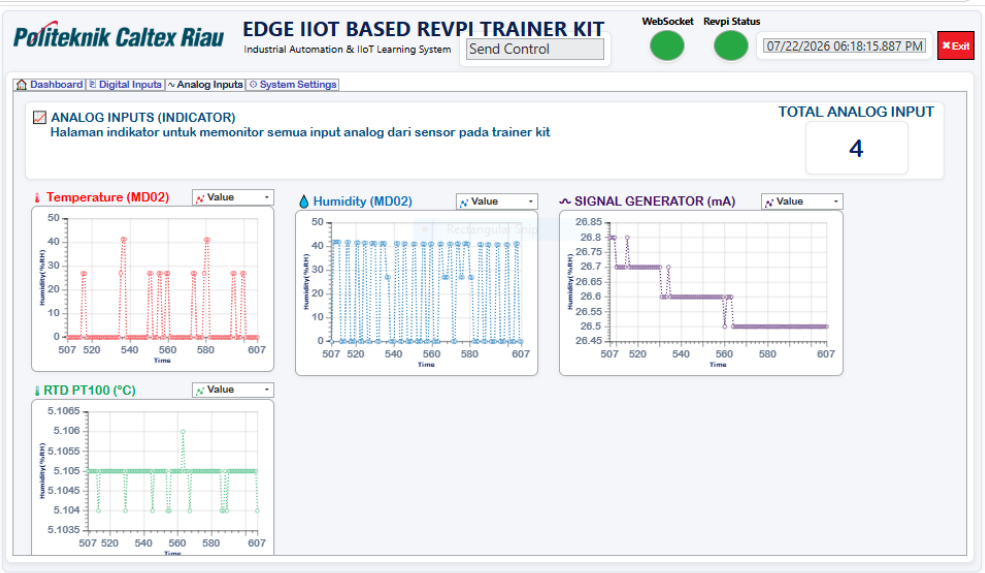
Gambar 4.15 menunjukkan **Loop WebSocket** yang mengelola proses komunikasi antara Dashboard NI G Web dan RevPi Gateway. Loop ini bertugas menerima data telemetry serta mengirimkan perintah kontrol melalui koneksi WebSocket sehingga komunikasi dua arah dapat berlangsung secara berkesinambungan.

Setelah seluruh blok diagram berhasil diimplementasikan, Dashboard NI G Web dijalankan sebagai antarmuka monitoring dan kontrol pada Trainer Kit Edge Industrial Internet of Things (IIoT). Dashboard menerima data *telemetry* dari RevPi Controller melalui RevPi Gateway serta mengirimkan *command* pengendalian secara *real-time*. Hasil implementasi dashboard ditunjukkan pada Gambar 4.16 dan Gambar 4.17.



Gambar 4.16 Dashboard Monitoring & Kontrol NI G Web

Berdasarkan Gambar 4.16, halaman utama Dashboard NI G Web digunakan sebagai pusat monitoring dan pengendalian sistem secara *real-time*. Halaman ini menampilkan grafik *Temperature Monitoring* dan *Humidity* Monitoring, indikator nilai Temperature, Humidity, RTD PT100, dan Analog Input, serta panel Indicator Panel yang menampilkan status LED, buzzer, fan, lamp, dan motor. Selain itu, dashboard juga menampilkan informasi Mode, Scenario, status koneksi WebSocket, status RevPi, indikator Object Detected, serta Metal Counter sebagai informasi kondisi sistem selama proses monitoring berlangsung.



Gambar 4.17 Dashboard Monitoring Analog Input NI G Web

Berdasarkan Gambar 4.17, halaman Analog Inputs digunakan untuk memantau seluruh parameter analog pada Trainer Kit Edge IIoT. Dashboard menampilkan grafik hasil pembacaan Temperature (MD02), Humidity (MD02), RTD PT100, dan Signal Generator (mA) secara terpisah sehingga setiap parameter dapat diamati dengan lebih mudah.

Berdasarkan hasil implementasi, Dashboard NI G Web telah berhasil berfungsi sebagai antarmuka monitoring dan kontrol pada *Trainer Kit Edge Industrial Internet of Things* (IIoT). Dashboard mampu menampilkan data *telemetry* serta mengirimkan perintah kontrol melalui RevPi Gateway secara *real-time*, sehingga integrasi antara Dashboard NI G Web, RevPi Gateway, dan RevPi Controller berjalan sesuai dengan rancangan sistem.

4.2 Pengujian Sistem *Trainer Kit Edge IIoT* dan Analisis Data Pengujian

Pengujian sistem dilakukan untuk memvalidasi kinerja *framework* perangkat lunak yang telah dikembangkan pada *Trainer Kit Edge Industrial Internet of Things (IIoT). Framework* tersebut merupakan *base application* yang menyediakan fungsi komunikasi WebSocket, akuisisi data sensor, pengendalian aktuator, pengelolaan mode operasi, serta mekanisme *logging* yang digunakan secara bersama oleh seluruh aplikasi pada *trainer kit*. Dengan demikian, pengujian difokuskan pada kemampuan *framework* dalam mempertahankan performa sistem ketika diimplementasikan pada berbagai karakteristik proses industri

Sebagai implementasi *framework* tersebut, proyek akhir ini berhasil mengembangkan dua skenario aplikasi yang mewakili karakteristik proses otomasi industri yang berbeda, yaitu *Metal Part Conveyor Monitoring System* sebagai representasi proses manufaktur diskrit (*discrete manufacturing*) dan *Automatic Mixing Tank Process Control System* sebagai representasi proses kontinu (*continuous process*). Kedua skenario dibangun menggunakan arsitektur perangkat lunak yang sama sehingga perbedaannya hanya terletak pada logika proses, jumlah variabel proses, serta kompleksitas pengendalian yang diterapkan.

Berdasarkan karakteristik tersebut, seluruh parameter pengujian, meliputi akuisisi data sensor, kontrol aktuator, serta komunikasi WebSocket, dievaluasi pada kedua skenario untuk mengetahui kemampuan framework dalam mempertahankan performa sistem pada tingkat kompleksitas aplikasi yang berbeda. Dengan demikian, hasil pengujian tidak hanya menunjukkan performa komunikasi sistem, tetapi juga memvalidasi bahwa *framework* mampu diimplementasikan pada berbagai karakteristik proses industri.

4.2.1 Pengujian Akuisisi Data Sensor

Pengujian akuisisi data sensor dilakukan untuk memverifikasi kemampuan framework dalam membaca dan mengirimkan data sensor dari RevPi Controller menuju Dashboard NI G Web melalui RevPi Gateway. Pengujian ini diterapkan pada kedua skenario aplikasi sehingga mekanisme akuisisi data dapat dievaluasi pada kondisi proses diskrit maupun proses kontinu. Parameter yang diuji meliputi sensor temperatur, kelembapan, RTD PT100, analog input, dan proximity. Hasil pengujian selanjutnya disajikan pada Tabel 4.1 berikut.

