PERANCANGAN SISTEM *MONITORING REAL-TIME* UNTUK PEMELIHARAAN *DIES* DALAM PROSES PRODUKSI OTOMOTIF BERBASIS IOT DENGAN GRAFANA

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Strata-1
Program Studi Teknik Industri - Fakultas Teknologi Industri
Universitas Islam Indonesia

Nama : Fildzah Shabrina
No. Mahasiswa : 22522373

PROGRAM STUDI TEKNIK INDUSTRI PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2026

PERNYATAAN KEASLIAN

Saya mengakui bahwa tugas akhir ini adalah hasil karya saya sendiri kecuali kutipan dan ringkasan yang seluruhnya sudah saya jelaskan sumbernya. Jika dikemudian hari ternyata terbukti pengakuan saya ini tidak benar dan melanggar peraturan yang sah maka saya bersedia ijazah yang telah saya terima ditarik kembali oleh Universitas Islam Indonesia.

Yogyakarta, 1- Juli -2026



(Fildzah Shabrina)

22522373

SURAT BUKTI PENELITIAN

UNIVERSITAS ISLAM INDONESIA

FAKULTAS
TEKNOLOGI INDUSTRI

Gedung KH. Was Mansur
Kampus Terpadu Universitas Islam Indonesia
Jl. Kaliurang km 14,5 Yogyakarta 55584
T (0274) 898444 ext. 4110, 4100
F (0274) 895007
E ft@uii.ac.id
W ft.uii.ac.id

SURAT KETERANGAN PENELITIAN

Nomor : 296/Ka.lab SIMANTI/20/Lab.SIMANTI/VI/2026

Assalamu'alaikum Warohmatullahi Wabarokaatuh

Dengan hormat,

Yang bertanda tangan dibawah ini, menerangkan bahwa:

Nama : Fildzah Shabrina

Nim : 22522373

Jurusan : Teknik Industri

Dosen Pembimbing : Putri Dwi Annisa, S.T., M.Sc.

Menyatakan bahwa mahasiswa tersebut diatas telah melaksanakan penelitian tugas akhir dengan judul **"PERANCANGAN SISTEM *MONITORING REAL-TIME* UNTUK PEMELIHARAAN *DIES* DALAM PROSES PRODUKSI OTOMOTIF BERBASIS IOT DENGAN GRAFANA"** mulai pelaksanaan penelitian 11 Maret 2026 sampai 4 Mei 2026.

Demikian surat keterangan penelitian ini kami buat. Atas perhatiannya dan kerja samanya kami mengucapkan terima kasih.

Wassalamu'alaikum Warohmatullahi Wabarokaatuh

Yogyakarta, 18 Juni 2026
Kepala Laboratorium
Sistem Manufaktur Terintegrasi

Putri Dwi Annisa, S.T., M.Sc.

LEMBAR PENGESAHAN PEMBIMBING

PERANCANGAN SISTEM *MONITORING REAL-TIME* UNTUK PEMELIHARAAN *DIES* DALAM PROSES PRODUKSI OTOMOTIF BERBASIS IOT DENGAN GRAFANA

TUGAS AKHIR

Disusun Oleh :

Nama : Fildzah Shabrina

No. Mahasiswa : 22522373

Yogyakarta, 1 Juli 2026

Dosen Pembimbing

(Putri Dwi Annisa, S.T., M.Sc)

LEMBAR PENGESAHAN DOSEN PENGUJI

PERANCANGAN SISTEM *MONITORING REAL-TIME* UNTUK PEMELIHARAAN *DIES* DALAM PROSES PRODUKSI OTOMOTIF BERBASIS IOT DENGAN GRAFANA

TUGAS AKHIR

Disusun Oleh :

Nama : Fildzah Shabrina

No. Mahasiswa : 22522373

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata-1 Teknik Industri Fakultas Tekonologi Industri Universitas Islam Indonesia

Yogyakarta, 1 Juli 2026

Tim Penguji

Putri Dwi Annisa, S.T., M.Sc. Ketua	
Ir. Muchamad Sugarindra, S.T., M.T.I., IPM. Anggota I	
Sri Indrawati, S.T., M.Eng. Anggota II	

Mengetahui,
Ketua Program Studi Teknik Industri Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

Ir. Muhammad Ridwan Andi Purnomo, S.T., M.Sc., Ph.D., IPM.
NIK. 015220101

HALAMAN PERSEMBAHAN

Laporan Tugas Akhir ini dipersembahkan sebagai wujud syukur kepada Allah SWT, dengan izin Ridho-Nya serta bakti kepada kedua orang tua atas segala doa dan dukungan yang tidak pernah terputus, serta penghargaan kepada diri saya atas seluruh proses pembelajaran, perjuangan, dan ketekunan yang telah dilalui hingga mencapai tahap ini. Karya ini menjadi salah satu langkah untuk mewujudkan cita-cita saya sejak kecil, yaitu menciptakan sesuatu yang dapat memberikan manfaat bagi banyak orang melalui penerapan ilmu pengetahuan dan teknologi, hingga akhirnya dapat diwujudkan melalui penelitian dan pengembangan sistem pada Tugas Akhir ini.

MOTTO

"Allah tidak membebani seseorang melainkan sesuai dengan kesanggupannya."
(QS. Al-Baqarah: 286)

"Selalu ada harga yang harus dibayar untuk sebuah impian."

— Tere Liye

"Akan ada pelajaran dalam setiap perjalanan, bahkan pada perjalanan yang terasa paling sulit."

— Tere Liye

KATA PENGANTAR

Bismillahirrahmanirrahim
Assalamu'alaikum Warahmatullahi Wabarakaatuh
Alhamdulillahirabbil'alamiin, segala puji dan syukur penulis panjatkan ke hadirat Allah SWT atas segala rahmat, karunia, dan nikmat-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **"Perancangan Sistem *Monitoring Real-time* untuk Pemeliharaan *Dies* dalam Proses Produksi Otomotif Berbasis IoT dengan Grafana"** sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Industri, Fakultas Teknologi Industri, Universitas Islam Indonesia. Shalawat serta salam senantiasa tercurah kepada junjungan Nabi Muhammad SAW yang telah membawa umat manusia menuju zaman yang penuh dengan ilmu pengetahuan dan peradaban.

Selama proses penyusunan Tugas Akhir, penulis memperoleh banyak pengalaman dan pembelajaran, baik dalam aspek akademik maupun pengembangan diri. Proses perancangan sistem, integrasi perangkat keras dan perangkat lunak, pelaksanaan pengujian, hingga penyusunan laporan menjadi perjalanan yang penuh tantangan dan memerlukan ketekunan. Penulis menyadari bahwa penyelesaian Tugas Akhir ini tidak terlepas dari bantuan, dukungan, doa, dan bimbingan dari berbagai pihak. Oleh karena itu, dengan segala kerendahan hati penulis menyampaikan rasa terima kasih kepada:

1. Allah SWT, yang dengan segala rahmat, karunia, dan pertolongan-Nya telah memberikan kekuatan, kesabaran, serta kemudahan kepada penulis dalam menghadapi setiap proses hingga dapat menyelesaikan Tugas Akhir ini.
2. Bapak Prof. Dr. Sri Kusumadewi, S.Si., M.T. selaku Dekan Fakultas Teknologi Industri, Universitas Islam Indonesia.
3. Bapak Dr. Drs. Imam Djati Widodo, M.Eng.Sc., selaku Ketua Jurusan Teknik Industri Fakultas Teknologi Industri Universitas Islam Indonesia
4. Bapak Ir. Muhammad Ridwan Andi Purnomo, S.T., M.Sc., Ph.D., IPM. selaku Ketua Program Studi Teknik Industri Program Sarjana Fakultas Teknologi Industri, Universitas Islam Indonesia.
5. Ibu Putri Dwi Annisa, S.T., M.Sc. selaku dosen pembimbing yang telah memberikan arahan, bimbingan, ilmu, motivasi, serta masukan yang sangat berharga selama proses penelitian dan penyusunan Tugas Akhir ini.
6. Kedua orang tua penulis, Bapak M. Ali dan Ibu Yunita Alviati serta saudara Muhammad Syafiq yang selalu memberikan do'a, motivasi, dan dukungan moril maupun materil kepada penulis
7. Teman-teman Asisten Seperjuangan di Laboratorium SIMAN 2022, yaitu Dinda, Daji, Yudhis, Alung, Sulthan dan Aji yang telah bersama-sama berjuang, memberikan semangat dan motivasi dalam menyelesaikan Tugas Akhir ini.
8. Zahra, Irma, Elip, Vira yang telah menjadi teman berdiskusi, berbagi pengalaman, saling memberikan semangat, serta menemani proses penyusunaan Tugas Akhir.
9. Teman-teman keluarga besar Tamasya dan Bank Soal yang senantiasa memberikan doa, dukungan, dan semangat kepada penulis selama proses penyusunan Tugas Akhir
10. Bapak Heri Susilo serta Asisten Laboratorium Sistem Manufaktur 2023, yang memberikan semangat dan dukungan kepada penulis.
11. Seluruh pihak yang tidak dapat disebutkan satu per satu yang telah memberikan do'a, dukungan kepada penulis dalam menyelesaikan Tugas Akhir.

Penulis menyadari bahwa laporan Tugas Akhir ini masih memiliki keterbatasan dan jauh dari sempurna dan penulis berharap penelitian ini dapat memberikan manfaat dalam masyarakat. Semoga kebaikan yang diberikan oleh semua pihak kepada penulis menjadi amal shaleh yang senantiasa mendapat balasan dan kebaikan yang berlipat dari *Allah Subhanahu Wa Ta'ala* Aamiin.

Yogyakarta, 1 Juli 2026



Fildzah Shabrina

NIM 22522373

ABSTRAK

Pemantauan penggunaan *dies* pada proses *High Pressure Die Casting* masih banyak mengandalkan pencatatan manual sehingga informasi mengenai kondisi pemakaian *dies* belum tersedia secara *real-time*. Keterbatasan ini memicu kurang optimalnya pelaksanaan pemeliharaan preventif (*preventive maintenance*) sekaligus menghambat proses evaluasi serta pengambilan keputusan berbasis kondisi riwayat *dies*. Untuk mengatasi permasalahan tersebut, penelitian ini merancang sistem *monitoring* pemeliharaan *dies* berbasis *Internet of things* (IoT) dengan pendekatan *Business Intelligence Operational* menggunakan metode *Prototype* melalui integrasi ESP32, MQTT Broker, Node-RED, InfluxDB, dan Grafana. Pengujian dilakukan terhadap akurasi pembacaan *shot*, latency pengiriman data, penyimpanan data, visualisasi *dahboard*, notifikasi Telegram, serta kemampuan *monitoring* lebih dari satu *dies*. Hasil pengujian menunjukkan bahwa sistem dapat menyajikan informasi penggunaan *dies* secara *real-time*, meliputi jumlah *shot, cycle shot, lifetime shot, usage percentage,* PM *remaining,* PM *count, status dies, history maintenance,* dan *history alert* secara *real-time*. Selain itu, sistem berhasil mengirimkan notifikasi melalui *dahboard* Grafana dan Telegram ketika parameter penggunaan *dies* mencapai kategori *WARNING*, *DUE*, dan *CRITICAL* sesuai standar *preventive maintenance* perusahaan. Selain itu, sistem berhasil membedakan data penggunaan setiap *dies* secara terpisah sehingga mendukung proses *monitoring preventive maintenance* secara lebih mudah, terstruktur, dan *real-time* dibandingkan pencatatan manual.

Kata Kunci**:** *Internet of things, Business Intelligence*, *Monitoring Dies, Preventive maintenance*, Grafana.

DAFTAR ISI

PERNYATAAN KEASLIAN ..... **ii**
SURAT BUKTI PENELITIAN ..... **iii**
LEMBAR PENGESAHAN PEMBIMBING ..... **iv**
LEMBAR PENGESAHAN DOSEN PENGUJI ..... **v**
HALAMAN PERSEMBAHAN ..... **vi**
MOTTO ..... **vii**
KATA PENGANTAR ..... **viii**
ABSTRAK ..... **x**
DAFTAR ISI ..... **xi**
DAFTAR TABEL ..... **xiii**
DAFTAR GAMBAR ..... **xiv**
BAB I PENDAHULUAN **1**
1.1 Latar Belakang 1
1.2 Rumusan Masalah 5
1.3 Tujuan Penelitian 5
1.4 Manfaat Penelitian 5
1.4.1 Bagi Perusahaan 5
1.4.2 Bagi Akademik 6
1.5 Batasan Penelitian 6
BAB II TINJAUAN PUSTAKA **8**
2.1 Kajian Literatur 8
2.2 Landasan Teori 11
2.2.1 *Die Casting* 11
2.2.2 IoT 12
2.2.3 *Business Intelligence* 13
2.2.4 Grafana 17
2.2.5 *Maintenance* 18
2.2.6 *Prototype* 19
BAB III METODE PENELITIAN **21**
3.1 Objek Penelitian 21
3.2 Jenis dan Sumber Data 21
3.2.1 Jenis Data 21
3.2.2 Sumber Data 22
3.3 Metode Penelitian 22
3.3.1 Mekanisme Sistem 22
3.3.2 Kerangka Konseptual 23
3.3.3 Metode *Prototype* 25
3.4 Analisis Kebutuhan Sistem 26
3.4.1 Identifikasi Kebutuhan Pengguna 26
3.4.2 Standarisasi *Preventive Maintenance* 27
3.4.3 Indikator Kinerja Sistem 28
3.5 Alur Penelitian 28
BAB IV PEMBANGUNAN SISTEM **32**
4.1 Pengumpulan Data 32
4.1.1 Deskripsi Umum Sistem 32

4.1.2 Identifikasi Kebutuhan Pengguna 34
4.1.3 Data Sistem 34
4.2 Perancangan Sistem 37
4.2.1 Struktur Perancangan Sistem 38
4.2.2 *Wiring* Rancangan Sistem 39
4.2.3 Implementasi Perangkat IoT (ESP32) 41
4.3 Arsitektur dan Pengolahan Data *Operational* 45
4.3.1 Arsitektur Sistem *Monitoring* 45
4.3.2 Akuisisi Data (*Input*) 47
4.3.3 Pengiriman dan Pengolahan Data (*Process*) 48
4.3.4 Visualisasi dan Penyajian Informasi (*Output*) 51
4.4 Alur Kerja Sistem 52
4.4.1 Alur Penggunaan Sistem Pengguna 56
4.4.2 Ketentuan *Preventive Maintenance* 58
4.4.3 Pengelolaan Master Data 59
4.5 Perancangan *Hardware* 61
4.5.1 Komponen *Hardware* 62
4.5.2 Penempatan Sensor Pada *Dies* 63
4.5.3 Perbandingan Sistem *Prototype* dan Implementasi Industri 63
4.6 Perancangan *Software* dan Data *Flow* 64
4.6.1 *Software* sistem *monitoring* 65
4.6.2 Data *Flow* Sistem *Monitoring* 66
4.6.3 Struktur Data *Monitoring* 67
4.7 Perancangan *Dahboard Monitoring* 68
4.7.1 *Dahboard Machine* Status *Dies* 69
4.7.2 *Dahboard Production Monitoring* 70
4.7.3 *Dahboard Preventive Maintenance* 71
4.7.4 *Dahboard History* dan *Alert* 72
BAB V PENGUJIAN SISTEM DAN PEMBAHASAN 73
5.1 Pengujian Sistem Data 73
5.1.1 Pengujian Akuisisi Data 73
5.1.2 Pengujian Pengiriman dan Penyimpanan Data 76
5.1.3 Pengujian *Dahboard Monitoring* 80
5.1.4 Pengujian *Alert Preventive Maintenance* 84
5.1.5 Pengujian *Monitoring Multi Dies* 90
5.1.6 Pengujian *Latency* Sistem *Monitoring* 93
5.2 Pembahasan 94
5.2.1 Analisis Kinerja Sistem *Monitoring* berbasis IoT 95
5.2.2 Analisis Implementasi IoT dan *Business Intelligence* 96
5.2.3 Analisis *Preventive* dan *Alert Maintenance* 97
5.2.4 Analisis *Monitoring Multi-Dies* 99
5.2.5 Implikasi Hasil Penelitian pada Sistem *Monitroring Dies* 100
BAB VI PENUTUP 103
6.1 Kesimpulan 103
6.2 Saran 103
DAFTAR PUSTAKA 105
LAMPIRAN A-1

DAFTAR TABEL

Tabel 2. 1 Kajian Literatur 9
Tabel 3. 1 Indikator Kinerja Sistem 28
Tabel 4. 1 Hasil Kebutuhan Sistem 34
Tabel 4. 2 Data Kebutuhan Sistem 35
Tabel 4. 3 Pin ESP32 40
Tabel 4. 4 Status *Preventive* 50
Tabel 4. 5 Ketentuan *Preventive Maintenance* 59
Tabel 4. 6 Acuan Parameter *Preventive Maintenance* 60
Tabel 4. 7 Tahapan Perubahan Parameter *Preventive maintenance* 60
Tabel 4. 8 Spesifikasi *Hardware Prototype* 62
Tabel 4. 9 Spesifikasi Sistem 64
Tabel 4. 10 *Software* Sistem *Monitoring* 65
Tabel 4. 11 Struktur Data *Monitoring* 67
Tabel 4. 12 Konsep *measurement, tag* dan *field* dalam *influx* 67
Tabel 4. 13 Status *Operational Dies* 68
Tabel 5. 1 Hasil Pengujian Status PM 89
Tabel 5. 2 Hasil Pengujian *Multi-Dies* 93
Tabel 5. 3 Uji *Latency* 93
Tabel 5. 4 Klasifikasi Status *Dies* 98
Tabel 5. 5 Improvisasi Penelitian 101