Tabel 4.1 Hasil Pengujian Akuisisi Data Sensor

Mode	Skenario	Waktu	Temp (C/G)	HUM (C/G)	RTD (C/G)	Analog (C/G)	Proximity (C/G)	Status
Local	Valid I/O	12/08/2026 14:50:12,906	–/–	–/–	26,2/26,2	4,018/4,018	False/False	Sesuai
Local	Valid I/O	12/08/2026 14:50:15,966	27,0/27,0	34,7/34,7	26,3/26,3	4,018/4,018	False/False	Sesuai
Local	Valid I/O	12/08/2026 14:50:17,960	27,0/27,0	34,7/34,7	26,3/26,3	4,017/4,017	False/False	Sesuai
Remote	Metal Part	12/08/2026 14:46:11,472	27,4/27,4	35,2/35,2	26,2/26,2	4,016/4,016	False/False	Sesuai
Remote	Metal Part	12/08/2026 14:46:13,530	27,4/27,4	35,2/35,2	26,3/26,3	4,017/4,017	False/False	Sesuai
Remote	Mixing Tank	12/08/2026 15:24:00,159	27,0/27,0	37,9/37,9	26,4/26,4	4,018/4,018	False/False	Sesuai
Remote	Mixing Tank	12/08/2026 15:24:02,173	27,0/27,0	37,9/37,9	26,4/26,4	4,018/4,018	False/False	Sesuai

Berdasarkan Tabel 4.1 seluruh parameter sensor berhasil dibaca oleh RevPi Controller dan diteruskan ke RevPi Gateway hingga ditampilkan pada Dashboard NI G Web dengan nilai yang sama pada ketiga skenario aplikasi. Hasil tersebut menunjukkan bahwa framework mampu mempertahankan integritas data sensor secara konsisten meskipun jumlah variabel dan kompleksitas proses berbeda pada setiap skenario.

Hasil pengujian juga menunjukkan bahwa proses akuisisi data dan pengiriman *telemetry* melalui komunikasi WebSocket dapat berjalan secara konsisten tanpa kehilangan data. Dengan demikian, sistem mampu melakukan monitoring kondisi sensor secara *real-time* sesuai dengan rancangan yang telah diimplementasikan.

4.2.2 Pengujian Kontrol Aktuator

Pengujian kontrol aktuator dilakukan untuk mengevaluasi kemampuan framework dalam mengeksekusi perintah pengendalian yang dikirimkan melalui Dashboard NI G Web. Pengujian diterapkan pada kedua skenario aplikasi sehingga mekanisme pengendalian dapat divalidasi pada berbagai karakteristik proses industri. Aktuator yang diuji meliputi LED, buzzer, motor DC, relay SSR, kipas (*fan*), dan perangkat keluaran lainnya sesuai kebutuhan masing-masing skenario. Hasil pengujian selanjutnya disajikan pada tabel berikut.

Tabel 4.2 Hasil Pengujian Kontrol Aktuator

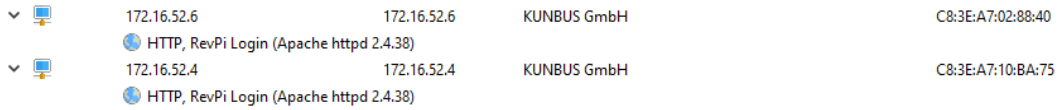
Mode	Skenario	Command ID	Waktu	Status	Berhasil
Local	Valid I/O	cmd_0001	12/08/2026 19:59:29.560	LOCAL_MODE	0
Local	Valid I/O	cmd_0002	12/08/2026 19:59:30.379	LOCAL_MODE	0
Local	Valid I/O	cmd_0003	12/08/2026 19:59:30.736	LOCAL_MODE	0
Remote	Metal Part	cmd_0001	12/08/2026 15:14:18.512	OK	1
Remote	Metal Part	cmd_0002	12/08/2026 15:14:18.621	OK	1
Remote	Metal Part	cmd_0003	12/08/2026 15:14:38.005	OK	1
Remote	Mixing Tank	cmd_0001	12/08/2026 18:52:48.428	OK	1
Remote	Mixing Tank	cmd_0002	12/08/2026 18:52:48.487	OK	1
Remote	Mixing Tank	cmd_0003	12/08/2026 18:53:54.203	OK	1

Berdasarkan Tabel 4.2 seluruh aktuator memberikan respons yang sesuai terhadap perintah yang dikirimkan. Kesesuaian status antara Dashboard NI G Web, RevPi Gateway, dan RevPi Controller menunjukkan bahwa *framework* mampu mempertahankan sinkronisasi proses pengendalian pada ketiga skenario aplikasi secara konsisten.

Selain itu, status aktuator pada RevPi Gateway, RevPi Controller, dan Dashboard NI G Web menunjukkan kondisi yang sama pada setiap pengujian. Hasil tersebut menunjukkan bahwa mekanisme pengendalian dan sinkronisasi status perangkat melalui komunikasi WebSocket telah berjalan dengan baik sehingga proses kontrol dapat dilakukan secara *real-time* sesuai dengan rancangan sistem.

4.2.3 Pengujian Komunikasi WebSocket

Pengujian komunikasi WebSocket dilakukan untuk mengevaluasi performa *framework* komunikasi yang dikembangkan pada *Trainer Kit Edge Industrial Internet of Things (IIoT). Framework* tersebut diimplementasikan pada dua skenario aplikasi. Kedua skenario menggunakan arsitektur komunikasi yang sama, namun memiliki kompleksitas logika proses, jumlah variabel, dan ukuran *payload* yang berbeda. Oleh karena itu, pengujian dilakukan untuk mengetahui kemampuan *framework* dalam mempertahankan performa komunikasi secara *real-time* pada berbagai karakteristik aplikasi industri. Sebelum pengujian dilakukan, konektivitas jaringan terlebih dahulu divalidasi menggunakan aplikasi **Advanced IP Scanner** untuk memastikan RevPi Controller, RevPi Gateway, dan komputer yang menjalankan Dashboard NI G Web berada pada jaringan lokal (*Local Area Network*) yang sama.



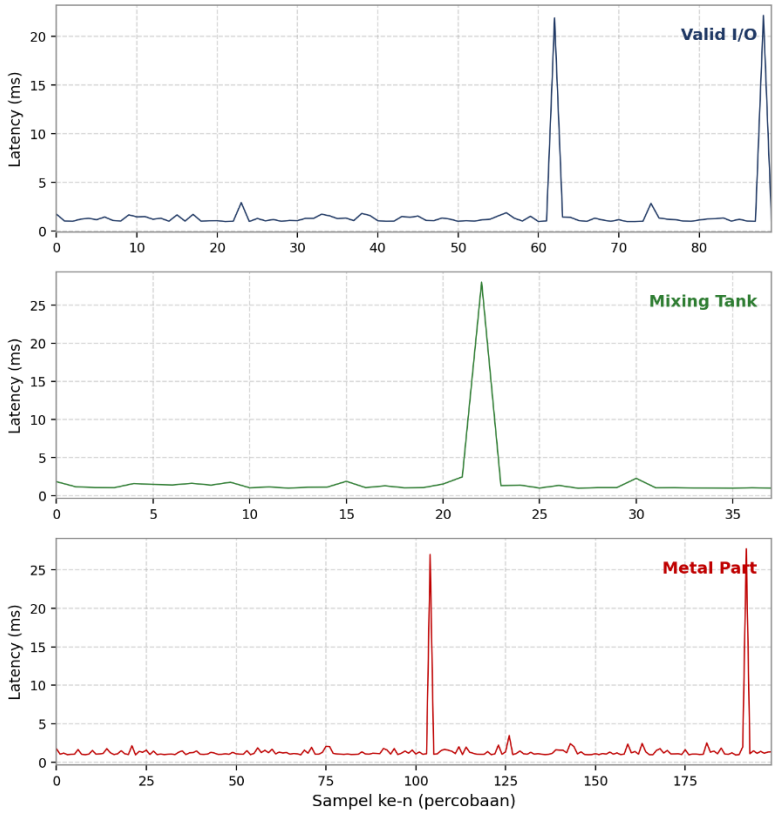
Gambar 4.18 Hasil Pemindaian Perangkat Menggunakan Advanced IP Scanner

Hasil pemindaian menggunakan Advanced IP Scanner menunjukkan bahwa RevPi Controller dengan alamat IP 172.16.52.4 dan RevPi Gateway dengan alamat IP 172.16.52.6 berhasil teridentifikasi pada subnet yang sama, yaitu jaringan 172.16.52.0/24, sehingga komunikasi WebSocket antarperangkat dapat berlangsung dengan baik. Selama proses pengujian, sistem menggunakan jaringan *Local Area Network (LAN)* kampus yang terintegrasi dengan jaringan internet dan digunakan secara bersama (*shared network*) oleh perangkat serta layanan lainnya. Dengan demikian, hasil pengujian merepresentasikan performa komunikasi sistem pada kondisi jaringan operasional yang sesungguhnya, yang berpotensi dipengaruhi oleh lalu lintas komunikasi dari pengguna lain. Setelah konektivitas jaringan tervalidasi, evaluasi komunikasi dilakukan terhadap beberapa parameter, meliputi latency eksekusi kontrol, *round trip time* (RTT), *control success rate, telemetry success rate*, throughput uplink, serta stabilitas interval pengiriman data (Δt). Seluruh parameter dianalisis berdasarkan hasil pengujian pada kedua skenario

sehingga dapat diketahui pengaruh kompleksitas proses terhadap performa komunikasi.

A. Pengujian Latency Eksekusi Kontrol

Pengujian *latency* eksekusi kontrol dilakukan untuk mengevaluasi waktu respons framework dalam mengeksekusi perintah kontrol yang dikirimkan dari Dashboard NI G Web menuju RevPi Controller melalui RevPi Gateway. Pengujian diterapkan pada dua skenario. Perbedaan karakteristik proses pada masing-masing skenario digunakan untuk mengevaluasi pengaruh kompleksitas aplikasi terhadap waktu respons sistem. Nilai *latency* dihitung menggunakan selisih *timestamp* antara saat perintah dikirim dan saat status aktuator berhasil diperbarui. Data hasil pengujian terlampir pada Gambar 4.19 dan Tabel 4.3**.**



Gambar 4.19 Hasil Pengujian *Latency Control* 3 Skenario

Tabel 4.3 Hasil Perbandingan *Latency Control* 3 Skenario

Skenario	N	Rata-rata (ms)	Std Deviasi (ms)	Min (ms)	Max (ms)
Valid I/O	90	1.708333	3.101226	0.96	22.17
Mixing Tank	38	1.977105	4.354073	0.96	28.02
Metal Part	200	1.5208	2.63002	0.94	27.75

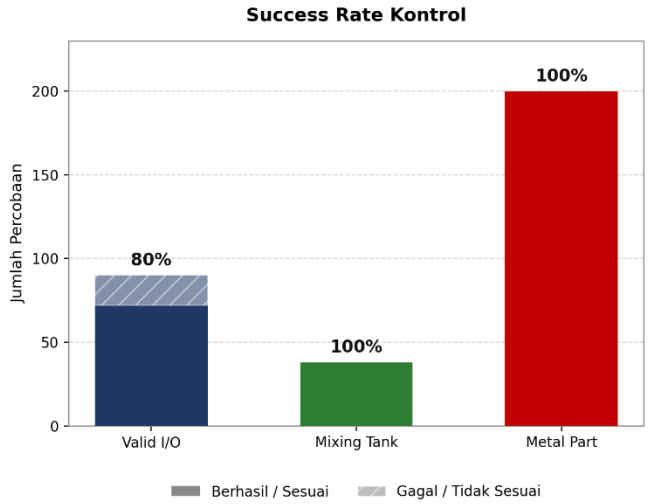
Berdasarkan hasil pengujian pada Tabel 4.3 diperoleh rata-rata *latency* eksekusi kontrol sebesar 1,71 ms pada skenario *Valid I/O,* 1,58 ms pada *Metal Part,* dan 1,98 ms pada *Mixing Tank*. Hasil tersebut menunjukkan bahwa sistem mampu mengeksekusi perintah kontrol dalam waktu yang sangat singkat pada seluruh skenario pengujian. Nilai *latency* yang relatif rendah mengindikasikan bahwa komunikasi antara Dashboard NI G Web, RevPi Gateway, dan RevPi Controller dapat berlangsung secara responsif. Selain itu, nilai rata-rata yang berada di bawah 2 ms menunjukkan bahwa proses penerimaan, pemrosesan, dan eksekusi perintah kontrol dapat dilakukan hampir secara langsung setelah perintah dikirimkan.

Variasi waktu eksekusi masih ditemukan selama pengujian, yang ditunjukkan oleh nilai standar deviasi sebesar 3,10 ms pada Valid I/O, 2,63 ms pada Metal Part, dan 4,35 ms pada Mixing Tank. Meskipun terdapat beberapa nilai maksimum yang mencapai 22,17 ms, 27,75 ms, dan 28,02 ms, sebagian besar data tetap berada pada rentang waktu yang rendah dan konsisten.

Berdasarkan hasil tersebut dapat disimpulkan bahwa *framework* yang dikembangkan mampu mempertahankan waktu respons kontrol yang cepat dan stabil. Hal ini menunjukkan bahwa komunikasi WebSocket yang diterapkan mampu mendukung kebutuhan pengendalian sistem secara *real-time* pada berbagai skenario aplikasi industri.

B. Pengujian *Success Rate Control*

Pengujian *Control Success Rate* dilakukan untuk mengevaluasi tingkat keberhasilan framework dalam mengeksekusi perintah kontrol yang dikirimkan dari Dashboard NI G Web menuju RevPi Controller. Parameter ini digunakan untuk memastikan bahwa setiap perintah yang dikirim dapat diterima, diproses, dan dieksekusi dengan benar pada ketiga skenario aplikasi. Hasil pengujian ditunjukkan pada Gambar 4.20 dan Tabel 4.4**.**



Gambar 4.20 Hasil Pengujian Succes Rate Kontrol

Tabel 4.4 Hasil Perbandingan *Succes Rate Control* 3 Skenario

Skenario	Jumlah Percobaan	Berhasil	Gagal	Success Rate (%)
Valid I/O	90	72	18	80
Mixing Tank	38	38	0	100
Metal Part	200	200	0	100

Nilai *success rate* kontrol dapat dihitung menggunakan Persamaan (4.2).

$$\boldsymbol{SR_{control}}\,(\%) = \frac{\boldsymbol{N_{executed}}}{\boldsymbol{N_{received}}} \times \mathbf{100\%} \qquad 4.2$$

Persamaan (4.2) dapat digunakan untuk menghitung tingkat keberhasilan pengiriman perintah kontrol. Sebagai contoh untuk skenario *valid io*, apabila jumlah perintah yang dikirimkan sebanyak 90 perintah dan berhasil diterima serta dieksekusi oleh sistem sebanyak 72 perintah dan gagal diterima sistem sebanyak 18 perintah, maka nilai *success rate* dapat dihitung sebagai berikut.

$$\boldsymbol{SR_{control}}\,(\%) = \frac{\mathbf{72}}{\mathbf{90}} \times \mathbf{100\%}$$
$$= \mathbf{80\%}$$