DAFTAR GAMBAR

Gambar 1. 1 Frekuensi *Trouble Dies* 4
Gambar 3. 1 Mekanisme Sistem 23
Gambar 3. 2 Kerangka Penelitian dikembangkan dari model (Garcia et al.) 23
Gambar 3. 3 Model *Prototype* 25
Gambar 3. 4 Alur Penelitian 29
Gambar 4. 1 Struktur sistem 38
Gambar 4. 2 *Wiring* ESP32 40
Gambar 4. 3 Implementasi *Hardware* 42
Gambar 4. 4 Implementasi MQTT 43
Gambar 4. 5 MQTT *Running* 43
Gambar 4. 6 Implementasi Node-RED 44
Gambar 4. 7 Arsitektur Sistem 46
Gambar 4. 8 Alur akuisisi data 48
Gambar 4. 9 *Flow* Pengolahan Data 48
Gambar 4. 10 Implementasi *Flow* Node-Red 49
Gambar 4. 11 Informasi *Extract* 50
Gambar 4. 12 *Transform* data 50
Gambar 4. 13 Alur Visualisasi Informasi *Monitoring* 52
Gambar 4. 14 Alur Sistem Proses *Monitoring* 54
Gambar 4. 15 *Activity Diagram* 57
Gambar 4. 16 Tahapan Parameter pada *Function Node* 61
Gambar 4. 17 Penempatan *LimitSwitch Dies* 63
Gambar 4. 18 Data *Flow* Sistem 66
Gambar 4. 19 Status *Dies* 69
Gambar 4. 20 *Monitoring Production* 70
Gambar 4. 21 *Monitoring Maintenance* 71
Gambar 4. 22 *Monitoring History* dan *Alert* 72
Gambar 5. 1 Scan *Barcode* 74
Gambar 5. 2 Hasil serial moni*t*or 74
Gambar 5. 3 *Trigger limit switch* 75
Gambar 5. 4 Hasil serial monitor *shot* 75
Gambar 5. 5 Hasil *Trigger* berulang 76
Gambar 5. 6 Hasil data ESP32 77
Gambar 5. 7 Node-RED *Connected* 77
Gambar 5. 8 Informasi *active dies* 78
Gambar 5. 9 Informasi data masuk *shot dies* 79
Gambar 5. 10 Data Explorer InfluxDB 80
Gambar 5. 11 *Monitoring* Status *Dies* 81
Gambar 5. 12 *Monitroing Production Dies* 82
Gambar 5. 13 *Monitoring* PM 82
Gambar 5. 14 *Monitoring History* dan *Alert* 83
Gambar 5. 15 Kondisi Normal 84
Gambar 5. 16 Kondisi *Warning* 85
Gambar 5. 17 Kondisi *Due Maintenance* 85

Gambar 5. 18 Kondisi *Critical* 86
Gambar 5. 19 Notifikasi Telegram *Alert Maintenance* 87
Gambar 5. 20 Status *Dies* Saat *Maintenance* 88
Gambar 5. 21 Status *Dies* Setelah *Maintenance* 88
Gambar 5. 22 *Dahboard* Setelah PM Selesai 88
Gambar 5. 23 *History Maintenance* dan *Alert* 89
Gambar 5. 24 Pergantian *Dies* 90
Gambar 5. 25 Proses *Multi dies* pada Node-RED 90
Gambar 5. 26 Status *Multi Dies* 91
Gambar 5. 27 Hasil *Monitoring Multi-Dies* 92

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan industri otomotif yang terus meningkat mendorong perusahaan manufaktur memproduksi berbagai model kendaraan baru dengan spesifikasi yang beragam. Kondisi tersebut berdampak pada meningkatnya jumlah dan variasi komponen yang harus diproduksi. Salah satu proses penting dalam industri manufaktur otomotif Adalah proses *High Pressure Die Casting* (HPDC) yang menggunakan *dies* sebagai cetakan logam dalam pembentukan komponen. Kondisi ini mengharuskan perusahaan untuk senantiasa menjaga *dies* tetap dalam kondisi baik melalui pemeliharaan yang direncanakan secara matang.

Die casting merupakan proses yang melibatkan injeksi logam cair ke dalam cetakan baja (*dies*) di bawah tekanan dan temperatur yang tinggi. Selama proses produksi berlangsung, *dies* mengalami siklus pembukaan dan penutupan secara terus-menerus serta menerima tekanan dan temperatur tinggi sehingga secara bertahap mengalami keausan. Karakteristik proses tersebut menyebabkan beban kerja yang besar berpotensi membuat *dies* rentan mengalami penurunan performa atau kerusakan jika tidak dilakukan perawatan rutin (Burggräf et al., 2024). Untuk mengantisipasi hal tersebut, perusahaan menerapkan *preventive maintenance* untuk menjaga agar mesin beroperasi dengan optimal. Akan tetapi, pelaksanaan *preventive maintenance* sering kali menghadapi tantangan, khususnya dalam penjadwalan dan prioritas perawatan. Pada perusahaan objek penelitian, *preventive maintenance* dilakukan pada interval 10.000–15.000 *inject* atau *shot* sesuai standar *operational* perusahaan, serta perawatan tidak boleh sampai melebihi batas *shot* yang telah ditetapkan supaya meminimalkan *trouble* saat produksi.

Menurut H. F. Ramadhan et al., (2022), kerusakan yang terjadi akibat keterlambatan pemeliharaan dapat menyebabkan peningkatan biaya *Operational* yang sangat besar, termasuk biaya perbaikan, penggantian komponen yang rusak, dan pembelian *dies* baru. Hal ini juga dipengaruhi oleh tingginya jumlah dan variasi *dies* yang digunakan dalam proses produksi serta karakteristik operasi mesin *die casting* yang berjalan secara kontinu. Perusahaan memiliki sekitar 95–100 *dies* yang digunakan dalam proses produksi. Menurut Lakeshore Die Cast (2026), biaya pembuatan *tooling* atau *dies* dapat berkisar antara USD 15.000 hingga USD 150.000 per unit, setara dengan ratusan juta sampai miliar rupiah, tergantung material, ukuran

dan kompleksitas *dies* yang diperlukan untuk produksi komponen otomotif yang presisi. Dengan demikian, semakin banyak jumlah dies yang digunakan, semakin besar pula nilai investasi yang harus dikelola oleh perusahaan serta kondisi tersebut menyebabkan proses pemantauan pemakaian *dies* menjadi semakin kompleks dan sulit dilakukan secara manual.

Kondisi tersebut memperlihatkan bahwa *dies* merupakan asset produksi dengan nilai investasi yang tinggi bagi perusahaan manufaktur. Keterlambatan atau ketidaktepatan dalam pelaksanaan pemeliharaan dapat menyebabkan kerusakan *dies*, yang pada akhirnya memengaruhi umur pemakaian *dies* serta meningkatkan biaya perbaikan maupun penggantian *dies*. Kerusakan *dies* juga berpotensi menimbulkan *downtime* produksi yang secara langsung menghambat pencapaian target produksi dan menimbulkan kerugian bagi perusahaan.

Dalam industri dengan produksi massal (*mass production*), seperti industri otomotif, kerusakan mendadak (*breakdown*) pada *dies* dapat mengakibatkan *downtime* yang langsung menghambat pencapaian target produksi harian dan berpotensi menimbulkan kerugian. Selain itu, tanpa pemeliharaan yang tepat, kualitas produk yang dihasilkan dapat menurun dan risiko cacat meningkat. *Preventive maintenance* diperlukan untuk mencegah terjadinya gangguan tersebut sekaligus menjaga konsistensi kualitas hasil produksi.

Industri otomotif yang diteliti dalam penelitian ini berfokus pada kendaraan roda dua merupakan salah satu sektor yang menerapkan proses *High Pressure Die Casting* (HPDC) dalam menghasilkan berbagai komponen berbahan aluminium, seperti *cylinder comp, crank case, holder cam shaft*, dan komponen lainnya. Pada proses ini, *dies* merupakan komponen utama yang digunakan secara berulang sehingga memerlukan *preventive maintenance* secara berkala.

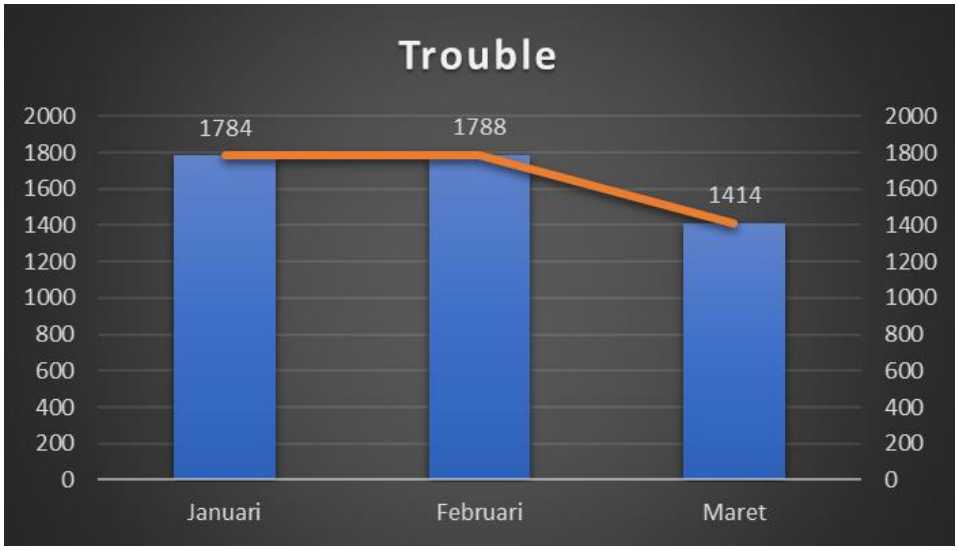
Perkembangan teknologi mendorong peluang kebutuhan sistem bagi perusahaan untuk menerapkan sistem pemantauan yang mampu menyajikan informasi penggunaan *dies* secara cepat dan akurat. Salah satu penerapannya adalah penggunaan sistem *monitoring* berbasis sensor yang dapat mendukung proses pengawasan kondisi *dies* oleh operator maupun manajer. Dengan adanya otomatisasi, pemantauan berbasis sensor, serta integrasi sistem informasi, perusahaan dapat melakukan pengawasan dan pemeliharaan mesin dengan lebih presisi dan secara *real-time* (Yuli Setiawannie & Nita Marikena, 2022).

Pada kondisi saat ini, pencatatan jumlah *shot* masih dilakukan secara manual sehingga informasi mengenai tingkat penggunaan *dies* belum dapat diperoleh secara langsung. Mekanisme tersebut menyebabkan proses pemantauan *preventive maintenance* sangat

bergantung pada pencatatan operator dan membutuhkan waktu lebih lama untuk menghasilkan informasi yang dibutuhkan. Dampaknya, keputusan terkait pelaksanaan *preventive maintenance* berpotensi mengalami keterlambatan. Oleh karena itu, diperlukan sistem *monitoring* yang mampu mengakuisisi data jumlah *shot* secara otomatis serta menyajikannya dalam bentuk informasi yang mudah dipahami sebagai dasar penentuan waktu pelaksanaan *preventive maintenance.*

Teknologi *Internet of Things* (IoT) berperan penting dalam mendukung integrasi perangkat pada sistem pemantauan *modern*. Melalui jaringan internet, sensor, mikrokontroler, dan sistem informasi dapat saling berkomunikasi sehingga data penggunaan *dies* dapat dikirimkan, diproses, dan ditampilkan secara *real-time*. emampuan tersebut memungkinkan seluruh informasi dari berbagai perangkat dikumpulkan dalam satu sistem yang terintegrasi sehingga proses pemantauan menjadi lebih efektif. Menurut Hidayat & Sari (2021), IoT memiliki potensi besar dalam mengoptimalkan berbagai aktivitas melalui kolaborasi antara perangkat pintar dan sensor cerdas yang saling terhubung dalam suatu jaringan internet.

Namun pada *industry* yang menjadi objek penelitian ini, pengaturan waktu pemeliharaan tiap-tiap *die casting* masih menjadi tantangan, proses pemantauan penggunaan *dies* masih dilakukan melalui pencatatan jumlah *shot* secara manual menggunakan lembar kerja *excel.* Data penggunaan *dies* harus diperbarui secara berkala oleh operator sehingga proses pemantauan *preventive maintenance* belum dapat dilakukan secara *real-time*. Selain itu, beragamnya jenis dan banyaknya jumlah *dies* yang digunakan dalam satu tipe produk, serta tingginya beban produksi yang membuat kesulitan dalam melakukan penjadwalan pemeliharaan yang tepat. Kondisi tersebut menyebabkan proses pemantauan memerlukan waktu lebih lama, meningkatkan potensi kesalahan pencatatan, serta menyulitkan pihak *maintenance* dalam mengetahui *dies* yang telah mendekati batas *preventive maintenance*. Selain itu, informasi penggunaan *dies* belum dapat disajikan secara otomatis dalam bentuk *dahboard* sehingga proses evaluasi masih bergantung pada pengolahan data secara manual. Berikut merupakan data dalam 3 bulan terakhir tercatat *trouble dies.*



Gambar 1. 1 Frekuensi *Trouble Dies*

Berdasarkan data perusahaan, jumlah aktivitas *repair dies* selama Januari, Februari, dan Maret 2025 masing-masing tercatat sebanyak 1.784, 1.788, dan 1.414 kali. Angka tersebut mengindikasikan bahwa frekuensi perbaikan *dies* masih tergolong tinggi sehingga dibutuhkan sistem *monitoring* yang mampu mendukung pelaksanaan *preventive maintenance* secara lebih terstruktur. Tiingkat kompleksitas pengelolaan *dies* juga dipengaruhi oleh variasi model dan tipe *dies* yang digunakan, dalam satu model produk terdapat sekitar 4 tipe *dies*, dengan masing-masing tipe terdiri dari sekitar 5 hingga 8 *dies*, sehingga dalam satu model dapat digunakan sekitar 20 hingga 27 *dies*. Secara keseluruhan, perusahaan memiliki sekitar 10 model *dies*, 6 model di antaranya hanya memakai 1 *dies* per model. Banyaknya jumlah *dies* yang harus dipantau dan bervariasinya kondisi tersebut membuat pengelolaan pemeliharaan menjadi tidak efisien serta rawan keterlambatan apabila tidak didukung sistem *monitoring* yang terintegrasi.

Dengan demikian, diperlukan suatu pendekatan yang mampu mengelola data penggunaan *dies* menjadi informasi yang terstruktur dan mudah diinterpretasikan sehingga dapat mendukung proses pengambilan keputusan dalam pelaksanaan *preventive maintenance*. Penyajian informasi pemakaian *dies* yang jelas dan mudah dipahami diharapkan dapat membantu perusahaan otomotif dalam menentukan waktu dan prioritas *preventive maintenance* sesuai dengan kondisi aktual pemakaian *dies*. Salah satu pendekatan yang dapat diterapkan adalah penerapan *Internet of Things* (IoT) dengan pendekatan *Business Intelligence Operational*. Melalui integrasi tersebut, data penggunaan *dies* dapat diakuisisi dan dipantau secara *real-time*, kemudian divisualisasikan dalam bentuk *dashboard* yang mendukung proses pemantauan dan pengambilan keputusan *preventive maintenance*.

1.2 Rumusan Masalah

Seiring dengan meningkatnya permintaan di industri otomotif, perusahaan perlu memastikan bahwa mesin-mesin yang digunakan dalam proses produksi, seperti *die casting* beroperasi secara optimal. Namun, kendala saat ini sering menghadapi tantangan yaitu tidak bisa melihat *shot* tiap-tiap *dies* secara cepat dan *real-time* dalam satu monitor, hal ini menjadikan pengelolaan jadwal pemeliharaan dan pemantauan *monitoring* masih belum efektif. Masalah ini ada dikarenakan banyaknya jenis dan jumlah *dies* yang digunakan dalam satu produksi, permintaan produksi yang tinggi serta keterbatasan sistem *monitoring* yang ada. Berdasarkan latar belakang yang diuraikan, rumusan masalah dalam penelitian ini sebagai berikut:

1. Bagaimana merancang sistem *monitoring* pemeliharaan *dies* berbasis *Internet of Things* (IoT) dengan menerapkan pendekatan *Business Intelligence Operational*?
2. Bagaimana sistem yang dirancang dapat menyajikan informasi pemakaian *dies* secara *real-time* serta memberikan peringatan ketika mendekati batas *preventive maintenance*?

1.3 Tujuan Penelitian

Tujuan penelitian ini disusun untuk menjawab rumusan masalah yang telah ditetapkan, yaitu sebagai berikut:

1. Merancang sistem *monitoring* pemeliharaan *dies* berbasis *Internet of Things* (IoT) dengan menerapkan pendekatan *Business Intelligence Operational* untuk mendukung proses pengambilan keputusan *preventive maintenance*.
2. Menghasilkan sistem yang mampu menyajikan informasi pemakaian *dies* secara *real-time* serta memberikan peringatan ketika mendekati batas *preventive maintenance*.

1.4 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan manfaat bagi perusahaan maupun dunia akademik. Adapun manfaat yang diharapkan dari penelitian ini adalah sebagai berikut:

1.4.1 Bagi Perusahaan

1. Membantu perusahaan memantau kondisi *die casting* secara *real-time* sehingga *preventive* maupun *corrective maintenance* dapat dilakukan lebih tepat waktu
2. Menurunkan risiko *downtime* yang tidak terencana akibat kerusakan *dies* yang terlambat terdeteksi.

3. Menjadi dasar pengambilan keputusan dari data *history*s jumlah *shot* dan aktivitas *maintenance* yang dapat digunakan untuk analisis, perbaikan desain *dies*, maupun evaluasi kebijakan *maintenance*.

1.4.2 Bagi Akademik

1. Menambah kajian mengenai penerapan IoT dan *dahboard* (Grafana) pada sistem pemeliharaan *die casting* di industri manufaktur
2. Menjadi referensi bagi penelitian lanjutan yang ingin mengembangkan sistem pemeliharaan dibidang *maintenance*, *management*, dan *Industrial* IoT dengan pendekatan *monitoring real-time*
3. Menyajikan studi kasus penerapan konsep industri pada konteks nyata di perusahaan, sehingga dapat mendukung pembelajaran dan penelitian di bidang teknik industri dan sistem manufaktur.

1.5 Batasan Penelitian

Batasan penelitian disusun untuk memperjelas ruang lingkup penelitian sehingga pembahasan lebih terarah. Adapun batasan dalam penelitian ini adalah sebagai berikut:

1. Penelitian difokuskan pada industri manufaktur yang mengoperasikan mesin *dies* dengan katrakteristik *mass production* serta memiliki jumlah mesin *dies* yang relatif banyak
2. Penelitian membahas penerapan pendekatan *Business Intellegence* dalam pengambilan keputusan pemeliharaan *dies* berdasarkan data hasil pemantauan pemakaian *dies.*
3. Penelitian berfokus pada perancangan dan analisis sistem *monitoring* parameter pemeliharaan *dies* pada mesin *die casting* berbasis *Internet of things* (IoT) tanpa membahas implementasi sistem dalam skala penuh di lingkungan industri.
4. Parameter pemeliharaan *dies* yang dipantau dalam penelitian ini dibatasi pada jumlah *shot* dan status pemakaian *dies*.
5. Sistem komunikasi data terbatas pada proses akuisisi dan pengiriman data dari perangkat pemantauan menuju sistem *monitoring*, tanpa membahas aspek keamanan jaringan secara mendalam.
6. Penelitian ini tidak mencakup perhitungan umur ekonomis *dies*, *predictive maintenance*, optimasi jadwal pemeliharaan, biaya *maintenance*, biaya produksi maupun analisis efisiensi produksi secara kuantitatif.

7. Penelitian ini tidak mencakup pengukuran *usability*, reability serta pengoptimalisasian perencanaan *scheduling*.
8. Pengujian sistem dalam penelitian ini dibatasi pada pengujian fungsi sistem *monitoring*, kemampuan penyajian informasi pemeliharaan *dies*, serta kemampuan sistem dalam memberikan informasi peringatan berdasarkan batas parameter yang ditetapkan.
9. *Prototype* yang dikembangkan menggunakan parameter *preventive maintenance* yang sama untuk seluruh *dies* sesuai standar *operational* perusahaan, sehingga penelitian ini belum membahas implementasi parameter *preventive maintenance* yang berbeda pada setiap jenis *dies*.

BAB II
TINJAUAN PUSTAKA

2.1 Kajian Literatur

Bab ini menyajikan tinjauan pustaka yang disusun dari berbagai sumber ilmiah, seperti artikel jurnal, penelitian terdahulu, dan literatur relevan yang berkaitan dengan topik penelitian. Tinjauan pustaka ini bertujuan sebagai landasan teori yang mendasari penelitian ini. Selain itu, pada bab ini juga membahas *state of the art* penelitian melalui analisis dan perbandingan terhadap penelitian terdahulu untuk mengidentifikasi kontribusi serta kebaruan penelitian yang dilakukan.

Sejumlah penelitian menunjukkan bahwa pemeliharaan mesin memiliki peran penting dalam menjaga kontinuitas dan efisiensi proses produksi di industri manufaktur. *Preventive maintenance* banyak diterapkan untuk mencegah terjadinya kerusakan mendadak yang dapat menyebabkan *downtime* serta peningkatan biaya *Operational*. Penelitian yang membandingkan *preventive* dan *corrective maintenance* menunjukkan bahwa pendekatan *preventive* mampu meningkatkan keandalan peralatan dan menurunkan biaya pemeliharaan secara signifikan. Beberapa studi melaporkan bahwa penerapan *preventive maintenance* dapat menurunkan biaya perawatan hingga sekitar 30% serta mengurangi *downtime* yang tidak terencana (Ardiansyah & Widjajati, 2021; Keller & Owen, 2025). Namun, sebagian besar penelitian *preventive maintenance* masih mengandalkan data *history*s dan analisis berbasis waktu, sehingga belum sepenuhnya mempertimbangkan kondisi aktual pemakaian peralatan.

Seiring dengan perkembangan teknologi, pemanfaatan IoT mulai banyak diterapkan dalam sistem *monitoring* pemeliharaan mesin. Melalui integrasi sensor dan perangkat yang saling terhubung, IoT memungkinkan akuisisi data operasional berlangsung secara otomatis sehingga kondisi mesin dapat dipantau secara berkelanjutan dan *real-time*. Beberapa penelitian menunjukkan bahwa penerapan IoT dalam *monitoring* mesin mampu meningkatkan ketersediaan informasi kondisi *Operational* serta mendukung deteksi dini terhadap potensi gangguan mesin (Bristy et al., 2025; Guerbaoui et al., 2025; Soori et al., 2023). Pada pemantauan parameter suhu mesin, penggunaan sensor IoT dilaporkan memiliki tingkat kesalahan pengukuran yang relatif rendah, dengan nilai error berada di bawah 2%, sehingga

data yang dihasilkan cukup andal untuk keperluan *monitoring* pemeliharaan (Firmansyah & Indrasari, 2025).

Dalam implementasi sistem *monitoring* berbasis IoT, visualisasi data menjadi komponen penting agar informasi yang dihasilkan dapat dipahami dengan mudah oleh pengguna. Penggunaan *dahboard* visualisasi berbasis *time-series* telah banyak diterapkan untuk menampilkan kondisi sistem secara *real-time* dan membantu pemantauan *Operational*. Beberapa penelitian menunjukkan bahwa penggunaan *dahboard* visualisasi mampu membantu identifikasi potensi gangguan dan mempercepat respons terhadap permasalahan sistem (Elradi, 2025; Garcia et al., 2022; Saputra & Marzuki, 2025; Yudha et al., 2023). Beberapa penelitian memanfaatkan platform visualisasi data berbasis *time-series*, seperti Grafana, untuk menampilkan data hasil *monitoring Operational*. Meskipun demikian, sebagian besar penerapan *dahboard* tersebut masih difokuskan pada pemantauan infrastruktur teknologi informasi atau sistem non-produksi, sehingga pemanfaatannya dalam konteks pemeliharaan *dies* di industri manufaktur masih terbatas.

Selain visualisasi data, konsep BI juga mulai dimanfaatkan dalam pengelolaan data pemeliharaan untuk mendukung pengambilan keputusan berbasis data. Melalui proses pengumpulan, pengolahan, dan penyajian data operasional, BI mampu menghasilkan informasi yang dapat dimanfaatkan dalam perencanaan, evaluasi, serta pengendalian aktivitas pemeliharaan mesin. Beberapa penelitian melaporkan bahwa penerapan BI dalam pemeliharaan industri mampu menurunkan *downtime* peralatan hingga sekitar 30–33% serta meningkatkan nilai *Mean Time Between Failure* (MTBF) secara signifikan (Fatchurramadhani, 2022; Katta, 2025; Nuraeni, 2024) Namun, pemanfaatan BI pada penelitian-penelitian tersebut umumnya masih terbatas pada analisis data *history*s dan belum terintegrasi secara langsung dengan sistem *monitoring* pemakaian mesin berbasis sensor.

Tabel 2. 1 Kajian Literatur

No	Penulis	Objek Penelitian				Metode			*Tools*		
		Mesin Produksi	***Die casting***	**IT / Jaringan**	**Lingkungan / Non-Produksi**	**IoT**	**BI**	**ML**	**Grafana**	**Power BI**	***Tableau***
1	(Keller & Owen, 2025)	✓				✓					
2	(Ardiansyah & Widjajati, 2021)	✓					✓				

No	Penulis	Objek Penelitian				Metode			*Tools*		
		Mesin Produksi	***Die casting***	**IT / Jaringan**	**Lingkungan / Non-Produksi**	**IoT**	**BI**	**ML**	**Grafana**	**Power BI**	***Tableau***
3	(Bristy et al., 2025)	✓				✓		✓	✓		
4	(Firmansyah & Indrasari, 2025)	✓				✓			✓		
5	(Guerbaoui et al., 2025)				✓	✓			✓		
6	(Elradi, 2025)			✓					✓		
7	(Saputra & Marzuki, 2025)			✓					✓		
8	(Yudha et al., 2023)	✓				✓			✓		
9	(Fatchurrama dhani, 2022)	✓					✓			✓	
10	(Katta, 2025)	✓					✓			✓	
11	(Nuraeni, 2024)	✓					✓				✓
12	(Mizgan & Ganea, 2022)	✓						✓			
13	(Garcia et al., 2022)	✓				✓			✓		
14	(Soori et al., 2023)	✓				✓					
15	(Elkateb et al., 2024)	✓				✓		✓			
16	**Penelitian Usulan**		✓			✓	✓		✓		

Dalam kajian literatur yang terlihat dalam tabel 2.1, objek penelitian dikelompokkan menjadi empat kategori, yaitu mesin produksi umum, *die casting*, infrastruktur teknologi informasi, dan lingkungan. Mesin produksi umum mencakup peralatan produksi seperti mesin mixing, genset, dan mesin *Operational* lainnya yang digunakan secara langsung dalam proses manufaktur. Kategori *die casting* mengacu pada penelitian yang secara khusus membahas *dies*

sebagai alat utama dalam proses produksi, termasuk pemantauan kondisi dan pemakaiannya. Infrastruktur teknologi informasi meliputi sistem non-produksi seperti server dan jaringan yang umumnya menjadi objek *monitoring* berbasis *dahboard*. Sementara itu, kategori lingkungan mencakup penelitian yang berfokus pada pemantauan parameter non-produksi, seperti suhu dan kelembaban, termasuk penerapannya pada sistem greenhouse atau environmental *monitoring*.

Berdasarkan hasil kajian terhadap berbagai penelitian terdahulu, terlihat bahwa sebagian besar penelitian masih berorientasi pada analisis pemeliharaan berbasis data historis atau pendekatan secara manual. Walaupun beberapa penelitian telah menerapkan IoT dan BI dalam sistem *monitoring* pemeliharaan mesin dan sistem produksi, penerapannya umumnya belum secara khusus diarahkan untuk mendukung pelaksanaan *preventive maintenance*, serta masih terbatas pada sektor tertentu seperti industri farmasi atau infrastruktur teknologi informasi.

Pemanfaatan *dahboard* visualisasi dan BI pada penelitian-penelitian terdahulu umumnya masih difungsikan sebagai alat pemantauan data, tanpa integrasi langsung dengan sistem *monitoring* pemakaian mesin berbasis sensor. Beberapa penelitian memanfaatkan *platform* visualisasi data berbasis *time-series*, seperti Grafana, untuk menampilkan data *monitoring Operational* secara *real-time*. Namun, pemanfaatan tersebut umumnya masih berdiri sendiri dan belum diintegrasikan secara spesifik untuk mendukung pengambilan keputusan *preventive maintenance* berbasis kondisi aktual pemakaian peralatan

Dalam konteks *die casting* sebagian besar penelitian lebih banyak berfokus pada analisis pemeliharaan berdasarkan data *histor*ys dan model perhitungan tertentu. Selain itu, beberapa penelitian hanya berfokus pada pemeliharaan dengan BI atau IoT tanpa mempertimbangkan integrasi antara keduanya sehingga belum memanfaatkan data jumlah *shot* serta status pemakaian *dies* secara *real-time* sebagai dasar pengambilan keputusan. Oleh karena itu, penelitian ini mengusulkan pengembangan sistem *monitoring* penggunaan *dies* berbasis IoT yang didukung dengan pendekatan *Business Intelligence Operational* untuk menyediakan informasi pemeliharaan yang terstruktur sebagai dasar dalam penentuan *preventive maintenance*.

2.2 Landasan Teori

2.2.1 *Die Casting*

Dies (cetakan logam) merupakan cetakan logam permanen yang menjadi komponen utama pada proses *die casting*. Komponen ini berfungsi membentuk logam cair menjadi produk sesuai

dengan geometri yang telah dirancang sehingga memiliki peran penting dalam menghasilkan komponen dengan dimensi dan kualitas yang sesuai spesifikasi (Kaswadi et al., 2024).

Die casting merupakan metode pengecoran logam yang dilakukan dengan menginjeksikan logam cair ke dalam rongga *dies* menggunakan tekanan tinggi. Umumnya, *dies* dibuat dari baja perkakas yang memiliki ketahanan terhadap temperatur dan tekanan selama proses produksi. Melalui metode ini, logam cair dibentuk menjadi komponen yang mendekati bentuk akhir produk sehingga proses permesinan lanjutan dapat diminimalkan. Karena itu, *die casting* menjadi salah satu metode manufaktur yang banyak digunakan untuk memproduksi komponen logam dengan tingkat presisi yang tinggi (Hidayar & Kardiman, 2024).

Dalam proses *die casting,* kelancaran produksi sangat bergantung pada kondisi mesin dan *dies* yang digunakan. Proses ini bisa terhenti apabila terjadi gangguan, baik karena *dies* mengalami kerusakan maupun mesin *die casting* yang bermasalah sehingga menimbulkan *downtime* serta mengganggu pencapaian target produksi.

2.2.2 IoT

Internet of things (IoT) merupakan konsep yang bertujuan untuk menghubungkan berbagai perangkat fisik dengan konektivitas internet sehingga perangkat saling berkomunikasi, mengirim dan menerima data, serta bekerja terintegrasi. IoT berpotensi dapat mengembangkan teknologi yang sangat potensial untuk mengoptimalkan berbagai aspek kehidupan, di mana sensor cerdas dan perangkat pintar saling berinteraksi dan bekerja sama melalui jaringan internet. (Hidayat & Sari, 2021).

2.2.2.1 Mikrokontroler ESP32

Mikrokontroler merupakan perangkat pemroses yang berfungsi menerima, mengolah, dan mengirimkan data dari sensor dalam suatu sistem berbasis IoT. Pada sistem *monitoring* industri, mikrokontroler berperan sebagai penghubung antara sensor dan sistem informasi yang lebih tinggi, seperti *server* atau *platform* visualisasi data.

Salah satu mikrokontroler yang banyak digunakan pada pengembangan sistem IoT adalah ESP32 yang dikembangkan oleh *Espressif Systems*. Dibandingkan dengan ESP8266 maupun beberapa mikrokontroler lainnya, ESP32 menawarkan kapasitas memori yang lebih besar, jumlah pin *General Purpose Input Output* (GPIO) yang lebih banyak, serta komunikasi *Wi-Fi* dan *Bluetooth Low Energy* (BLE). Keunggulan tersebut memungkinkan ESP32

menghubungkan sensor dengan jaringan internet tanpa memerlukan modul komunikasi tambahan sehingga lebih efisien dalam implementasi sistem IoT (Anggara et al., 2023).

ESP32 merupakan mikrokontroler berbasis *System on Chip* (SoC) yang mengintegrasikan prosesor, memori, modul *Wi-Fi* IEEE 802.11 b/g/n, *Bluetooth* versi 4.2, serta berbagai *peripheral* dalam satu chip. Dengan spesifikasi tersebut, ESP32 mampu menjalankan proses akuisisi data, pengolahan data, dan komunikasi jaringan secara bersamaan. Oleh sebab itu, ESP32 banyak dimanfaatkan sebagai pengendali utama dalam berbagai sistem elektronika maupun aplikasi IoT karena mampu menyediakan konektivitas internet secara langsung tanpa memerlukan perangkat tambahan (Nizam et al., 2022).

2.2.2.2 MQTT

Message Queuing Telemetry Transport (MQTT) merupakan protokol komunikasi berbasis *client-server* yang dirancang untuk pertukaran data pada sistem *Internet of Things* (IoT). MQTT berfungsi dalam mendukung pengiriman data secara *real-time* dengan latensi rendah serta penggunaan sumber daya jaringan yang relatif kecil. MQTT menerapkan model komunikasi *publish-subscribe*, di mana perangkat pengirim (*publisher*) mengirimkan data ke suatu *topic*, sedangkan perangkat penerima (*subscriber*) memperoleh data berdasarkan *topic* yang telah didaftarkan. Dengan mekanisme tersebut, komunikasi antarperangkat dapat berlangsung secara efisien meskipun perangkat memiliki keterbatasan sumber daya maupun bandwidth jaringan. Oleh karena itu, MQTT banyak digunakan pada sistem *machine-to-machine* (M2M) dan berbagai aplikasi IoT (Islamy & Wisudawati, 2023).

Implementasi MQTT memanfaatkan protokol TCP/IP sebagai lapisan *transport*, sehingga pengiriman data berlangsung dalam bentuk aliran *byte* (*bytestream*) antara *client* dan server melalui koneksi jaringan yang andal. Dengan memanfaatkan TCP sebagai media transportasi, MQTT mampu menjamin pengiriman pesan secara terstruktur sesuai mekanisme komunikasi yang telah ditentukan (Islamy & Wisudawati, 2023).

2.2.3 *Business Intelligence*

Business Intelligence (BI) merupakan pendekatan yang digunakan untuk mengolah data dari berbagai sumber menjadi informasi yang bernilai bagi proses pengambilan keputusan. Data yang diolah dapat berasal dari aktivitas internal perusahaan maupun lingkungan eksternal, termasuk kompeti*tor*, sehingga memungkinkan organisasi memperoleh gambaran menyeluruh

terhadap kondisi bisnis. Melalui proses pengorganisasian dan penyajian data yang terstruktur, BI mendukung akses informasi yang mudah dianalisis dalam berbagai aktivitas pengambilan keputusan. (Panda & Padhy, 2025).