Berdasarkan hasil pengujian pada Gambar 4.20 dan Tabel 4.4 diperoleh tingkat keberhasilan eksekusi kontrol yang berbeda pada masing-masing skenario aplikasi. Skenario *Valid I/O* menghasilkan *success rate* sebesar 80%, dengan 72 perintah berhasil dieksekusi dari total 90 percobaan. Sementara itu, skenario *Mixing Tank* dan *Metal Part* masing-masing memperoleh *success rate* sebesar 100%, dimana seluruh perintah yang dikirimkan berhasil diterima dan dieksekusi oleh sistem tanpa kegagalan. Hasil tersebut menunjukkan bahwa mekanisme

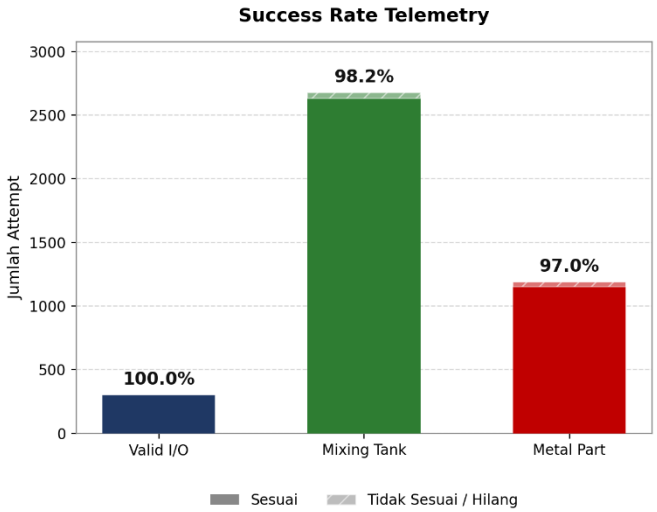
komunikasi WebSocket yang diterapkan mampu mendukung proses pengiriman dan eksekusi perintah kontrol secara andal. Nilai *success rate* sebesar 100% pada skenario Mixing Tank dan Metal Part mengindikasikan bahwa komunikasi antara Dashboard NI G Web, RevPi Gateway, dan RevPi Controller dapat berlangsung secara konsisten selama pengujian.

Pada skenario *Valid I/O* terdapat 18 data tidak sesuai dari 90 percobaan yang dilakukan. Seluruh ketidaksesuaian tersebut terjadi pada *Local Mode,* sehingga tidak mengindikasikan adanya kegagalan pada komunikasi WebSocket. Kondisi tersebut lebih dipengaruhi oleh proses pembacaan input fisik dan sinkronisasi data pada sistem, karena status input diperoleh secara langsung dari perangkat keras yang terhubung ke RevPi Controller.

Secara keseluruhan, dari 328 percobaan yang dilakukan pada ketiga skenario diperoleh 310 perintah berhasil dieksekusi dan 18 perintah tercatat tidak sesuai, sehingga menghasilkan *success rate* keseluruhan sebesar 94,51%. Berdasarkan hasil tersebut dapat disimpulkan bahwa framework yang dikembangkan memiliki tingkat keandalan yang baik dalam mendukung proses pengendalian berbasis WebSocket secara *real-time*.

C. Pengujian *Success Rate Telemetry*

Pengujian *Telemetry Success Rate* dilakukan untuk mengevaluasi tingkat keberhasilan pengiriman data *telemetry* dari RevPi Controller menuju Dashboard NI G Web melalui RevPi Gateway. Pengujian diterapkan pada kedua skenario aplikasi untuk memastikan bahwa seluruh data sensor dan status aktuator dapat dikirim serta ditampilkan secara utuh tanpa kehilangan informasi selama proses komunikasi berlangsung. Hasil pengujian ditunjukkan pada Gambar 4.21 dan Tabel 4.5.



Gambar 4.21 Hasil Pengujian *Succes Rate Telemetry*

Tabel 4.5 Hasil Perbandingan *Succes Rate Telemetry* 3 Skenario

Skenario	Jumlah Attempt	Sesuai	Tidak Sesuai/Hilang	Success Rate (%)
Valid I/O	301	301	0	100
Mixing Tank	2676	2629	47	98
Metal Part	1188	1152	36	96

Nilai *success rate* telemetry dapat dihitung menggunakan Persamaan (4.3).

$$\boldsymbol{SR_{telemetry}}(\%) = \frac{\boldsymbol{N_{success}}}{\boldsymbol{N_{attempt}}} \times \mathbf{100}\% \qquad 4.3$$

Persamaan (4.3) digunakan untuk menghitung tingkat keberhasilan pengiriman data *telemetry*. Sebagai contoh untuk skenario *mixxing tank,* apabila jumlah data telemetry yang dikirim sebanyak 2676 data dan jumlah data yang berhasil diterima oleh Dashboard sebanyak 2629 data, dan data *lostnya* adalah sebesar 47 data. Maka nilai *success rate* dapat dihitung sebagai berikut.

$$\boldsymbol{SR_{telemetry}}(\%) = \frac{\mathbf{2629}}{\mathbf{2676}} \times \mathbf{100}\%$$
$$= \mathbf{98}\%$$

Berdasarkan hasil pengujian pada Gambar 4.21 dan Tabel 4.5, diperoleh tingkat keberhasilan pengiriman data telemetry yang tinggi pada seluruh skenario aplikasi. Skenario *Valid I/O* menghasilkan *success rate* sebesar 100%, dengan 301 data telemetry berhasil diterima sesuai dari total 301 percobaan. Sementara itu, skenario *Mixing Tank* memperoleh *success rate* sebesar 98%, dengan 2629 data sesuai dari 2676 percobaan, dan skenario *Metal Part* memperoleh *success rate* sebesar 97%, dengan 1152 data sesuai dari 1188 percobaan. Hasil tersebut

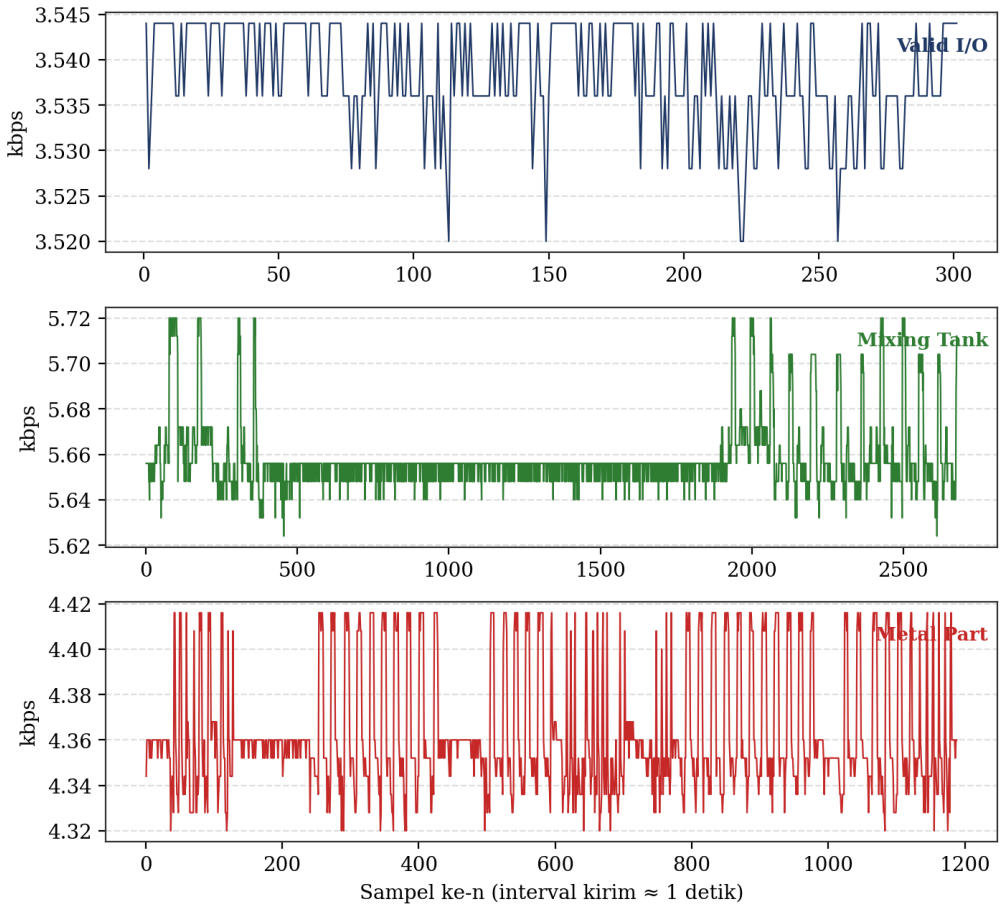
menunjukkan bahwa mekanisme komunikasi WebSocket yang diterapkan mampu mendukung proses pengiriman data *telemetry* secara andal. Tingginya nilai *success rate* pada ketiga skenario mengindikasikan bahwa data sensor, status aktuator, dan informasi proses dapat ditransmisikan dari RevPi Controller menuju Dashboard NI G Web secara konsisten selama pengujian berlangsung.