Implementasi BI diawali dengan proses *Extract, Transform, Load* (ETL), yaitu mekanisme untuk mengambil data dari berbagai sumber operasional, mengolahnya sesuai kebutuhan analisis, kemudian menyimpannya ke dalam *data warehouse*(Kimball & Ross, 2002). Pada tahap transformasi, data dapat melalui proses validasi, agregasi, maupun penyesuaian format agar kualitas dan konsistensinya terjaga. Data yang telah tersimpan selanjutnya dianalisis menggunakan berbagai metode, termasuk *data mining*, untuk mengidentifikasi pola maupun kecenderungan tertentu yang dapat dimanfaatkan sebagai dasar penyusunan informasi. Hasil analisis tersebut kemudian disajikan kepada pengambil keputusan dalam bentuk ringkasan atau visualisasi sehingga mampu mendukung proses pengambilan keputusan yang didasarkan pada data aktual, bukan semata-mata pada intuisi atau pengalaman pengambil keputusan (Chen et al., 2012; Shapouri, 2020)

Secara umum, BI dapat dibedakan berdasarkan tujuan penggunaannya. *Business Intelligence* tradisional biasanya digunakan untuk menganalisis data *histor*ys yang bertujuan mendukung pengambilan keputusan strategis dan taktis, dengan pembaruan data yang bersifat periodik, seperti laporan mingguan atau bulanan. Menurut Watson & Wixom, (2004) menjelaskan bahwa pendekatan BI tradisional cukup efektif untuk evaluasi kinerja jangka menengah dan panjang, namun kurang sesuai untuk mendukung pengambilan keputusan yang membutuhkan respons cepat di tingkat *Operational*. Seiring meningkatnya kebutuhan industri untuk merespons kondisi sistem secara lebih cepat, pendekatan BI kemudian berkembang ke arah pemanfaatan data yang lebih aktual dan berorientasi pada kegiatan *Operational* (Chaudhuri et al., 2011).

Near real-time Business Intelligence merupakan pendekatan BI yang muncul sebagai respons terhadap meningkatnya tekanan kompetitif dalam dunia bisnis. Pendekatan ini, yang juga dikenal sebagai *Operational* BI atau *just-in-time* BI, bertujuan untuk meminimalkan jeda waktu antara proses perolehan data *Operational* dan ketersediaan hasil analisis atas data tersebut. Dengan berkurangnya latensi data, informasi dapat dimanfaatkan hampir secara langsung untuk mendukung pengambilan keputusan yang membutuhkan respons cepat terhadap kondisi aktual. Penerapan *near real-time* BI memungkinkan organisasi merespons kondisi

aktual secara proaktif, sehingga dapat meningkatkan kualitas layanan, loyalitas pelanggan, serta kinerja *Operational* perusahaan (Watson & Wixom, 2004).

BI *Operational* merupakan pendekatan BI yang berfokus pada pemanfaatan data *Operational* secara *real-time* atau *near real-time* untuk mendukung pengambilan keputusan harian (Sacu & Spruit, 2010). Pada konteks industri manufaktur, BI *Operational* sering diintegrasikan dengan sistem IoT untuk mengumpulkan data pemakaian mesin secara berkelanjutan dan menampilkannya dalam bentuk *dahboard monitoring*.

Pada lingkungan manufaktur, arsitektur *Business Intelligence Operational* dirancang untuk mengelola data *time-series* yang dihasilkan secara kontinu selama proses produksi berlangsung. Arsitektur ini mengintegrasikan seluruh proses mulai dari akuisisi data, pengolahan, penyimpanan, analisis, hingga penyajian informasi sehingga data operasional dapat dimanfaatkan sebagai dasar pengambilan keputusan secara *real-time*. Menurut Garcia et al. (2022), arsitektur *Business Intelligence* terdiri atas beberapa komponen utama yang saling terhubung, yaitu sumber data (*data source*), ETL (*Extract, Transform, Load*), media penyimpanan data, analisis, dan dashboard visualisasi. Masing-masing komponen memiliki peran yang saling melengkapi dalam mengubah data operasional menjadi informasi yang mudah dipahami oleh pengguna. Berikut penjelasan masing-masing komponen utama arsitektur *BI Operational* diantara lain:

1. Sumber Data (*Data Source*)

   Sumber data berasal dari sistem *Operational* perusahaan, seperti mesin produksi, peralatan kerja, dan sensor yang memantau kondisi atau pemakaian peralatan secara berkelanjutan. Data ini bersifat dinamis dan dihasilkan secara kontinu selama proses produksi berlangsung.

2. ETL

   ETL merupakan proses inti dalam *Business Intelligence* yang berfungsi untuk menyiapkan data sebelum digunakan dalam analisis. Proses ETL dilakukan dalam konteks *Operational*, yaitu dengan waktu pemrosesan yang singkat dan berorientasi pada data *real-time*.

   Tahap *Extract* dilakukan dengan mengambil data langsung dari sumber data *Operational*. Tahap *Transform* mencakup proses pengolahan data, seperti validasi, pengelompokan, perhitungan status, dan penyesuaian format data agar sesuai dengan

kebutuhan analisis. Selanjutnya, tahap *Load* dilakukan dengan memuat data hasil *transform*asi ke dalam sistem penyimpanan atau sistem visualisasi.

3. Data *Warehouse*

   Data *warehouse* merupakan media penyimpanan yang berfungsi sebagai tempat untuk menyimpan hasil proses ETL sehingga data dapat digunakan kembali pada proses analisis maupun visualisasi. Dalam pendekatan BI *Operational*, penyimpanan data umumnya menggunakan *data warehouse* dengan struktur multidimensi, seperti *star schema* atau *snowflake schema*, yang terdiri atas tabel fakta dan tabel dimensi untuk menyimpan data historis, misalnya data produksi, waktu operasi, dan *downtime* mesin (Inmon, 2002; Kimball & Ross, 2002).

   Namun, BI *Operational* tidak selalu memanfaatkan *data warehouse* multidimensi. Karena berorientasi pada penyajian informasi secara *real-time* atau *near real-time*, data operasional lebih sering disimpan dalam basis data yang sederhana atau basis data *time-series* yang mampu menangani data kontinu dari mesin dan sensor produksi secara efisien (Chaudhuri et al., 2011).

4. Analisis

   Komponen analisis data berfungsi untuk mengolah data yang telah disimpan menjadi informasi yang bermakna. Analisis ini bersifat sederhana dan berorientasi pada kondisi aktual, seperti perhitungan batas pemakaian, klasifikasi status, atau identifikasi kondisi abnormal. Analisis difokuskan pada pengolahan data yang mendukung pengambilan keputusan cepat dan langsung di tingkat *Operational*.

5. Visualisasi dan Penyajian Informasi

   Data yang telah dianalisis selanjutnya disajikan dalam bentuk *dahboard* atau visualisasi yang menampilkan kondisi aktual sistem secara ringkas dan mudah dipahami. Lapisan visualisasi ini menjadi antarmuka utama antara sistem BI dan pengguna, sehingga informasi yang dihasilkan dapat langsung digunakan dalam pengambilan keputusan *Operational*

Arsitektur BI dapat dipandang sebagai sistem yang mengintegrasikan sumber data, proses ETL, penyimpanan data, analisis, dan visualisasi, yang dapat diadaptasi untuk kebutuhan *monitoring* proses manufaktur berbasis data waktu nyata.

2.2.4 Grafana

Grafana merupakan *platform open-source* yang digunakan untuk keperluan *monitoring* dan visualisasi data yang mampu menghubungkan berbagai sumber data. Melalui *dahboard* interaktif, Grafana memungkinkan penyajian metrik dalam bentuk grafik, tabel, maupun bentuk visualisasi lainnya sehingga mempermudah interpretasi data *Operational*. Kemampuan integrasi lintas platform serta fitur notifikasi (*alerting*) menjadikan Grafana efektif dalam membantu pengguna menganalisis dan mengevaluasi kinerja sistem. Aplikasi ini dapat diakses melalui peramban web dengan konfigurasi awal pada port 3000, serta memerlukan pengaturan koneksi ke sumber data sebelum digunakan (Saory et al., 2025).

Grafana adalah *platform open-source* yang difokuskan pada pemantauan dan visualisasi data, terutama data berbasis *time-series*. Sistem ini dirancang untuk mengolah dan menampilkan data yang bersumber dari berbagai *database*, seperti InfluxDB, Prometheus, MySQL, maupun sistem penyimpanan data lainnya, ke dalam bentuk *dahboard* interaktif yang informatif dan mudah dipahami oleh pengguna (Elradi, 2025)

Dalam penerapan BI *Operational*, Grafana berfungsi sebagai lapisan visualisasi (*visualization layer*) yang menyampaikan informasi kondisi sistem secara *real-time* maupun *near real-time*. *Dahboard* Grafana memungkinkan pengguna untuk memantau parameter *Operational*, melihat tren data *histor*ys, serta memperoleh gambaran kondisi aktual sistem produksi dalam satu tampilan terintegrasi. Peran ini menjadikan Grafana sebagai antarmuka utama antara sistem BI dan pengguna *Operational*, seperti operator maupun manajemen pemeliharaan (Saputra & Marzuki, 2025).

Grafana menyediakan berbagai bentuk visualisasi, seperti grafik garis, indikator status, dan tabel, yang memungkinkan penyajian data secara ringkas namun informatif. Selain itu, Grafana juga mendukung fitur *alerting* yang dapat memberikan peringatan ketika nilai parameter tertentu melewati batas yang telah ditentukan. Fitur ini mendukung pengambilan keputusan *Operational* yang cepat dengan memberikan informasi kondisi abnormal secara langsung kepada pengguna. Penggunaan Grafana dalam sistem *monitoring* pemeliharaan memungkinkan informasi pemakaian peralatan disajikan secara terstruktur dan mudah dipahami, sehingga dapat mendukung proses pengambilan keputusan *preventive maintenance* berbasis kondisi aktual sistem.

2.2.5 *Maintenance*

Maintenance atau pemeliharaan merupakan serangkaian aktivitas penting dalam menjaga performa optimal peralatan perusahaan guna mencegah terjadinya gangguan maupun kerusakan yang dapat menghambat jalannya proses produksi. Pada industri manufaktur, kegiatan pemeliharaan memiliki peran penting dalam menjaga keandalan peralatan, meningkatkan efisiensi operasional, serta memastikan proses produksi dapat berlangsung secara berkelanjutan (I. Ramadhan & Fitriani, 2025)..

Secara umum, pemeliharaan dibedakan menjadi dua kategori, yaitu pemeliharaan terencana (*planned maintenance*) dan pemeliharaan tidak terencana (*unplanned maintenance*). Pemeliharaan terencana meliputi *preventive maintenance* dan *predictive maintenance* yang bertujuan mencegah terjadinya kerusakan melalui kegiatan pemeliharaan berkala maupun berdasarkan kondisi aktual peralatan. Sementara itu, pemeliharaan tidak terencana dilakukan setelah terjadi kerusakan yang menyebabkan mesin tidak dapat beroperasi sebagaimana mestinya sehingga diperlukan tindakan perbaikan secara segera.

2.2.5.1 *Breakdown Maintenance*

Breakdown maintenance merupakan kegiatan pemeliharaan yang dilakukan setelah mesin atau peralatan mengalami kerusakan sehingga tidak lagi mampu menjalankan fungsinya secara normal (Sumasto et al., 2023). Pada pendekatan ini, mesin tetap dioperasikan hingga terjadi kegagalan fungsi, kemudian dilakukan tindakan perbaikan atau penggantian komponen yang mengalami kerusakan. Dalam implementasinya, perusahaan menetapkan bahwa gangguan yang dapat diselesaikan dalam waktu maksimal 100 menit masih dikategorikan sebagai *breakdown maintenance*.

2.2.5.2 *Preventive maintenance*

Preventive maintenance merupakan strategi pemeliharaan terencana sebelum terjadi kerusakan pada mesin atau peralatan. Pendekatan ini bertujuan mempertahankan kondisi peralatan agar tetap beroperasi secara optimal, mengurangi risiko *downtime*, serta memperpanjang umur pakai mesin (Sumasto et al., 2023). Dengan melakukan pemeliharaan secara berkala, potensi kerusakan yang dapat mengganggu proses produksi dapat diminimalkan. Pada perusahaan, pelaksanaan *preventive maintenance* umumnya memerlukan waktu sekitar 1 hingga 2 hari

sesuai dengan ruang lingkup pekerjaan yang dilakukan. Secara umum, *preventive maintenance* terbagi menjadi dua jenis, yaitu:

A. *Periodic Maintenance* (Perawatan berkala)

   Periodic maintenance merupakan pemeliharaan yang dilakukan berdasarkan interval waktu tertentu, seperti harian, mingguan, bulanan, maupun tahunan. Kegiatan yang dilakukan meliputi inspeksi, pembersihan, pelumasan, serta penggantian komponen secara berkala untuk menjaga keandalan mesin dan mencegah terjadinya kerusakan yang tidak direncanakan.

B. *Predictive Maintenance* (Perawatan Prediktif)

   Predictive Maintenance merupakan metode pemeliharaan yang didasarkan pada kondisi aktual mesin. Berbeda dengan *periodic maintenance* yang mengacu pada interval waktu tertentu, pendekatan ini memanfaatkan analisis kondisi maupun tren kinerja mesin untuk memprediksi potensi kerusakan sehingga tindakan pemeliharaan dapat dilakukan sebelum terjadi kegagalan fungsi.

2.2.5.3 *Corrective maintenance*

Corrective maintenance merupakan tindakan pemeliharaan ketika ditemukan kerusakan atau penyimpangan seperti suatu produk (setengah jadi atau jadi) gagal memenuhi rencana atau syarat produksi yang menyebabkan mesin maupun proses produksi tidak dapat beroperasi sesuai dengan kondisi yang diharapkan (Sumasto et al., 2023). Tujuan utama dari pemeliharaan ini adalah mengidentifikasi penyebab kerusakan, melakukan perbaikan terhadap komponen yang bermasalah, serta mengembalikan mesin ke kondisi operasional normal. Pada perusahaan, suatu gangguan dikategorikan sebagai *corrective maintenance* apabila proses penanganannya memerlukan waktu lebih dari 100 menit.

2.2.6 *Prototype*

Metode *prototype* merupakan salah satu pendekatan dalam pengembangan perangkat lunak yang digunakan untuk menghasilkan model awal sistem sebelum proses implementasi dilakukan secara penuh. Melalui model awal tersebut, pengembang dapat menunjukkan gambaran cara kerja sistem sekaligus memperoleh umpan balik dari pengguna untuk menyempurnakan kebutuhan maupun fungsi sistem yang akan dikembangkan (Noor Dhiana et al., 2025).

Selain digunakan sebagai media demonstrasi, *prototype* juga berfungsi untuk membantu proses identifikasi kebutuhan pengguna yang belum terdefinisi secara rinci. Pendekatan ini banyak diterapkan dalam pengembangan sistem informasi ketika pengguna telah memiliki gambaran umum mengenai sistem yang diinginkan, tetapi belum mampu menjelaskan spesifikasi maupun fitur secara lengkap. Dengan adanya *prototype*, proses komunikasi antara pengembang dan pengguna menjadi lebih efektif (Syarif & Risdiansyah, 2024).

BAB III
METODE PENELITIAN

3.1 Objek Penelitian

Objek penelitian ini adalah *dies* yang digunakan pada proses *High Pressure Die Casting* (HPDC) di industri manufaktur komponen kendaraan roda dua untuk memproduksi berbagai komponen aluminium dengan jumlah *dies* yang cukup banyak. *Dies* merupakan cetakan logam yang digunakan secara berulang dalam proses produksi dan memiliki batas pemakaian tertentu sehingga memerlukan kegiatan *preventive maintenance* secara berkala untuk menjaga kualitas produk dan mencegah terjadinya kerusakan.

Industri yang menjadi objek penelitian dalam bagian *High Pressure Die Casting Maintenance* (HPDCM). Setiap model produk terdiri atas beberapa tipe *dies* dengan karakteristik penggunaan dan interval *preventive maintenance* yang berbeda. Kondisi tersebut menyebabkan proses pemantauan penggunaan *dies* secara manual menjadi kurang efektif, terutama dalam mengetahui jumlah *shot* setiap *dies* dan menentukan waktu *preventive maintenance* secara tepat.

Pada penelitian ini, implementasi sistem dilakukan pada 1 mesin *die casting* dengan 2 *dies* sebagai *prototype*. Pemilihan *prototype* bertujuan untuk memvalidasi kemampuan sistem dalam melakukan *monitoring* penggunaan *dies* secara *real-time* serta membedakan data penggunaan masing-masing *dies* secara terpisah.

Penelitian ini berfokus pada pemantauan pemakaian *dies* melalui parameter jumlah *shot* dan status pemakaian selama proses produksi berlangsung. Pemilihan *dies* sebagai objek penelitian didasarkan pada perannya dalam proses produksi serta kebutuhan akan informasi pemakaian *dies* yang terstruktur untuk mendukung kegiatan pemeliharaan.

3.2 Jenis dan Sumber Data

3.2.1 Jenis Data

Jenis data penelitian ini memanfaatkan data kuantitatif sebagai jenis data utama. Data tersebut terdiri atas beberapa komponen sebagai berikut:

1. Data jumlah *shot* pemakaian *dies* yang dihasilkan selama proses produksi

2. Data status pemakaian *dies* yang menunjukkan kondisi *dies* berdasarkan batas pemakaian *preventive maintenance*
3. Data identitas *dies* serta nilai batas *preventive maintenance* yang digunakan sebagai acuan sistem *monitoring*.

3.2.2 Sumber Data

Sumber data yang digunakan pada penelitian ini terdiri atas 2 jenis data, diantara lain:

1. Data Primer merupakan data yang diperoleh secara langsung melalui hasil pembacaan sensor terhadap jumlah *shot dies* selama proses produksi serta hasil observasi proses *monitoring*.
2. Data sekunder diperoleh dari dokumen perusahaan yang berkaitan dengan *preventive maintenance*, instruksi kerja (IK), serta literatur ilmiah dan penelitian terdahulu yang mendukung pengembangan sistem *monitoring*.

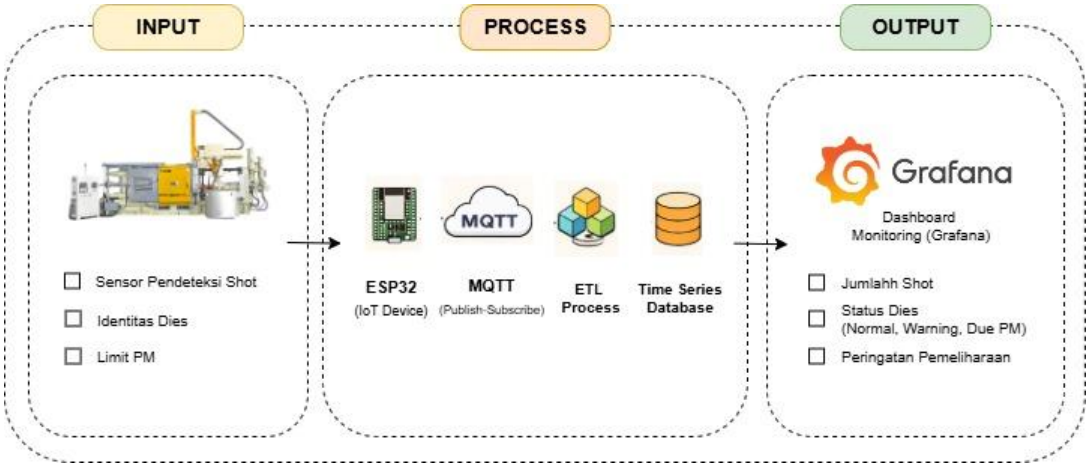
3.3 Metode Penelitian

Metode pengembangan sistem dalam penelitian ini menggunakan metode *prototype*. Metode ini dipilih karena sistem yang dikembangkan masih berupa *prototype* sehingga memungkinkan dilakukan proses evaluasi dan penyempurnaan secara bertahap sesuai kebutuhan pengguna. Melalui pendekatan ini, rancangan sistem dapat diuji terlebih dahulu sebelum diterapkan pada kondisi *operational* yang lebih luas.

Proses penelitian dilaksanakan dengan beberapa tahapan dimulai dari identifikasi kebutuhan pengguna, pengumpulan data, analisis kebutuhan sistem, perancangan *prototype*, implementasi sistem *monitoring* berbasis IoT, pengujian sistem, serta analisis terhadap hasil pengujian. Rangkaian tahapan tersebut dilakukan untuk memastikan sistem yang dikembangkan mampu memenuhi kebutuhan *monitoring preventive maintenance dies* secara *real-time*.

3.3.1 Mekanisme Sistem

Mekanisme sistem *monitoring* pada penelitian ini disusun berdasarkan pendekatan *Input–Process–Output* untuk menggambarkan alur pengolahan data pemakaian *dies* secara sistematis. Pendekatan ini digunakan untuk menjelaskan *transform*asi data dari tahap akuisisi hingga menjadi informasi yang mendukung keputusan pemeliharaan.



Gambar 3. 1 Mekanisme Sistem

Pada Gambar 3.1 menjelaskan mekanisme kerja sistem *monitoring*, diantara lain;

1. *Input*

   Data Jumlah *shot* yang dihasilkan dari siklus mesin *die casting* serta identitas *dies* yang sedang digunakan dalam proses produksi

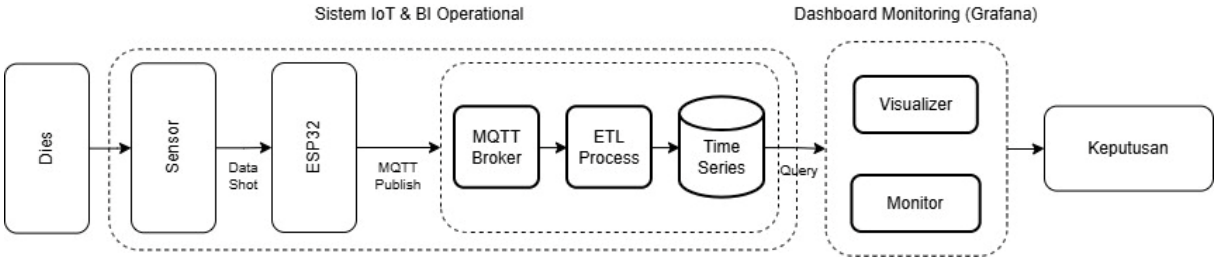
2. *Process*

   Akuisisi data oleh sensor dan ESP32, pengiriman data melalui *protocol* MQTT, Proses ETL, penyimpanan data ke basis data *time-series*, serta peringatan status pemakaian *dies* berdasarkan batas *preventive maintenance*

3. *Output*

   Informasi oemakaian *dies* berupa jumlah *shot* kumulatif, status pemakaian *dies* (normal, mendekati batas, dan perlu dilakukan *preventive maintenance*), serta peringatan pemeliharaan yang ditampilkan melalui *dahboard monitoring*.

3.3.2 Kerangka Konseptual



Gambar 3. 2 Kerangka Penelitian dikembangkan dari model (Garcia et al.)

Kerangka konseptual penelitian yang disajikan pada Gambar 3.1 diadaptasi dari penelitian Garcia et al., (2022). Kerangka tersebut menggambarkan alur penyelesaian permasalahan pada proses produksi, khususnya yang berkaitan dengan sistem *monitoring* pemeliharaan *dies*, dan

menjadi acuan dalam perancangan sistem pada penelitian ini. Permasalahan ini diatasi melalui pendekatan BI dengan mengintegrasikan IoT sebagai sistem penghubung antara *dies* dengan visualisasi *dahboard*. Sistem IoT dan BI yang digunakan dalam penelitian ini antara lain adalah:

1. Sensor

   Sensor digunakan untuk mendeteksi setiap *shot* yang terjadi selama proses produksi. Sensor berfungsi sebagai alat akuisisi data awal yang mencatat jumlah pemakaian *dies* secara langsung dari mesin.

2. ESP32

   ESP32 berperan sebagai perangkat IoT yang menerima data dari sensor. ESP32 melakukan pemrosesan awal data sebelum mengirimkannya ke sistem pusat.

3. Data *Shot*

   Data *shot* merupakan data utama yang dihasilkan dari proses produksi, yang menunjukkan jumlah siklus pemakaian *dies*. Data ini menjadi parameter utama dalam pemantauan kondisi pemakaian *dies*.

4. MQTT (*Publish-subscribe*)

   Data *shot* yang telah diterima oleh ESP32 dikirimkan ke sistem pusat menggunakan protokol komunikasi MQTT dengan mekanisme *publish–subscribe*. Mekanisme ini memungkinkan pengiriman data secara *real-time* dan efisien.

5. MQTT *Broker*

   MQTT *Broker* berfungsi sebagai perantara komunikasi yang menerima data dari ESP32 dan mendistribusikannya ke komponen pengolahan data selanjutnya dalam sistem.

6. Proses ETL (*Extract, Transform, Load*)

   Data yang diterima oleh MQTT *Broker* kemudian melalui proses ETL, yang bertujuan untuk mengekstraksi data, menyesuaikan format dan struktur data, serta mempersiapkannya untuk proses penyimpanan dan analisis.

7. Basis *Data Time-series*

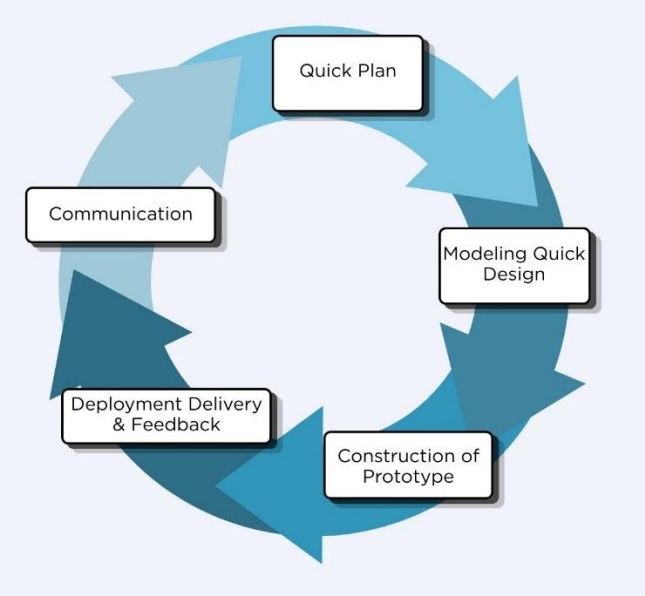
   Data hasil proses ETL disimpan dalam basis data *time-series*. Basis data ini digunakan karena data pemakaian *dies* bersifat berurutan berdasarkan waktu dan terus diperbarui selama proses produksi berlangsung.

8. *Query Data*

   Data yang tersimpan dalam basis data *time-series* diakses melalui proses *query* untuk memperoleh informasi pemakaian *dies* sesuai dengan kebutuhan pemantauan dan analisis *Operational*.

3.3.3 Metode *Prototype*

Metode *Prototype* diterapkan sebagai pendekatan pengembangan sistem karena sesuai dengan karakteristik penelitian yang masih berada pada tahap pembuatan prototipe. Melalui metode ini, sistem dapat dikembangkan, dievaluasi, dan disempurnakan secara bertahap berdasarkan kebutuhan pengguna. Menurut Kurniawan et al. (2023), metode *Prototype* terdiri dari 5 tahapan, yaitu *communication*, *quick plan*, *modeling quick design*, *construction of prototype*, serta *deployment, delivery and feedback.*



Gambar 3. 3 Model *Prototype*

1. Identifikasi Kebutuhan Pengguna (*Communication*)

   Pada tahap ini dilakukan identifikasi kebutuhan melalui observasi proses pemeliharaan *dies* dan diskusi dengan pihak yang bertanggung jawab terhadap pengelolaan *dies*. Tahap ini bertujuan untuk mengetahui kebutuhan sistem yang diharapkan, seperti *monitoring* jumlah *shot* secara otomatis, informasi kondisi *dies*, *dahboard monitoring*, serta notifikasi *preventive maintenance*.

2. Perancangan *Prototype* (*Quick Plan*)

   Berdasarkan kebutuhan yang telah diidentifikasi, dilakukan perancangan *prototype* sistem *monitoring* yang meliputi perancangan arsitektur sistem, alur komunikasi data, struktur basis data, *dahboard monitoring*, serta antarmuka pengguna. Perancangan ini menjadi dasar dalam proses implementasi sistem.

3. Pembangunan *Prototype* (*Modeling Quick Design*)

   Tahap pembangunan *prototype* dilakukan dengan mengintegrasikan perangkat keras dan perangkat lunak yang digunakan pada penelitian. Perangkat keras terdiri atas sensor dan ESP32 sebagai akuisisi data, sedangkan perangkat lunak meliputi MQTT Broker sebagai media komunikasi data, Node-RED sebagai pengolah data, InfluxDB sebagai basis data *time-series*, Grafana sebagai *dahboard monitoring*, serta Telegram sebagai media notifikasi.

4. Pengujian *Prototype* (*Construction of Prototype*)

   Prototype yang telah dibangun kemudian diuji untuk mengetahui apakah seluruh fungsi sistem berjalan sesuai dengan kebutuhan. Pengujian dilakukan terhadap akurasi pembacaan jumlah *shot*, keterlambatan pengiriman data (*latency*), keberhasilan penyimpanan data ke basis data, keberhasilan visualisasi *dahboard*, serta keberhasilan pengiriman notifikasi Telegram.

5. Evaluasi dan Penyempurnaan (*Deployment, Delivery and Feedback*)

   Hasil pengujian digunakan sebagai dasar evaluasi terhadap *prototype* yang telah dikembangkan. Apabila masih ditemukan kekurangan pada fungsi sistem, maka dilakukan penyempurnaan hingga *prototype* mampu memenuhi kebutuhan *monitoring* pemeliharaan *dies* sesuai dengan tujuan penelitian.

3.4 Analisis Kebutuhan Sistem

3.4.1 Identifikasi Kebutuhan Pengguna

Tahap identifikasi kebutuhan pengguna merupakan tahapan awal pada metode *Prototype* yang bertujuan untuk memperoleh kebutuhan sistem sebelum dilakukan proses perancangan. Pada tahap ini, identifikasi dilakukan melalui observasi terhadap proses *preventive maintenance dies* di area produksi serta diskusi dengan pihak yang bertanggung jawab dalam pengelolaan *dies*, yaitu *foreman*, *supervisor*, dan *manager*. Observasi dilakukan untuk memahami alur kerja, proses pemantauan penggunaan dies, serta mekanisme pelaksanaan *preventive maintenance*

yang sedang berjalan saat ini. Selanjutnya, diskusi dilakukan untuk menggali kebutuhan pengguna terhadap sistem yang akan dikembangkan. Beberapa aspek yang menjadi fokus dalam proses identifikasi meliputi kebutuhan informasi yang diperlukan pengguna, kendala yang dihadapi pada proses *monitoring dies*, fitur yang diharapkan tersedia pada sistem, serta kebutuhan terkait penyajian informasi untuk mendukung pengambilan keputusan *preventive maintenance*.

3.4.2 Standarisasi *Preventive Maintenance*

Penentuan standar *preventive maintenance* pada penelitian ini didasarkan pada referensi teknis *die casting* serta standar *operational* perusahaan. Menurut North American Die Casting Association (2021), interval *preventive maintenance* maupun umur pakai (*die life*) tidak memiliki nilai baku yang berlaku untuk seluruh *dies* karena dipengaruhi oleh berbagai faktor, seperti material *dies*, jenis paduan (*alloy*), desain produk, kondisi operasi, perlakuan panas (*heat treatment*), serta target kualitas produk. Oleh karena itu, penentuan interval *preventive maintenance* umumnya mengacu pada rekomendasi *manufacturer* maupun pengalaman *operational* masing-masing perusahaan.

Sejalan dengan hal tersebut, GMM Mould Technology (2024) menjelaskan bahwa setiap *dies* memiliki interval *preventive maintenance* yang berbeda sesuai karakteristik *tooling*. Sebagai contoh, beberapa tindakan inspeksi dan *preventive maintenance* direkomendasikan dilakukan pada interval tertentu, seperti 10.000 *shot*, 20.000–30.000 *shot*, maupun interval lainnya, bergantung pada tingkat keausan (*wear*), kompleksitas *dies*, material, serta kondisi proses produksi. Hal ini menunjukkan bahwa tidak terdapat satu standar interval *preventive maintenance* yang bersifat universal untuk seluruh *dies*.

Berdasarkan acuan tersebut, penelitian ini menggunakan standar *preventive maintenance* yang diterapkan pada perusahaan objek penelitian, yaitu 10.000–15.000 *shot*. Nilai tersebut digunakan sebagai parameter dalam sistem *monitoring* untuk menentukan status penggunaan *dies*. Pada *prototype* yang dikembangkan, seluruh *dies* menggunakan standar *preventive maintenance* yang sama sehingga nilai *warning threshold, preventive* target, dan *maximum tolerance* disimpan sebagai parameter konfigurasi pada sistem. Apabila sistem dikembangkan lebih lanjut pada lingkungan industri yang memiliki karakteristik *dies* yang berbeda, parameter tersebut dapat disesuaikan untuk setiap *dies* sesuai rekomendasi *manufacturer* maupun kebijakan perusahaan.

3.4.3 Indikator Kinerja Sistem

Tahap pengujian bertujuan mengevaluasi kesesuaian antara fungsi yang telah dirancang dengan implementasi sistem *monitoring* yang dikembangkan. Pengujian difokuskan pada kemampuan sistem dalam melakukan akuisisi data, pengiriman data, penyimpanan data, visualisasi informasi, serta pengiriman notifikasi selama proses *monitoring* berlangsung. Penentuan indikator kinerja sistem dilakukan berdasarkan fungsi utama sistem *monitoring* sehingga setiap komponen dapat dievaluasi secara terukur.

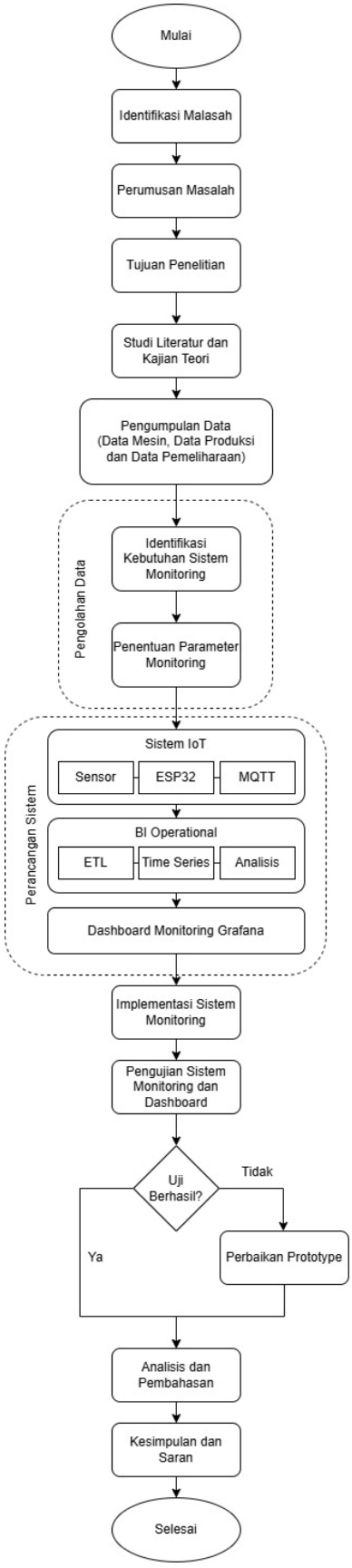
Tabel 3. 1 Indikator Kinerja Sistem

No	Indikator Pengujian	Kriteria Keberhasilan
1	Akuisisi Data	Jumlah *shot* terbaca sesuai kondisi aktual.
2	Pengiriman dan Penyimpanan Data	Data berhasil dikirim ke Node-RED dan tersimpan di InfluxDB.
3	*Dahboard Monitoring*	Informasi *monitoring* ditampilkan sesuai data terbaru.
4	Alert *Preventive Maintenance*	*Dahboard* dan Telegram menampilkan notifikasi sesuai status *dies*.
5	*Monitoring Multi Dies*	Sistem membedakan identitas dan data setiap *dies*.
6	*Latency* Sistem *Monitoring*	Data diterima dan ditampilkan dalam waktu respons yang dapat diterima selama proses *monitoring*.