Pada skenario *Mixing Tank* dan *Metal Part* masih ditemukan sejumlah data telemetry yang tidak sesuai atau tidak tercatat pada proses validasi, masing-masing sebanyak 47 data dan 36 data. Namun, nilai *success rate* yang tetap berada di atas 95% menunjukkan bahwa sebagian besar data *telemetry* berhasil diterima dan ditampilkan oleh sistem. Kondisi tersebut tidak secara langsung mengindikasikan adanya kegagalan pada komunikasi WebSocket, melainkan dapat dipengaruhi oleh proses pengiriman, penerimaan, maupun pencatatan data *telemetry* yang berlangsung secara periodik selama sistem beroperasi.

Secara keseluruhan, dari 4165 percobaan pengiriman telemetry pada ketiga skenario diperoleh 4082 data sesuai dan 83 data tidak sesuai atau tidak tercatat, sehingga menghasilkan *success rate* keseluruhan sebesar 98,01%. Berdasarkan hasil tersebut dapat disimpulkan bahwa framework yang dikembangkan memiliki tingkat keandalan yang sangat baik dalam mendukung proses monitoring berbasis WebSocket secara *real-time*. Selain itu, hasil pengujian menunjukkan bahwa komunikasi WebSocket mampu mempertahankan proses pertukaran data *telemetry* secara konsisten pada berbagai skenario aplikasi industri.

D. Pengujian Throughput Uplink

Pengujian *Throughput Uplink* dilakukan untuk mengevaluasi laju transmisi data *telemetry* dari RevPi Controller menuju Dashboard NI G Web. Pengujian diterapkan pada kedua skenario aplikasi untuk mengetahui pengaruh jumlah variabel proses dan ukuran *payload* terhadap kebutuhan bandwidth komunikasi framework. sebagaimana ditunjukkan pada Gambar 4.22 dan Tabel 4.6 .



Gambar 4.22 Hasil Pengujian Troughput Uplink 3 Skenario

Tabel 4.6 Perbandingan troughput uplink 3 skenario

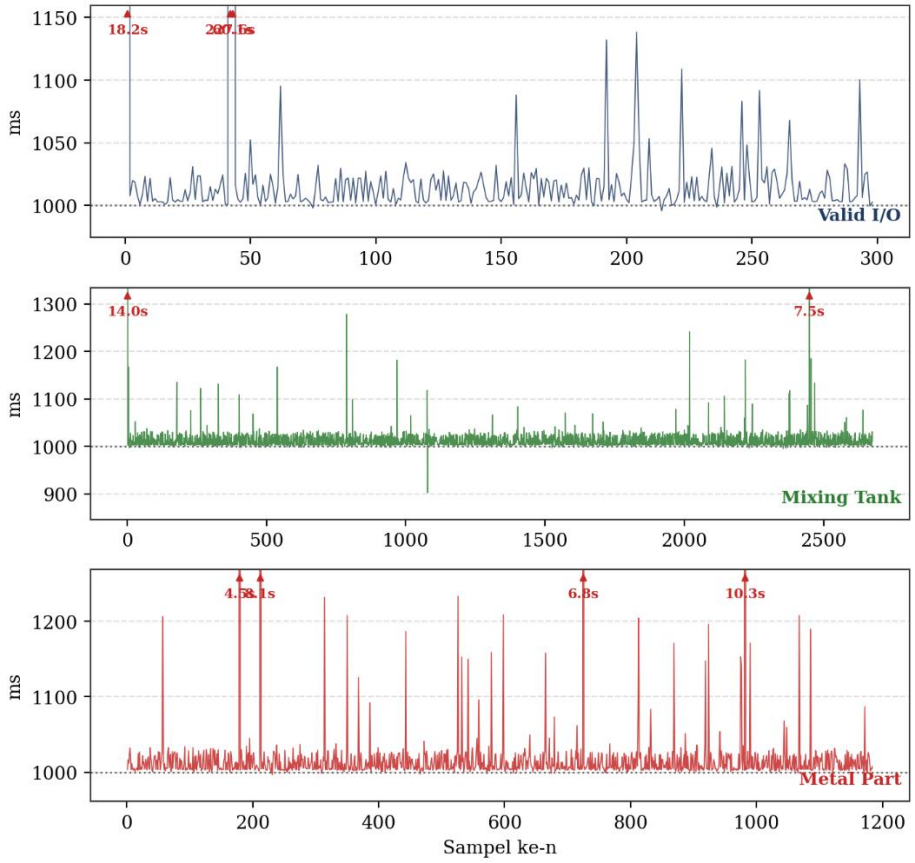
Skenario	N	Rata-rata (kbps)	Std Deviasi (kbps)	Min (kbps)	Max (kbps)
Valid I/O	301	3.538233	0.005961	3.52	3.544
Mixing Tank	2676	5.658613	0.016084	5.624	5.72
Metal Part	1188	4.362451	0.02788	4.32	4.416

Berdasarkan hasil pengujian pada Gambar 4.22 dan Tabel 4.6, diperoleh rata-rata *throughput uplink* yang berbeda pada masing-masing skenario aplikasi. Skenario *Valid I/O* menghasilkan rata-rata *throughput* sebesar 3,54 kbps, skenario *Mixing Tank* sebesar 5,66 kbps, dan skenario *Metal Part* sebesar 4,36 kbps. Hasil tersebut menunjukkan bahwa komunikasi WebSocket mampu mempertahankan proses pengiriman data telemetry secara konsisten pada seluruh skenario pengujian. Perbedaan nilai *throughput* yang diperoleh pada setiap skenario dipengaruhi oleh jumlah parameter telemetry, frekuensi perubahan data, serta ukuran payload yang dikirimkan dari RevPi Controller menuju Dashboard NI G Web.

Skenario *Mixing Tank* menghasilkan nilai *throughput* tertinggi sebesar 5,66 kbps karena melibatkan pengiriman data proses yang lebih banyak dan berlangsung secara kontinu selama pengujian. Sementara itu, skenario *Metal Part* menghasilkan *throughput* sebesar 4,36 kbps, sedangkan skenario *Valid I/O* memiliki nilai *throughput* terendah sebesar 3,54 kbps karena jumlah data yang dipertukarkan relatif lebih sedikit dibandingkan dua skenario lainnya.

E. Pengujian Stabilitas Interval Pengiriman Δt

Pengujian Pengujian stabilitas interval pengiriman data (Δt) dilakukan untuk mengevaluasi konsistensi periode pengiriman *telemetry* dari RevPi Controller menuju Dashboard NI G Web. Parameter ini digunakan untuk mengetahui kemampuan framework dalam mempertahankan interval pengiriman sesuai konfigurasi sistem meskipun dijalankan pada dua skenario aplikasi dengan kompleksitas proses yang berbeda. sebagaimana ditunjukkan pada Gambar 4.23 dan Tabel 4.7,



Gambar 4.23 Hasil Pengujian Stabilitas Interval

Tabel 4.7 Hasil Perbandingan Stabilitas Interval 3 Skenario

Skenario	Jumlah Sampel	Jeda Reconnect (>1500ms)	Rata-rata Normal (ms)	Std Deviasi Normal (ms)	Min Normal (ms)	Max Normal (ms)
Valid I/O	298	3	1014.727	19.03169	995.902	1138.304
Mixing Tank	2675	2	1012.25	15.86817	902.146	1278.413
Metal Part	1184	4	1014.52	24.75549	996.46	1233.07

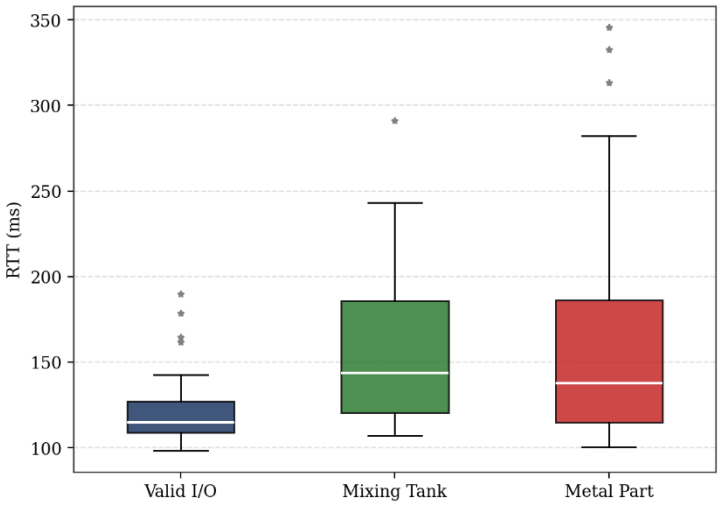
Berdasarkan hasil pengujian pada Gambar 4.23 dan Tabel 4.7, diperoleh rata-rata interval pengiriman data sebesar 1014,73 ms pada skenario *Valid I/O,* 1012,25 ms pada *Mixing Tank,* dan 1014,52 ms pada *Metal Part.* Nilai tersebut menunjukkan bahwa interval pengiriman data berada sangat dekat dengan interval target sebesar 1000 ms. Selain itu, nilai standar deviasi yang relatif kecil, yaitu 19,03 ms, 15,87 ms, dan 24,76 ms, menunjukkan bahwa variasi interval pengiriman data berada dalam rentang yang rendah sehingga kestabilan komunikasi tetap terjaga selama pengujian berlangsung.

Pada pengujian juga ditemukan beberapa kejadian *reconnect* dengan jeda pengiriman melebihi 1500 ms, yaitu sebanyak 3 kali pada Valid I/O, 2 kali pada Mixing Tank, dan 4 kali pada Metal Part. Namun, jumlah kejadian tersebut relatif kecil dibandingkan jumlah sampel yang diperoleh sehingga tidak memberikan pengaruh yang signifikan terhadap kestabilan pengiriman data secara keseluruhan.

Berdasarkan hasil tersebut dapat disimpulkan bahwa framework yang dikembangkan memiliki stabilitas interval pengiriman data yang baik dan mampu mempertahankan konsistensi pengiriman telemetry untuk mendukung proses monitoring berbasis WebSocket secara *real-time*.

F. Pengujian *Round Trip Time*

Pengujian *Round Trip Time* (RTT) dilakukan untuk mengevaluasi waktu komunikasi secara menyeluruh (*end-to-end communication*) antara Dashboard NI G Web, RevPi Gateway, dan RevPi Controller. Berbeda dengan *latency* eksekusi kontrol, parameter RTT mencakup seluruh proses komunikasi mulai dari pengiriman perintah, penerusan data oleh RevPi Gateway, eksekusi pada RevPi Controller, hingga diterimanya kembali respons oleh Dashboard. Pengujian dilakukan pada dua skenario aplikasi untuk mengetahui pengaruh kompleksitas proses terhadap waktu komunikasi keseluruhan framework.



Gambar 4.24 Hasil Pengujian Distribusi *Round Trip Time* 3 Skenario

Tabel 4.8 Ringkasan Perbandingan RTT 3 Skenario

Skenario	N	Rata-rata (ms)	Std Deviasi (ms)	Min (ms)	Max (ms)
Valid I/O	90	119.8256	15.89947	98.139	189.871
Metal Part	200	151.6675	44.87338	100.429	345.435
Mixing Tank	38	156.1415	43.81531	107.168	290.859

Berdasarkan hasil pengujian pada Gambar 4.24 dan Tabel 4.8, diperoleh nilai rata-rata *Round Trip Time* (RTT) sebesar 119,83 ms pada skenario *Valid I/O*, 156,14 ms pada *Mixing Tank*, dan 151,67 ms pada *Metal Part.* Hasil tersebut menunjukkan bahwa komunikasi WebSocket mampu mempertahankan waktu respons dua arah yang relatif rendah pada seluruh skenario pengujian.

Nilai RTT yang lebih tinggi pada skenario Mixing Tank dan Metal Part dipengaruhi oleh kompleksitas proses yang lebih besar, dimana sistem tidak hanya melakukan pertukaran data telemetry tetapi juga menjalankan logika kontrol dan pengolahan data secara bersamaan. Meskipun demikian, rata-rata RTT pada seluruh skenario masih berada di bawah 200 ms, sehingga respons komunikasi tetap dapat dikategorikan baik untuk kebutuhan monitoring dan pengendalian *real-time*.

Selain itu, nilai standar deviasi RTT sebesar 15,90 ms pada Valid I/O, 43,82 ms pada Mixing Tank, dan 44,87 ms pada Metal Part menunjukkan adanya variasi waktu respons selama pengujian. Variasi tersebut terlihat dari beberapa nilai maksimum yang mencapai 189,87 ms, 290,86 ms, dan 345,44 ms. Namun, berdasarkan distribusi data pada grafik *boxplot*, sebagian besar nilai RTT tetap terkonsentrasi pada rentang normal dan hanya terdapat beberapa nilai ekstrem (*outlier*) yang muncul selama pengujian.

Berdasarkan hasil tersebut dapat disimpulkan bahwa framework yang dikembangkan mampu mempertahankan performa komunikasi dua arah yang stabil dengan waktu respons yang relatif rendah. Hal ini menunjukkan bahwa komunikasi WebSocket yang diterapkan sesuai untuk mendukung kebutuhan monitoring dan pengendalian sistem secara *real-time* pada berbagai skenario aplikasi industri.