Berdasarkan indikator tersebut, pengujian dilakukan secara bertahap meliputi proses akuisisi data, pengiriman dan penyimpanan data, visualisasi *dahboard*, pengiriman notifikasi *preventive maintenance*, kemampuan sistem dalam memonitor lebih dari satu *dies*, serta *latency* sistem *monitoring*. Hasil pengujian terhadap masing-masing indikator dibahas pada Bab V.

3.5 Alur Penelitian

Alur penelitian disusun untuk memberikan gambaran mengenai tahapan pelaksanaan penelitian, mulai dari proses perencanaan hingga diperolehnya hasil penelitian. Rangkaian tahapan tersebut divisualisasikan dalam bentuk *flowchart* yang ditampilkan dalam Gambar 3.4 sebagai berikut.



Gambar 3. 4 Alur Penelitian

Berikut merupakan penjelasan dari alur penelitian ini:

1. Identifikasi Masalah

   Tahap identifikasi masalah dilakukan untuk mengenali permasalahan yang terjadi pada proses pemeliharaan *dies* di mesin *die casting*. Fokus utama pada tahap ini adalah kesulitan dalam menentukan waktu pelaksanaan *preventive maintenance* akibat banyaknya jumlah *dies* yang digunakan serta belum tersedianya sistem *monitoring* yang memadai.

2. Perumusan Masalah

   Permasalahan yang telah diidentifikasi kemudian dirumuskan secara lebih spesifik agar penelitian memiliki fokus yang jelas dan dapat dijawab melalui perancangan sistem *monitoring* yang sesuai.

3. Tujuan Penelitian

   Pada tahap ini ditetapkan tujuan yang ingin dicapai melalui penelitian, yaitu merancang sistem *monitoring* pemeliharaan *dies* berbasis IoT yang terintegrasi dengan *Operational* BI sebagai pendukung dalam penentuan waktu *preventive maintenance*.

4. Studi Literatur dan Kajian Teori

   Tahap studi literatur dan kajian teori dilakukan untuk memperoleh landasan teoritis yang mendukung penelitian. Kajian ini mencakup berbagai konsep dan penelitian terdahulu mengenai *IoT*, *BI*, *preventive maintenance*, serta sistem *monitoring* di lingkungan industri.

5. Perancangan Sistem

   Tahap perancangan dilakukan untuk menyusun arsitektur sistem *monitoring* yang terdiri dari:

   a. Sistem IoT, meliputi sensor, ESP32, dan protokol komunikasi MQTT.
   b. *BI Operational*, yang mencakup proses ETL, penyimpanan data *time-series*, dan analisis data *Operational*.
   c. *Dahboard Monitoring* Grafana, sebagai media visualisasi data pemeliharaan secara *real-time*.

6. Implementasi Sistem *Monitoring*

   Tahap implementasi merupakan penerapan rancangan sistem yang telah dibuat. Kegiatan pada tahap ini meliputi pemasangan perangkat IoT, konfigurasi komunikasi

data, integrasi antarperangkat, serta pengembangan *dashboard monitoring* sebagai media visualisasi informasi.

7. Pengujian Sistem

   Pengujian dilakukan untuk memastikan sistem berjalan sesuai dengan rancangan, meliputi keakuratan data, pembaruan data secara *real-time*, dan fungsi visualisasi *dahboard*.

8. Perbaikan *Prototype*

   Setelah dilakukan tahap pengujian sistem, apabila pengujian tidak berhasil, maka akan dilanjutkan ke tahap perbaikan *prototype*, yang dimana perbaikan ini dilakukan untuk mengecek kembali rancangan sistem yang memerlukan perbaikan.

9. Analisis dan Pembahasan

   Hasil pengujian sistem dianalisis dan dibahas untuk menilai kinerja sistem serta kesesuaiannya dengan tujuan penelitian.

10. Kesimpulan dan Saran

    Tahap akhir penelitian berisi penyusunan kesimpulan berdasarkan hasil yang telah diperoleh serta pemberian saran sebagai rekomendasi untuk pengembangan sistem pada penelitian selanjutnya.

BAB IV
PEMBANGUNAN SISTEM

4.1 Pengumpulan Data

Tahap pengumpulan data penelitian ini merupakan implementasi dari fase *Communication* dalam metode *Prototype*. Pada tahap ini dilakukan identifikasi kebutuhan pengguna sebagai dasar dalam menentukan kebutuhan fungsional sistem *monitoring* yang akan dibangun. Pada fase ini dilakukan identifikasi kebutuhan sistem melalui observasi, wawancara, dan studi dokumen untuk memperoleh kebutuhan pengguna sebagai dasar pembangunan *prototype*.

Data yang dikumpulkan digunakan sebagai dasar dalam proses perancangan dan pengembangan sistem *monitoring* pemeliharaan *dies* berbasis IoT dan BI *Operational*. Data yang dikumpulkan difokuskan pada parameter *Operational* yang berkaitan langsung dengan aktivitas pemakaian *dies* selama proses produksi berlangsung. Proses pengumpulan data dilakukan melalui observasi proses produksi, identifikasi kebutuhan *maintenance dies*, pengamatan alur pemakaian *dies*, serta simulasi *monitoring* menggunakan perangkat IoT. Selain itu, dilakukan identifikasi terhadap kebutuhan informasi yang digunakan oleh operator dan bagian *maintenance* dalam memantau kondisi *dies* selama proses produksi berlangsung. Seluruh data yang diperoleh kemudian dimanfaatkan sebagai dasar dalam penyusunan logika *monitoring*, penetapan parameter *preventive maintenance*, pengembangan *dashboard monitoring*, serta pelaksanaan pengujian terhadap sistem yang dikembangkan.

4.1.1 Deskripsi Umum Sistem

Sistem *monitoring* dirancang untuk membantu proses pemantauan pemakaian *dies* pada mesin *die casting* secara lebih terstruktur dibandingkan metode pencatatan manual yang sebelumnya digunakan. Pada proses produksi *die casting*, jumlah *dies* yang digunakan cukup banyak dan terdiri dari berbagai model serta tipe *dies* yang digunakan secara bergantian sesuai kebutuhan produksi. Kondisi tersebut menyebabkan proses pemantauan *shot dies* menjadi lebih sulit dilakukan apabila masih mengandalkan pencatatan manual.

Selain itu, aktivitas produksi yang berlangsung secara kontinu menyebabkan informasi mengenai kondisi pemakaian *dies* sering kali tidak diperoleh secara cepat. Dampaknya, proses *preventive maintenance* berpotensi mengalami keterlambatan karena tidak adanya sistem *monitoring* yang mampu memberikan informasi kondisi *dies* secara aktual selama proses produksi berlangsung. Untuk mengatasi permasalahan tersebut, penelitian ini mengembangkan

sistem *monitoring* berbasis IoT dan *Operational* BI yang mampu melakukan pemantauan *shot dies* secara langsung. Pendekatan IoT digunakan untuk mendukung proses akuisisi dan pengiriman data *Operational* secara otomatis, sedangkan *Operational* BI digunakan untuk mengolah data *monitoring* menjadi informasi *Operational* yang lebih mudah dipahami oleh operator maupun bagian *maintenance*.

Menurut Soori et al. (2023), penerapan *Internet of things* pada lingkungan manufaktur memungkinkan proses *monitoring Operational* dilakukan secara lebih efisien melalui pengumpulan data berbasis sensor dan komunikasi data secara kontinu. Selain itu, Watson & Wixom (2004) menjelaskan bahwa *Operational* BI berfokus pada pemanfaatan data *Operational* berbasis waktu aktual untuk mendukung pengambilan keputusan pada level *Operational*. Sistem *monitoring* yang dikembangkan terdiri dari beberapa komponen utama yang saling terintegrasi, yaitu *limit switch* sebagai pendeteksi *shot dies*, ESP32 DevKitC V4 sebagai perangkat akuisisi data, MQTT sebagai protokol komunikasi data, Node-RED sebagai media pengolahan data *Operational*, InfluxDB sebagai basis data *time-series*, serta Grafana sebagai *dahboard* visualisasi *monitoring*. Proses *monitoring* dimulai ketika *limit switch* mendeteksi aktivitas *shot* pada *prototype dies*. Data hasil pembacaan kemudian diproses oleh ESP32 sebelum dikirim menuju MQTT *Broker* menggunakan mekanisme *publish-subscribe*. Setelah data diterima oleh sistem pusat, Node-RED melakukan proses validasi data, perhitungan jumlah *shot*, pengelompokan data *dies*, serta klasifikasi status *preventive maintenance*. Data hasil pengolahan selanjutnya disimpan pada InfluxDB dan ditampilkan melalui *dahboard* Grafana dalam bentuk visualisasi *monitoring*.

Pemanfaatan MQTT dipilih karena memiliki overhead komunikasi yang relatif rendah dibandingkan protokol komunikasi berbasis HTTP sehingga lebih sesuai untuk sistem *monitoring* IoT yang membutuhkan pengiriman data kontinu dengan penggunaan bandwidth yang lebih efisien (Banks & Gupta, 2015). Sementara itu, penggunaan InfluxDB dipilih karena basis data *time-series* lebih sesuai untuk menangani data *monitoring* berbasis *timestamp* dan *history Operational* mesin (Garcia et al., 2022). *Dahboard monitoring* yang dikembangkan tidak hanya menampilkan jumlah *shot dies*, tetapi juga menampilkan status *Operational dies*, persentase pemakaian *dies*, *remaining preventive maintenance*, *history maintenance*, serta *alert* kondisi *dies* yang mendekati batas *preventive maintenance*. Informasi tersebut diharapkan dapat membantu operator maupun bagian *maintenance* dalam memantau kondisi *dies* secara lebih cepat dan terstruktur. Dengan pendekatan tersebut, sistem yang dikembangkan pada penelitian

ini tidak hanya berfungsi sebagai *prototype monitoring* berbasis IoT, tetapi juga sebagai implementasi konsep *Operational Business Intelligence* pada proses *preventive maintenance dies* di lingkungan manufaktur.

4.1.2 Identifikasi Kebutuhan Pengguna

Tahap identifikasi kebutuhan pengguna merupakan implementasi fase pada metode *Prototype* yang bertujuan untuk memperoleh kebutuhan sistem sebelum dilakukan sebelum proses perancangan dilakukan. Identifikasi kebutuhan pengguna dilakukan melalui observasi proses *preventive maintenance*, wawancara dengan supervisor maintenance dan operator produksi, serta studi dokumen perusahaan. Tahap ini bertujuan untuk memperoleh kebutuhan sistem yang akan menjadi dasar dalam perancangan *prototype*.

Pada penelitian ini identifikasi kebutuhan dilakukan melalui observasi proses *preventive maintenance* dan diskusi dengan manager dan *foreman* maintenance, serta studi dokumen yang berkaitan dengan pengelolaan *dies*. Berdasarkan hasil diskusi dengan Manager HPDCM, diperoleh beberapa kebutuhan utama sistem, yaitu kemampuan menampilkan jumlah *shot* setiap *dies* secara otomatis, memberikan informasi kondisi *dies* yang mendekati batas *preventive maintenance*, menyajikan *dahboard monitoring* yang mudah dipahami, menampilkan riwayat penggunaan *dies*, mendukung penyusunan perencanaan *preventive maintenance* secara lebih fleksibel, serta memungkinkan proses pemantauan dilakukan secara *real-time* tanpa pencatatan manual. Berdasarkan hasil identifikasi kebutuhan pengguna, sistem yang dikembangkan harus mampu memenuhi kebutuhan fungsional maupun nonfungsional.

Tabel 4. 1 Hasil Kebutuhan Sistem

No	Kebutuhan Sistem
1	*Monitoring* jumlah *shot* otomatis
2	*Monitoring* lebih dari satu *dies*
3	*Dahboard real-time*
4	Status NORMAL-*WARNING-DUE-CRITICAL*
5	Notifikasi Telegram
6	*History maintenance*
7	Barcode identitas *dies*

4.1.3 Data Sistem

Data yang digunakan pada penelitian ini merupakan data *Operational* yang berkaitan langsung dengan aktivitas pemakaian *dies* selama proses produksi berlangsung. Pemilihan parameter data

dilakukan berdasarkan kebutuhan *monitoring preventive maintenance* serta kebutuhan visualisasi informasi pada *dahboard monitoring*.

Pada sistem *die casting*, jumlah *shot* menjadi salah satu parameter penting karena berhubungan langsung dengan tingkat pemakaian *dies*. Semakin tinggi jumlah *shot* yang digunakan, maka tingkat keausan *dies* juga akan meningkat. Oleh karena itu, pemantauan jumlah *shot* secara kontinu diperlukan untuk membantu proses penentuan *preventive maintenance* sebelum terjadi kerusakan yang dapat mengganggu proses produksi. Menurut Mobley et al. (2008), pemanfaatan data *Operational* secara kontinu pada sistem *maintenance* dapat membantu perusahaan dalam mendeteksi kondisi peralatan sebelum terjadi kegagalan yang menyebabkan *downtime* produksi. Selain itu, *monitoring* berbasis data aktual dinilai lebih efektif dibandingkan pendekatan *maintenance* yang hanya mengandalkan jadwal periodik tetap.

Penentuan batas *preventive maintenance* pada penelitian ini didasarkan pada hasil wawancara dengan pihak yang terlibat dalam pengelolaan *dies*. Berdasarkan informasi yang diperoleh, *preventive maintenance dies* pada kondisi aktual dilakukan pada *cycle shot* 12.000, sedangkan batas maksimum penggunaan *dies* berada pada *cycle shot* 15.000. Karena penelitian ini menggunakan *prototype* untuk kebutuhan pengujian sistem, nilai tersebut disesuaikan menjadi 1.200 *cycle shot* sebagai batas *preventive maintenance* dan 1.500 *cycle shot* sebagai batas maksimum penggunaan *dies*. Penyesuaian dilakukan agar simulasi pengujian dapat dilaksanakan dalam waktu yang lebih efisien tanpa mengubah prinsip kerja sistem. Dengan demikian, rasio antara batas *preventive maintenance* dan batas maksimum penggunaan *dies* pada *prototype* tetap mengikuti kondisi aktual yang digunakan di lapangan.

Data pada sistem *monitoring* ini diperoleh melalui proses pembacaan aktivitas *shot* menggunakan *limit switch* yang terhubung dengan ESP32 DevKitC V4. Setiap aktivitas *shot* akan menghasilkan data baru yang dikirimkan menuju sistem *monitoring* menggunakan protokol MQTT. Berikut merupakan parameter data yang digunakan untuk kebutuhan sistem *monitoring* pada Tabel 4.2.

Tabel 4. 2 Data Kebutuhan Sistem

No	Parameter	Deskripsi	Sumber	Tujuan Penggunaan
1	*Cycle shot*	Jumlah siklus pemakaian *dies*	ESP32 dan Sistem IoT	*Monitoring* pemakaian *dies*
2	*Lifetime Shot*	Total *Shot dies* sepanjang masa penggunaan	Sistem IoT	Histroi pemakaian *dies*

No	Parameter	Deskripsi	Sumber	Tujuan Penggunaan
3	Identitas *dies* (*dies* ID)	Identitas *dies* yang sedang digunakan	Sistem IoT dan Sistem *Monitoring*	Tracking *dies* aktif
4	*Usage percentage*	Persentase pemakaian *dies* terhadap *limit* PM	Sistem *Monitoring*	Penentuan status *preventive maintenance*
5	*Remaining Shot*	Sisa *shot* menuju batas *preventive maintenance*	Sistem *Monitoring*	*Monitoring* kondisi *Operational dies*
6	*Limit* PM	Batas maksimal *shot preventive maintenance*	Data Observasi	Penentuan status PM
7	*Timestamp*	Waktu pencatatan data	Sistem *Monitoring*	Analisis *history*s dan tren
8	Status *Operational*	Status Kondisi *Dies*	Sistem *Monitoring*	*Monitoring dies*
9	*Maintenance Count*	Jumlah *preventive maintenance*	Sistem *Monitoring*	*History Maintenance*

Parameter *cycle shot* digunakan sebagai parameter utama dalam proses *monitoring* karena menunjukkan jumlah pemakaian *dies* sejak *preventive maintenance* terakhir dilakukan. Parameter ini menjadi dasar dalam penentuan status *preventive maintenance* karena lebih merepresentasikan kondisi aktual pemakaian *dies* dibandingkan total *lifetime shot*.

Sementara itu, *lifetime shot* digunakan untuk mencatat total penggunaan *dies* sepanjang masa *Operational*nya. Data tersebut digunakan sebagai *history* pemakaian jangka panjang dan dapat dimanfaatkan untuk evaluasi umur *dies*.

Parameter *usage percentage* digunakan untuk menunjukkan tingkat pemakaian *dies* terhadap batas *preventive maintenance* yang telah ditentukan. Pendekatan perbandingan antara nilai aktual pemakaian dan batas *maintenance* umum digunakan pada sistem *monitoring maintenance* untuk membantu proses klasifikasi kondisi peralatan secara *Operational* (Mobley et al., 2008). Persentase tersebut digunakan untuk menentukan kategori status *preventive maintenance*, yaitu NORMAL, *WARNING*, *DUE*, dan *CRITICAL*. Perhitungan *usage percentage* dilakukan menggunakan persamaan berikut:

$$Usage\ Percentage = \frac{Cycle\ Shot}{PM\ Limit} \times 100\%$$

Selain itu, sistem juga menghitung *remaining shot* untuk mengetahui sisa kapasitas pemakaian *dies* sebelum mencapai batas *preventive maintenance*. Informasi *remaining shot*

digunakan sebagai indikator awal untuk membantu operator dalam mempersiapkan jadwal *maintenance dies*. *Remaining shot* dihitung menggunakan persamaan berikut:

$$Remaining\ Shot = PM\ Limit - Cycle\ Shot$$

Penggunaan parameter *timestamp* pada sistem *monitoring* bertujuan untuk mencatat waktu setiap aktivitas *monitoring* sehingga data dapat disimpan sebagai data *time-series*. Data *time-series* memiliki peran penting pada sistem *monitoring modern* karena memungkinkan proses analisis *history*s, *trend monitoring*, dan pelacakan perubahan kondisi sistem secara berkelanjutan (Kleppmann, 2021).