4.3 Analisis Sistem

Berdasarkan seluruh rangkaian pengujian yang telah dilakukan, sistem *Trainer Kit Edge Industrial Internet of Things (IIoT)* berbasis Revolution Pi berhasil menunjukkan kinerja yang baik dari aspek perangkat keras (*hardware*) maupun perangkat lunak (*software*). Integrasi antara sensor, aktuator, RevPi Controller, RevPi Gateway, dan Dashboard NI G Web mampu membentuk suatu sistem monitoring dan pengendalian yang bekerja secara terpadu dalam mendukung konsep Edge IIoT.

Dari aspek *hardware*, seluruh perangkat input dan output yang digunakan pada skenario *Valid I/O, Mixing Tank*, dan *Metal Part* dapat beroperasi sesuai dengan fungsinya. Sensor dan perangkat input berhasil diakuisisi oleh RevPi Controller, kemudian diproses menjadi data telemetry yang dapat digunakan untuk kebutuhan monitoring sistem. Selain itu, aktuator seperti indikator dan perangkat keluaran lainnya mampu merespons perintah kontrol yang diberikan baik melalui dashboard maupun melalui input lokal, sehingga menunjukkan bahwa proses akuisisi data dan pengendalian perangkat berjalan dengan baik.

Dari aspek *software*, implementasi arsitektur berbasis WebSocket berhasil mendukung komunikasi dua arah antara Dashboard NI G Web, RevPi Gateway, dan RevPi Controller secara *real-time*. Hasil pengujian menunjukkan bahwa sistem memiliki tingkat keberhasilan kontrol dan telemetry yang tinggi, waktu eksekusi kontrol yang rendah, stabilitas interval pengiriman data yang baik, serta nilai RTT yang masih berada pada rentang yang mendukung kebutuhan aplikasi industri. Selain itu, nilai *throughput* yang relatif rendah menunjukkan bahwa pertukaran data dapat dilakukan secara efisien tanpa memerlukan penggunaan bandwidth yang besar.

Implementasi arsitektur dua lapis antara RevPi Gateway dan RevPi Controller juga menunjukkan kemampuan sistem dalam memisahkan fungsi

komunikasi dan fungsi kontrol proses. RevPi Gateway berperan sebagai pusat komunikasi dan distribusi data, sedangkan RevPi Controller berfokus pada akuisisi data sensor dan eksekusi logika kontrol. Pendekatan ini memungkinkan sistem bekerja lebih terstruktur, mudah dikembangkan, serta mendukung konsep komputasi tepi (*edge computing*) yang menjadi salah satu karakteristik utama pada implementasi IIoT modern.

Secara keseluruhan, hasil pengujian membuktikan bahwa sistem yang dikembangkan mampu menjalankan siklus monitoring dan pengendalian secara utuh, mulai dari akuisisi data lapangan, pengiriman data telemetry, visualisasi pada dashboard, hingga eksekusi perintah kontrol pada perangkat aktuator. Dengan demikian, Trainer Kit Edge IIoT berbasis Revolution Pi dengan komunikasi WebSocket dan Dashboard NI G Web yang dikembangkan telah memenuhi tujuan penelitian dan layak digunakan sebagai media pembelajaran, simulasi, maupun pengembangan aplikasi Edge IIoT pada lingkungan akademik dan industri.

Berdasarkan hasil tersebut, solusi yang dikembangkan mampu menjawab permasalahan penelitian mengenai kebutuhan media pembelajaran Edge IIoT yang tidak hanya menyediakan fungsi monitoring dan pengendalian secara real-time, tetapi juga memiliki arsitektur yang modular serta mudah dikembangkan.

BAB 5
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil implementasi, pengujian, dan analisis yang telah dilakukan, maka diperoleh beberapa kesimpulan sebagai berikut.

1. *Trainer Kit Edge Industrial Internet of Things* (IIoT) berbasis Revolution Pi berhasil dirancang dengan mengintegrasikan RevPi Controller, RevPi Gateway, sensor, aktuator, dan Dashboard NI G Web dalam satu sistem yang memenuhi karakteristik sistem industri serta mendukung mode *Local* dan *Remote*.
2. Mekanisme akuisisi data sensor dan kontrol aktuator berhasil diimplementasikan pada Revolution Pi sebagai perangkat *edge*, meliputi pembacaan sensor proximity, RTD PT100, XY-MD02, dan signal generator 4–20 mA serta pengendalian LED, buzzer, motor DC, dan relay SSR sesuai logika kontrol yang dirancang.
3. Komunikasi data dua arah berbasis WebSocket antara Revolution Pi dan Dashboard NI G Web berhasil dibangun menggunakan arsitektur Controller–Gateway sehingga memungkinkan pertukaran data *telemetry* dan kontrol secara *real-time*.
4. Sistem monitoring dan kontrol berbasis web berhasil diintegrasikan sehingga data sensor dapat ditampilkan dan perintah kontrol dapat dieksekusi secara sinkron melalui Dashboard NI G Web. Integrasi sistem telah divalidasi melalui skenario *Metal Part Conveyor Monitoring System* dan *Automatic Mixing Tank Process Control System.*
5. Hasil evaluasi menunjukkan bahwa sistem mampu mendukung komunikasi data secara *real-time* dengan *control success rate* sebesar 94,51% dan *telemetry success rate* sebesar 98,01%. Selain itu, sistem menghasilkan *latency* kontrol sebesar 1,58–1,98 ms, *throughput uplink* sebesar 3,54–5,66 kbps, interval pengiriman data sebesar 1012–1015 ms, serta *RTT* sebesar 119,83–156,14 ms. Hasil tersebut menunjukkan bahwa sistem memiliki performa komunikasi yang cepat, stabil, dan andal untuk mendukung

aplikasi Edge IIoT.

5.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, beberapa saran untuk pengembangan sistem pada penelitian selanjutnya adalah sebagai berikut.

1. *Trainer kit* dapat dikembangkan dengan menambahkan skenario praktikum baru tanpa mengubah keseluruhan arsitektur sistem yang telah dibangun, sehingga fleksibilitas dan keberlanjutan pengembangan sistem dapat tetap terjaga.
2. Tampilan antarmuka Dashboard NI G Web dapat ditingkatkan dengan mengoptimalkan desain visual, seperti penggunaan elemen *Scalable Vector Graphics* (SVG) dan komponen antarmuka yang lebih interaktif agar memberikan pengalaman pengguna yang lebih baik.
3. Perangkat keras *trainer kit* dapat dioptimalkan melalui penambahan atau penyempurnaan sensor, aktuator, maupun modul I/O sehingga mampu mendukung implementasi skenario otomasi industri yang lebih kompleks.
4. Sistem komunikasi dapat dikembangkan dengan menambahkan dukungan protokol komunikasi industri selain WebSocket, seperti MQTT, OPC UA, atau Modbus TCP, sehingga *trainer kit* dapat digunakan sebagai media pembelajaran berbagai teknologi komunikasi pada *sistem Industrial Internet of Things (IIoT).*

DAFTAR PUSTAKA

Alabadi, M., Habbal, A., & Wei, X. (2022). Industrial Internet of Things: Requirements, Architecture, Challenges, and Future Research Directions. In *IEEE Access* (Vol. 10, pp. 66374–66400). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ACCESS.2022.3185049

Beňo, L., Kučera, E., Drahoš, P., & Pribiš, R. (2024). Transforming industrial automation: voice recognition control via containerized PLC device. *Scientific Reports*, *14*(1), 1–17. https://doi.org/10.1038/s41598-024-81172-w

GitHub - banscript-ssh/websocket · GitHub. (n.d.). Retrieved July 27, 2026, from https://github.com/banscript-ssh/websocket