4.2 Perancangan Sistem

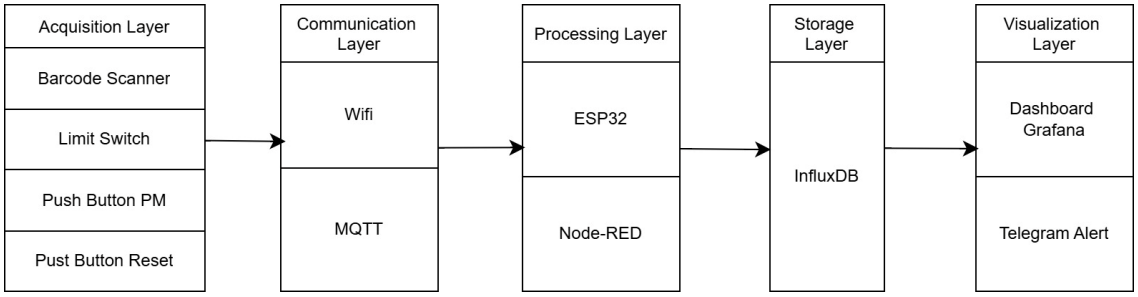
Berdasarkan hasil identifikasi kebutuhan pada tahap *Communication*, selanjutnya dilakukan tahap *Quick Plan* sebagai dasar perancangan *prototype*. Pada tahap ini dilakukan penentuan arsitektur sistem, pemilihan perangkat keras, perangkat lunak, serta mekanisme komunikasi data yang akan digunakan. Hal ini digunakan dalam merealisasikan rancangan sistem untuk *monitoring* pemakaian *dies* berbasis IoT dan *Operational* BI yang telah disusun pada tahap perancangan. Tahap ini bertujuan untuk mengintegrasikan perangkat akuisisi data, komunikasi data, pengolahan data *Operational*, penyimpanan data, serta visualisasi *monitoring* ke dalam satu sistem yang saling terhubung. Menurut Elkateb et al. (2024), implementasi sistem *monitoring* berbasis IoT pada lingkungan industri memerlukan integrasi antara perangkat akuisisi data, komunikasi data, pengolahan data, dan visualisasi informasi agar proses *monitoring* dapat berjalan secara berkelanjutan dan mendukung kegiatan pemeliharaan secara lebih proaktif.

Pada penelitian ini, sistem *monitoring* dibangun menggunakan beberapa komponen utama yang terdiri dari ESP32 DevKitC V4 sebagai perangkat akuisisi data, *barcode scanner* sebagai identifikasi *dies*, *limit switch* sebagai pendeteksi aktivitas *shot*, MQTT sebagai protokol komunikasi data, Node-RED sebagai media pengolahan data *Operational*, InfluxDB sebagai basis data *time-series*, serta Grafana sebagai *dahboard* visualisasi *monitoring*. Pembangunan sistem dilakukan secara bertahap mulai dari integrasi perangkat keras, konfigurasi komunikasi data, penyusunan alur pengolahan data, hingga implementasi *dahboard monitoring*. Setiap komponen dirancang agar mampu bekerja secara terintegrasi sehingga data pemakaian *dies* dapat diperoleh, diolah, disimpan, dan ditampilkan secara otomatis. Melalui pembangunan sistem ini diharapkan proses *monitoring* pemakaian *dies* dapat dilakukan secara lebih

terstruktur dibandingkan metode pencatatan manual, sehingga informasi yang dihasilkan dapat digunakan untuk mendukung pelaksanaan *preventive maintenance*. Tahapan pembangunan sistem yang dilakukan pada penelitian ini meliputi struktur perancangan sistem, *wiring* perangkat *monitoring*, implementasi perangkat IoT, serta perancangan arsitektur dan pengolahan data *Operational*.

4.2.1 Struktur Perancangan Sistem

Struktur sistem *monitoring* pada penelitian ini dirancang menggunakan pendekatan IoT yang mengintegrasikan perangkat akuisisi data, komunikasi data, pengolahan data *Operational*, penyimpanan data, serta visualisasi *monitoring* dalam satu arsitektur sistem yang saling terhubung. Pendekatan ini digunakan agar proses *monitoring* pemakaian *dies* dapat dilakukan secara lebih terstruktur dan mendukung penyajian informasi *Operational* secara berkelanjutan dan ditampilkan dalam bentuk informasi yang mendukung kegiatan *preventive maintenance*. Berikut struktur perancangan sistem *monitoring* yang dikembangkan pada penelitian ini dapat dilihat melalui gambar 4.1



Gambar 4. 1 Struktur sistem

Menurut Xu et al. (2018), arsitektur sistem IoT umumnya terdiri dari beberapa lapisan utama, yaitu *acquisition layer, communication layer, processing layer, storage layer,* dan *visualization layer*. Pembagian lapisan tersebut bertujuan untuk mempermudah integrasi antar komponen dan meningkatkan fleksibilitas pengembangan sistem. Pada penelitian ini, *Acquisition layer* berfungsi untuk memperoleh data aktivitas *shot dies* dari perangkat *monitoring*. Data yang diperoleh kemudian dikirim melalui *communication layer* menggunakan protokol MQTT sebagai media komunikasi antara ESP32 dan Node-RED. Selanjutnya, *processing layer* melakukan pengolahan data *monitoring* yang meliputi validasi data, pengelolaan parameter *monitoring*, serta pembentukan informasi yang dibutuhkan oleh sistem.

Data hasil pengolahan kemudian disimpan pada *storage layer* sebagai data hist*o*ris berbasis waktu yang dapat digunakan untuk proses *monitoring* maupun analisis riwayat

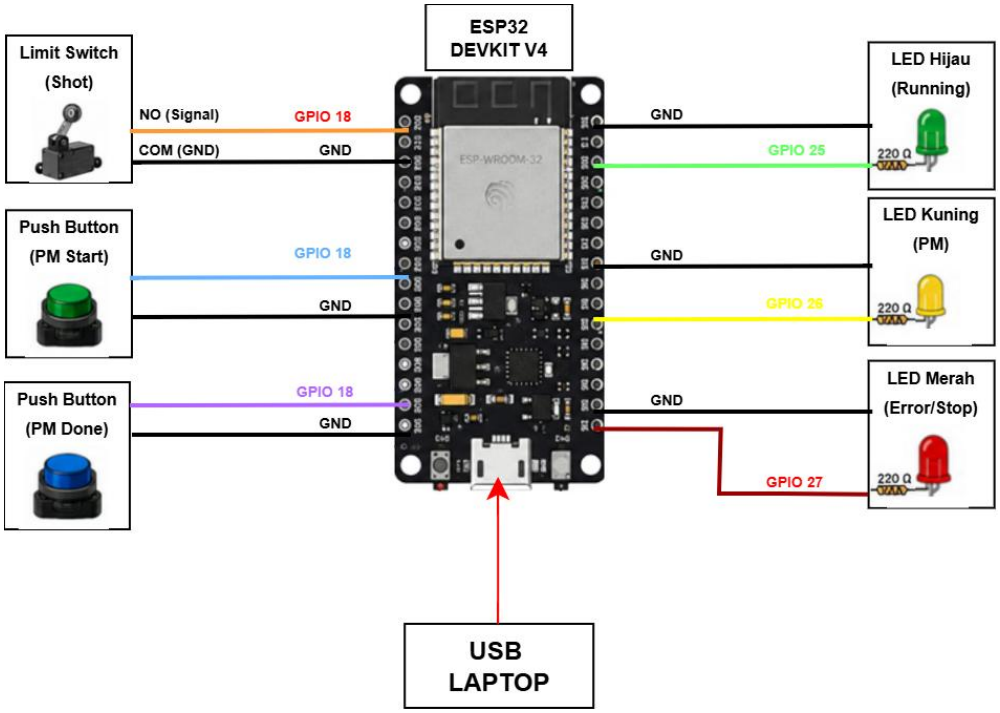
penggunaan *dies*. Pada tahap akhir, *visualization layer* menyajikan informasi *monitoring* dalam bentuk *dahboard* sehingga informasi seperti penggunaan *dies*, status *dies*, dan kondisi *preventive maintenance* dapat dipahami dengan lebih mudah oleh operator, *supervisor*, maupun bagian *maintenance*.

Pembagian fungsi ke dalam beberapa lapisan memungkinkan sistem *monitoring* dikembangkan secara lebih modular. Selain mempermudah proses integrasi, pendekatan ini juga mendukung pengembangan sistem pada implementasi berikutnya apabila jumlah *dies*, parameter *monitoring*, maupun kebutuhan pengguna mengalami peningkatan.

4.2.2 *Wiring* Rancangan Sistem

Wiring sistem merupakan proses perancangan dan penghubungan antar komponen perangkat keras dalam suatu sistem elektronik atau sistem *monitoring* agar setiap perangkat dapat saling terintegrasi dan bekerja sesuai dengan fungsi yang dirancang. Pada sistem *monitoring* berbasis IoT, *wiring* tidak hanya berfungsi sebagai jalur koneksi perangkat, tetapi juga berpengaruh terhadap kestabilan pembacaan data, komunikasi sistem, dan kontinuitas *Operational monitoring*.

Wiring sistem *monitoring* dirancang untuk menghubungkan perangkat *input*, perangkat IoT, serta media komunikasi data agar seluruh komponen sistem dapat bekerja secara terintegrasi selama proses *monitoring* berlangsung. Perancangan *wiring* dilakukan dengan mempertimbangkan kemudahan implementasi *prototype*, kestabilan pembacaan data, serta fleksibilitas pengembangan sistem pada tahap implementasi berikutnya. Berikut konfigurasi *wiring* pada sistem *monitoring* yang dikembangkan melalui Gambar 4.2.



Gambar 4. 2 *Wiring* ESP32

Menurut Espressif Systems (2021), konfigurasi koneksi perangkat yang sesuai berpengaruh terhadap kestabilan pembacaan sinyal dan komunikasi data pada sistem berbasis mikrokontroler. Pada penelitian ini, *limit switch* digunakan sebagai sensor utama untuk mendeteksi aktivitas *shot* pada *prototype dies*. Penggunaan *limit switch* dipilih karena lebih mudah diimplementasikan pada *prototype* mekanisme *dies* yang menggunakan sistem gerak simulasi.

Selain itu, *limit switch* memiliki mekanisme pembacaan kontak fisik yang lebih sederhana sehingga mempermudah proses pengujian sistem pada tahap pengembangan *prototype*. Pendekatan ini digunakan agar proses simulasi *shot dies* dapat dilakukan secara stabil tanpa harus melakukan integrasi langsung dengan mesin produksi aktual. *Wiring* sistem ini dilakukan dengan *limit switch* yang dihubungkan ke pin *input digital* ESP32 DevKitC V4 sebagai sumber data aktivitas *shot dies*. Ketika *limit switch* aktif, ESP32 akan membaca perubahan status *input* dan mengolahnya sebagai data *shot*. Selain itu, *push button* juga dihubungkan ke ESP32 untuk mendukung fungsi reset *preventive maintenance* dan pergantian *dies* aktif pada sistem *monitoring*. Berikut merupakan konfigurasi pin ESP32 yang digunakan:

Tabel 4. 3 Pin ESP32

Komponen	GPIO
Limit switch (*Shot*)	18
PM *Start*	19

Komponen	GPIO
PM *Done*	23
LED *Running*	25
LED *PM*	26

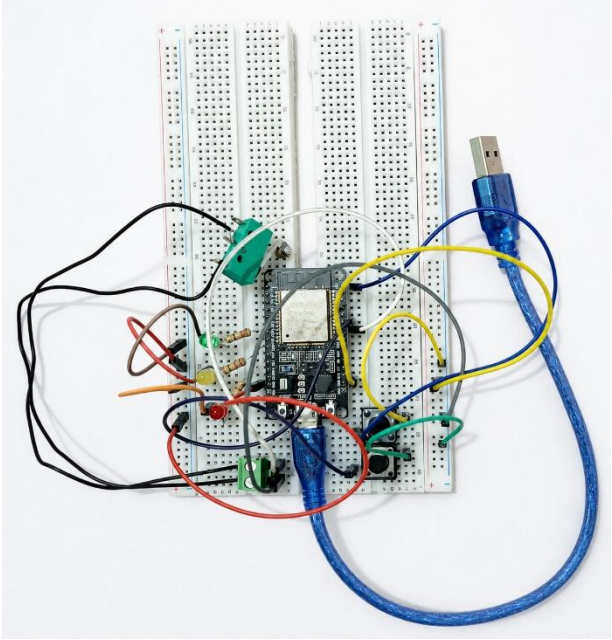
Seperti tabel 4.3 bahwa ESP32 selanjutnya terhubung dengan jaringan WiFi yang disambungkan melalui GPIO (*general purpose input output*) untuk mengirimkan data *monitoring* menuju sistem pengolahan data menggunakan protokol MQTT. Selama tahap pengembangan *prototype*, sumber daya sistem diperoleh melalui koneksi USB laptop dengan tegangan 5V DC. Perancangan *wiring* dilakukan dengan menyusun koneksi antar perangkat secara terstruktur sehingga proses instalasi, pengujian, dan pemeliharaan sistem dapat dilakukan dengan lebih mudah. Susunan koneksi tersebut juga memudahkan proses identifikasi apabila terjadi kesalahan pada salah satu komponen selama pengoperasian sistem *monitoring*.

4.2.3 Implementasi Perangkat IoT (ESP32)

Implementasi perangkat IoT pada penelitian ini dilakukan menggunakan ESP32 DevKitC V4 sebagai perangkat utama untuk membaca data aktivitas *dies* dan mengirimkan data *monitoring* menuju sistem pengolahan data. ESP32 berperan sebagai perangkat akuisisi data yang menghubungkan aktivitas *Operational dies* dengan platform *monitoring* yang digunakan.

ESP32 merupakan mikrokontroler yang mendukung konektivitas WiFi dan *Bluetooth* sehingga perangkat ini banyak digunakan pada pengembangan sistem IoT yang membutuhkan komunikasi data secara nirkabel (Systems, 2021). Pada sistem ini, ESP32 menerima *input* dari *limit switch* yang dipasang pada mekanisme pergerakan *prototype dies*. Setiap kali terjadi aktivitas *shot*, *limit switch* akan menghasilkan sinyal *digital* yang dibaca oleh ESP32 sebagai penambahan jumlah *shot dies*.

Data yang diperoleh kemudian diolah menjadi beberapa parameter *monitoring*, yaitu *cycle shot*, *lifetime shot*, *usage percentage*, *remaining shot*, dan status *Operational dies*. Pengolahan awal dilakukan langsung pada ESP32 untuk mengurangi beban pemrosesan pada sistem *monitoring*. Setelah proses pengolahan selesai, data *monitoring* dikirimkan melalui jaringan WiFi menggunakan protokol MQTT menuju MQTT *Broker*. Mekanisme ini memungkinkan data *monitoring* diperbarui secara berkelanjutan selama sistem beroperasi. Alur implementasi perangkat ESP32 pada *prototype* sistem *monitoring* melalui Gambar 4.3 sebagai berikut.



Gambar 4. 3 Implementasi *Hardware*

Selain melakukan pembacaan data *shot*, ESP32 juga digunakan untuk mengelola fungsi reset *preventive maintenance* yang dilakukan secara manual oleh operator. Ketika tombol reset diaktifkan setelah kegiatan *preventive maintenance* dilakukan, nilai *cycle shot* akan kembali ke nol, sedangkan data *lifetime shot* tetap disimpan sebagai *history* total pemakaian *dies*. Pemisahan antara *cycle shot* dan *lifetime shot* dilakukan agar sistem dapat membedakan jumlah pemakaian *dies* sejak *preventive maintenance* terakhir dengan total pemakaian *dies* selama masa *Operational*nya.

Selanjutnya, data hasil pembacaan *limit switch* selanjutnya dikirim menuju sistem *monitoring* menggunakan protokol MQTT melalui jaringan WiFi. Protokol tersebut digunakan sebagai media komunikasi antara ESP32 dan sistem pengolahan data. Pada penelitian ini, ESP32 berperan sebagai publisher yang mengirimkan data *monitoring* secara kontinu ke *topic* MQTT tertentu agar dapat diterima dan diproses oleh Node-RED. Hasil pembacaan dan pengiriman data *monitoring* oleh ESP32 dapat dilihat melalui serial monitor yang dicantumkan Gambar 4.4.

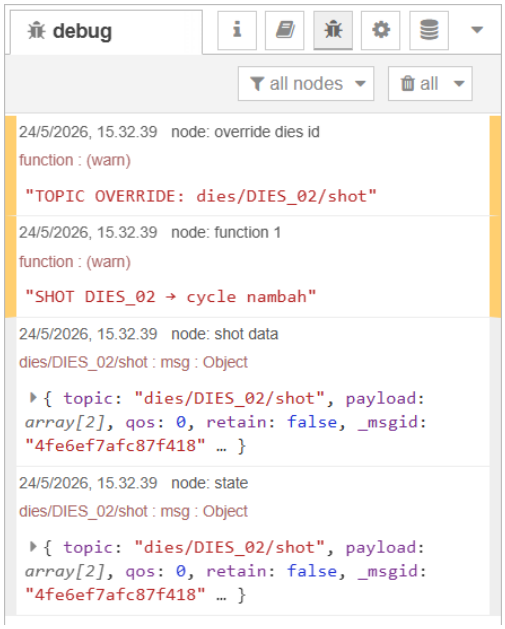


Gambar 4. 4 Implementasi MQTT

Penggunaan MQTT pada implementasi ESP32 dipilih karena memiliki overhead komunikasi yang rendah sehingga lebih sesuai untuk sistem *monitoring* berbasis IoT yang membutuhkan proses pengiriman data secara kontinu dan ringan. Selain itu, mekanisme *publish-subscribe* memungkinkan distribusi data dilakukan secara lebih fleksibel tanpa membebani perangkat akuisisi data. Berikut merupakan proses publish data *monitoring* menuju MQTT *Broker* melalui Gambar 4.5



Gambar 4. 5 MQTT *Running*



Gambar 4. 6 Implementasi Node-RED

Pada tahap ini, ESP32 dikonfigurasi menggunakan Arduino IDE dan dihubungkan dengan jaringan WiFi lokal untuk mendukung komunikasi data *monitoring*. Setelah perangkat berhasil terhubung ke jaringan, ESP32 akan melakukan proses publish data secara otomatis setiap kali terjadi perubahan data *monitoring*. Selain berfungsi sebagai pengirim data, ESP32 juga digunakan untuk mengelola logika dasar sistem seperti pembacaan *trigger shot*, perhitungan *cycle shot*, pengelolaan *lifetime shot*, *reset preventive maintenance*, serta pengelolaan status *operational dies*. Pengolahan data awal pada ESP32 dilakukan agar sebagian proses *monitoring* dapat diselesaikan langsung pada perangkat sebelum data dikirim ke sistem pusat. Pendekatan ini membantu mengurangi beban pemrosesan pada sistem pusat serta menjaga proses *monitoring* tetap berjalan dengan baik ketika jumlah data yang diproses meningkat. Menurut Shi et al. (2016), konsep edge processing pada perangkat IoT memungkinkan sebagian pengolahan data dilakukan langsung pada perangkat akuisisi sehingga dapat meningkatkan efisiensi sistem *monitoring* dan mengurangi ketergantungan terhadap proses pengolahan terpusat.