Halenar, I., Halenarova, L., Tanuska, P., & Vazan, P. (2024). Machine Condition Monitoring System Based on Edge Computing Technology. *Sensors*, *25*(1). https://doi.org/10.3390/s25010180

Heraeus Sensor Technology. (2016). *Platinum Resistance Temperature Detector SMD 0603 (V) Heraeus Sensor Technology USA*. www.hst-us.com

Hlayel, M., Mahdin, H., & Adam, H. A. M. (2025). Latency Analysis of WebSocket and Industrial Protocols in Real-Time Digital Twin Integration. *International Journal of Engineering Trends and Technology*, *73*(1), 120–135. https://doi.org/10.14445/22315381/IJETT-V73I1P110

Kucera, E., Haffner, O., & Janecky, D. (2022). Monitoring of Discrete-event System Controlled by Revolution Pi Using 3D Engine. *Proceedings of the 31st International Conference on Cybernetics and Informatics, K and I 2022*. https://doi.org/10.1109/KI55792.2022.9925937

Mahmood, Zaigham. (2019). *The Internet of Things in the industrial sector : security and device connectivity, smart environments, and industry 4.0* (C. and C. Networks, Ed.). Springer. https://doi.org/https://doi.org/10.1007/978-3-030-24892-5

Melnikov, A., & Fette, I. (2020). The WebSocket Protocol. *Internet Engineering Task Force (IETF)*.

Pimentel, V., & Nickerson, B. G. (2012). *Programmatic Web Interfaces Communicating and Displaying Real-Time Data with WebSocket*. http://rxtx.qbang.

PLC4Training. (2023). *RevPI, Raspberry PI-Trainer, SET PLC training system*. https://www.plc4training.com/en/revpi-trainingsboard-v1206pi-s1.html

Sisinni, E., Saifullah, A., Han, S., Jennehag, U., & Gidlund, M. (2018). Industrial internet of things: Challenges, opportunities, and directions. *IEEE Transactions on Industrial Informatics*, *14*(11), 4724–4734. https://doi.org/10.1109/TII.2018.2852491

Syahrudin, A. N., & Kurniawan, T. (2018). *INPUT DAN OUTPUT PADA BAHASA PEMROGRAMAN PYTHON*.

techsquare. (2023). *RevPi Next+ Training Kit*. https://www.techsquare.co.th/product/revpi-next-plus-training-kit/

Tidrea, A., Korodi, A., & Silea, I. (2023). Elliptic Curve Cryptography Considerations for Securing Automation and SCADA Systems. *Sensors*, *23*(5), 1–19. https://doi.org/10.3390/s23052686

Younan, M., Houssein, E. H., Elhoseny, M., & Ali, A. A. (2020). Challenges and recommended technologies for the industrial internet of things: A comprehensive review. *Measurement*, *151*, 107198. https://doi.org/10.1016/j.measurement.2019.107198

Zhang, P., Wu, Y., & Zhu, H. (2020). Open ecosystem for future industrial internet of things (IIoT): Architecture and application. *CSEE Journal of Power and Energy Systems*, *6*(1), 1–11. https://doi.org/10.17775/CSEEJPES.2019.01810

LAMPIRAN A

Program Python

1. Main Gateway

```
# ==== # IMPORT LIBRARY # ==== #
import argparse
import asyncio
import logging
import time
from app.server import WebSocketGatewayServer
from app.discovery import broadcast_beacon
logger = logging.getLogger("main")

# ==== # MAIN ENTRY # ==== #
async def main(host: str, port: int):
    server = WebSocketGatewayServer(host=host, port=port)
    await server.start()
    print(f"[GATEWAY] Server running on ws://{host}:{port}")
    print(f"[GATEWAY] Startup timestamp: {time.time()}")
    # Monitoring task
    heartbeat_task =
asyncio.create_task(server.monitor_clients())
    # Beacon task - biar controller bisa auto-discover IP
gateway ini
    beacon_task =
asyncio.create_task(broadcast_beacon(port))
    try:
        while True:
            await asyncio.sleep(3600)
    finally:
        print("[GATEWAY] Stopping server...")
        heartbeat_task.cancel()
        beacon_task.cancel()
        await server.stop()

# ==== # RUN # ====#
if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--host", default="0.0.0.0")
    parser.add_argument("--port", type=int, default=8765)
    parser.add_argument("--log", default="info")
    args = parser.parse_args()
    level = getattr(logging, args.log.upper(), logging.INFO)
    logging.basicConfig(
        level=level,
        format="%(asctime)s %(levelname)s %(name)s:
%(message)s",
    )
    asyncio.run(main(args.host, args.port))
```

2. Main Controller

```
# ==== # IMPORT LIBRARY # ==== #
import argparse
import asyncio
import logging
from revpi.client_revpi import run_client, set_controller_id
from revpi.dispatcher import run_dispatcher
from revpi.md02_poller import run_md02_poller
from revpi.logging import (
    init_db,
    init_csv,
)
logger = logging.getLogger("main")

# ==== # ASYNC MAIN RUNTIME # ==== #
async def async_main(
    host: str,
    port: int,
    controller_id: str,
    mode: str,
    gate: str,
):
    init_db()
    init_csv()
    set_controller_id(controller_id)
    logger.info("Controller logger initialized")
    logger.info("Controller ID : %s", controller_id)
    logger.info("Training Mode : %s", mode.upper())
    if mode == "logic":
        logger.info("Logic Gate   : %s", gate.upper())
    while True:
        try:
            await asyncio.gather(
                # WebSocket Communication
                run_client(
                    host,
                    port,
                ),
                # Training Logic
                run_dispatcher(
                    mode=mode,
                    gate=gate,
                ),
                run_md02_poller(),
            )
        except Exception as e:
            logger.exception(
                "Controller disconnected: %s",
                e
            )
        logger.warning("Reconnect in 3 seconds...")
        await asyncio.sleep(3)
```

```
# ====# MAIN ENTRY # ===== #
def main():
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--host",
        default="192.168.1.106"
    )
    parser.add_argument(
        "--port",
        type=int,
        default=8765
    )
    parser.add_argument(

        default="revpi01"
    )
    parser.add_argument(
        "--mode",
        default="io",
        choices=[
            "io",
            "logic",
            "traffic",
            "process",
            "conveyor"
        ],
        help="Training mode"
    )
    parser.add_argument(
        "--gate",
        default="and",
        choices=[
            "and","or","xor",
            "nand","nor","xnor",
            "not"
        ],
        help="Logic Gate"
    )
    parser.add_argument(
        "--log",
        default="info"
    )
    args = parser.parse_args()
    level = getattr(
        logging,
        args.log.upper(),
        logging.INFO
    )
    logging.basicConfig(
        level=level,
        format="%(asctime)s %(levelname)s %(name)s:
%(message)s"
    )
```

```
    logger.info("=" * 50)
    logger.info("RevPi Controller Starting")
    logger.info("Host         : %s", args.host)
    logger.info("Port         : %d", args.port)
    logger.info("Controller   : %s", args.id)
    logger.info("Mode         : %s", args.mode.upper())
    logger.info("=" * 50)
    try:
        asyncio.run(
            async_main(
                args.host,
                args.port,
                args.id,
                args.mode,
                args.gate,
            )
        )
    except KeyboardInterrupt:
        logger.info("Shutting down RevPi Controller...")

# ==== # RUN PROGRAM # ==== #
if __name__ == "__main__":
    main()
```

LAMPIRAN B

Program G Web Development Software

1. Main Program