Implementasi ESP32 tidak hanya digunakan sebagai perangkat pembaca sensor, tetapi juga menjadi bagian dari sistem untuk mendukung proses akuisisi dan pengelolaan data secara berkelanjutan. Integrasi tersebut memungkinkan data *monitoring* dikirim dan diolah secara otomatis sebelum ditampilkan pada *dahboard Business Intelligence*. Dengan pendekatan ini, sistem yang dirancang juga memiliki peluang untuk dikembangkan lebih lanjut guna

mendukung *monitoring multi-dies* maupun implementasi pada lingkungan manufaktur yang lebih luas.

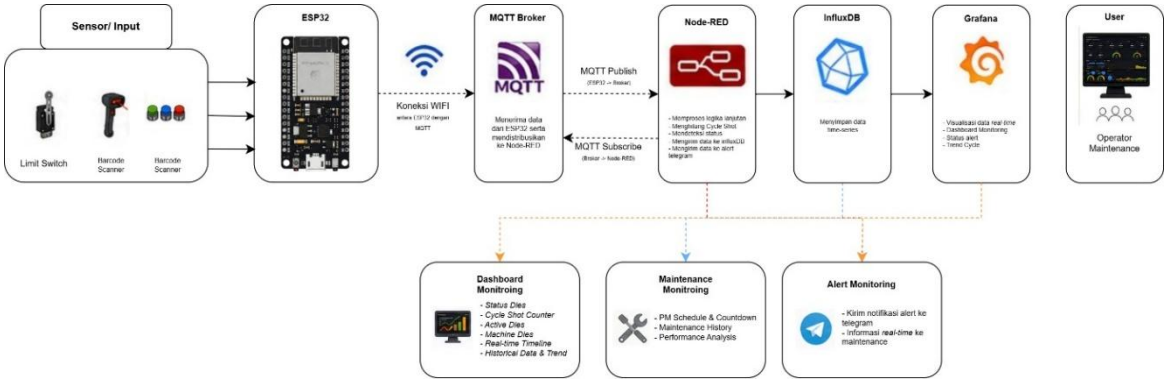
4.3 Arsitektur dan Pengolahan Data *Operational*

Selanjutnya dilakukan tahap implementasi fase *Modeling Quick Design* yang berfokus pada penyusunan rancangan arsitektur sistem, alur komunikasi data, serta mekanisme pengolahan data sebelum *prototype* perangkat keras maupun perangkat lunak diimplementasikan. Arsitektur dan pengolahan data *Operational* pada penelitian ini dirancang untuk mengelola data pemakaian *dies* mulai dari proses pengambilan data hingga penyajian informasi *monitoring*. Integrasi tersebut dilakukan agar data yang diperoleh dari aktivitas *dies* dapat diolah menjadi informasi yang mendukung pelaksanaan *preventive maintenance* secara lebih terstruktur.

Pada penelitian ini, data aktivitas *shot dies* diperoleh melalui perangkat IoT yang terhubung dengan sistem *monitoring*. Data yang diterima kemudian diproses untuk menghasilkan parameter *monitoring* seperti *cycle shot*, *lifetime shot*, *usage percentage*, *remaining shot*, dan status *preventive maintenance*. Selanjutnya, data disimpan sebagai data *history*s dan ditampilkan melalui *dahboard monitoring* secara *near real-time*. Secara umum, proses pengolahan data pada sistem *monitoring* terdiri atas 3 tahapan utama, yaitu *input*, *process*, serta *output*. Ketiga tahapan tersebut saling terhubung untuk menghasilkan informasi *monitoring* yang dapat digunakan oleh operator maupun bagian *maintenance*.

4.3.1 Arsitektur Sistem *Monitoring*

Arsitektur sistem *monitoring* menggambarkan hubungan antar komponen yang terlibat dalam proses pemantauan pemakaian *dies*. Arsitektur ini dirancang untuk memastikan data yang berasal dari aktivitas *dies* dapat diterima, diolah, disimpan, dan ditampilkan secara berkelanjutan pada *dahboard monitoring*.



Gambar 4. 7 Arsitektur Sistem

Berdasarkan Gambar 4.7, proses *monitoring* dimulai ketika aktivitas *shot dies* terdeteksi oleh *limit switch* yang terhubung dengan ESP32 DevKitC V4. Data hasil pembacaan kemudian dikirim menuju sistem *monitoring* melalui jaringan komunikasi dan diteruskan ke sistem pengolahan data. Setelah data diterima, sistem melakukan pembaruan parameter *monitoring* yang terdiri dari *cycle shot*, *lifetime shot*, *usage percentage*, *remaining shot*, serta status *preventive maintenance*. Data yang telah diproses kemudian disimpan sebagai data *history*s dan digunakan sebagai sumber informasi pada *dahboard monitoring*. Data hasil pembacaan kemudian dikirim menuju MQTT *Broker* menggunakan protokol MQTT melalui jaringan WiFi. Penggunaan MQTT pada arsitektur *monitoring* dipilih karena mendukung komunikasi ringan dengan mekanisme *publish-subscribe* yang lebih sesuai untuk sistem *monitoring* berbasis data kontinu.

Menurut Banks & Gupta (2015), MQTT dirancang untuk mendukung komunikasi *machine-to-machine* dan *Internet of things* dengan penggunaan bandwidth yang relatif rendah serta mendukung proses distribusi data secara efisien. Setelah data diterima oleh MQTT *Broker*, proses pengolahan data dilakukan menggunakan Node-RED sebagai processing *layer* pada sistem *monitoring*. Penggunaan Node-RED memungkinkan proses pengolahan data dilakukan secara visual melalui *flow* programming sehingga mempermudah integrasi antar layanan *monitoring* dan pengembangan sistem secara terstruktur.

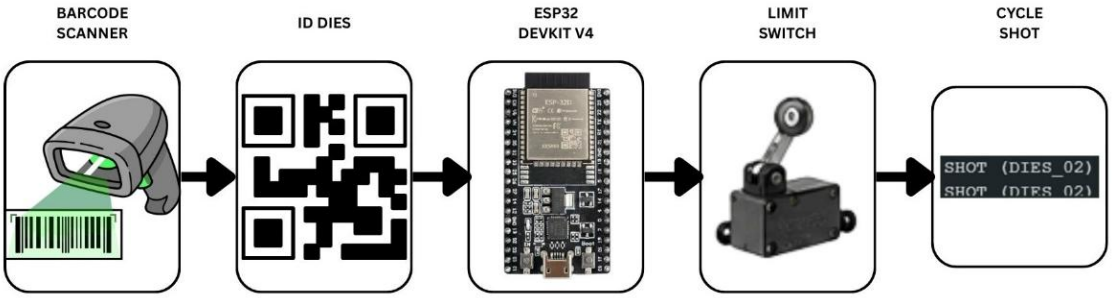
Selain itu, Node-RED juga berperan sebagai media implementasi proses ETL (*Extract*, *Transform*, *Load*) pada sistem *monitoring*. Proses *extract* dilakukan dengan mengambil data *monitoring* dari MQTT *Broker*, proses *transform* dilakukan melalui pengolahan parameter *monitoring* dan klasifikasi status *dies*, sedangkan proses *load* dilakukan dengan mengirimkan data hasil pengolahan menuju basis data *monitoring*. Menurut Chaudhuri et al. (2011), proses ETL memiliki peran penting dalam arsitektur *Business Intelligence* karena digunakan untuk

mengubah data *Operational* menjadi informasi yang lebih terstruktur dan siap digunakan dalam proses analisis maupun visualisasi data. *Storage layer* menggunakan InfluxDB sebagai basis data *monitoring* yang berfungsi untuk menyimpan data *Operational dies* secara *history*s. Data yang tersimpan selanjutnya digunakan sebagai sumber informasi pada *dahboard monitoring*. Arsitektur basis data *time-series* dinilai lebih sesuai untuk *monitoring Operational* karena mampu menangani data *history*s *monitoring* dengan performa query yang lebih cepat dibandingkan basis data relasional konvensional (Garcia et al., 2022)

Selanjutnya, Grafana digunakan sebagai visualization *layer* untuk menampilkan data *monitoring* dalam bentuk *dahboard Operational*. *Dahboard monitoring* digunakan untuk menyajikan informasi pemakaian *dies* secara lebih ringkas dan mudah dipahami oleh operator maupun bagian *maintenance*. Informasi yang ditampilkan pada *dahboard* terdapat jumlah *shot dies*, *usage percentage*, *remaining shot*, status *Operational dies*, *history maintenance*, dan *alert preventive maintenance* Melalui arsitektur tersebut, seluruh komponen dapat bekerja secara terintegrasi sehingga informasi pemakaian *dies* dapat diperoleh secara lebih cepat dibandingkan pencatatan manual. Selain itu, data *history*s yang tersimpan memungkinkan proses *monitoring* dilakukan tidak hanya berdasarkan kondisi aktual, tetapi juga berdasarkan riwayat penggunaan *dies* sebelumnya.

4.3.2 Akuisisi Data (*Input*)

Tahap akuisisi data merupakan proses awal dalam sistem *monitoring* yang berfungsi untuk memperoleh data *Operational* dari aktivitas pemakaian *dies* selama proses produksi berlangsung. Data yang diperoleh pada tahap ini selanjutnya digunakan sebagai dasar dalam proses *monitoring preventive maintenance* dan pengolahan informasi *Operational* pada sistem BI. Menurut Lee et al. (2020), proses akuisisi data pada sistem IoT memiliki peran penting karena menentukan kualitas data *monitoring* yang akan digunakan dalam proses analisis *Operational* dan *maintenance*. Berikut merupakan alur akuisisi data pada sistem *monitoring*.

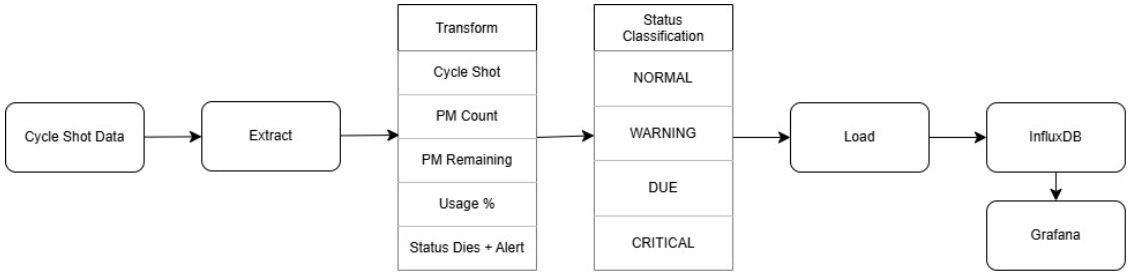


Gambar 4. 8 Alur akuisisi data

Berdasarkan Gambar 4.8, data diperoleh dari dua sumber utama, yaitu *barcode scanner* dan *limit switch*. *Barcode scanner* digunakan untuk mengidentifikasi *dies* yang sedang digunakan pada proses produksi, sedangkan *limit switch* digunakan untuk mendeteksi aktivitas *shot dies* selama sistem beroperasi. Ketika *barcode dies* dipindai, sistem akan membaca identitas *dies* dan menjadikannya sebagai referensi *monitoring*. Setelah *dies* aktif terdaftar pada sistem, setiap aktivitas *shot* yang terdeteksi oleh *limit switch* akan dicatat sebagai penambahan jumlah pemakaian *dies*. Data yang diperoleh pada tahap akuisisi terdiri dari identitas *dies* (*dies* ID), *cycle shot*, *lifetime shot*, *timestamp*, serta status *Operational dies*. Data tersebut selanjutnya digunakan sebagai *input* utama dalam proses pengolahan data *monitoring*.

4.3.3 Pengiriman dan Pengolahan Data (*Process*)

Setelah data diperoleh dari tahap akuisisi, sistem melakukan proses pengiriman dan pengolahan data untuk menghasilkan informasi *monitoring* yang dapat digunakan dalam kegiatan *preventive maintenance*. Tahap ini berfungsi untuk mengubah data aktivitas *dies* menjadi parameter *monitoring* yang lebih mudah dipahami oleh pengguna. Berikut alur pengiriman dan pengolahan data pada sistem *monitoring* yang dikembangkan disajikan dalam Gambar 4.9.

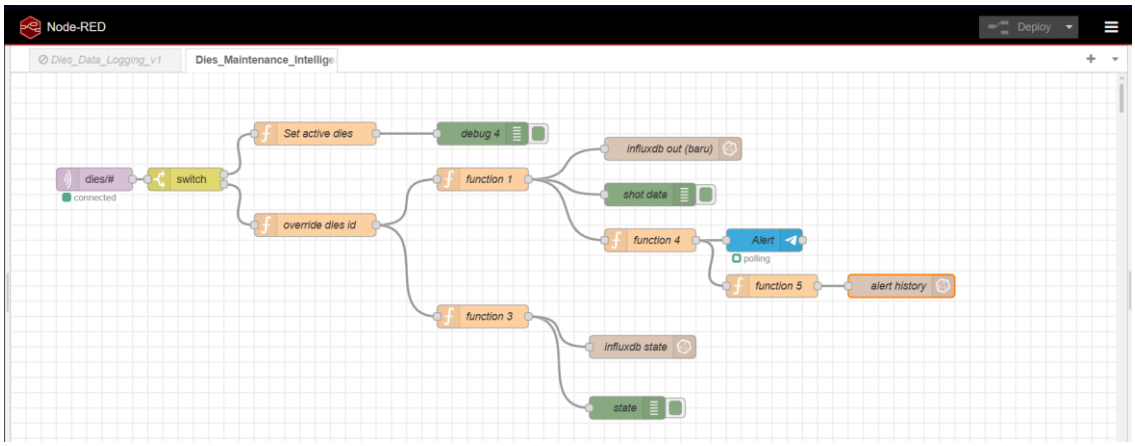


Gambar 4. 9 *Flow* Pengolahan Data

Pada sistem *monitoring* yang dikembangkan, data hasil pembacaan *shot dies* dikirim oleh ESP32 menuju MQTT *Broker* menggunakan protokol MQTT melalui jaringan WiFi.

Proses pengiriman dilakukan menggunakan mekanisme *publish-subscribe* sehingga data *monitoring* dapat diteruskan menuju sistem pengolahan data secara berkelanjutan. Penggunaan MQTT dipilih karena memiliki kebutuhan *bandwidth* yang rendah dan sesuai kebutuhan sistem *monitoring* berbasis IoT yang membutuhkan pengiriman data secara kontinu. Menurut Banks & Gupta (2015), MQTT dirancang untuk mendukung komunikasi *machine-to-machine* dan *Internet of things* dengan mekanisme komunikasi yang ringan, efisien, dan mudah diimplementasikan.

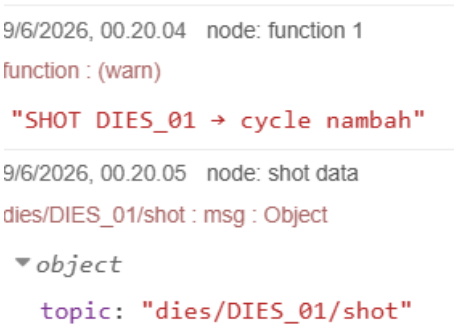
Setelah data diterima oleh MQTT *Broker*, proses pengolahan data dilakukan menggunakan Node-RED sebagai media integrasi dan pengolahan data pada sistem *monitoring*. Node-RED digunakan untuk menerima data *monitoring*, melakukan validasi data, mengelola parameter *monitoring*, serta meneruskan data menuju basis data *monitoring*.



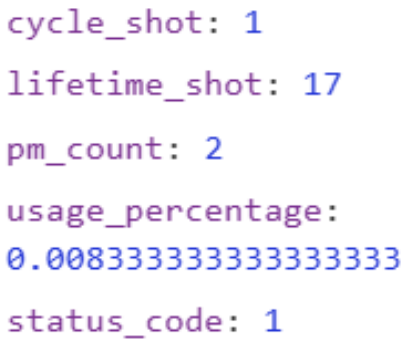
Gambar 4. 10 Implementasi *Flow* Node-Red

Berdasarkan Gambar 4.10, Node-RED digunakan untuk menghubungkan komunikasi data, proses pengolahan data, penyimpanan data, dan *dahboard monitoring* dalam satu alur sistem yang terintegrasi. Selain itu, Node-RED juga digunakan untuk mengimplementasikan proses ETL (*Extract, Transform, Load*) pada sistem *monitoring*. Proses *extract* dilakukan dengan mengambil data *monitoring* dari MQTT *Broker*. Selanjutnya pada tahap *transform*, sistem melakukan validasi data *monitoring*, pengelompokan identitas *dies*, pengolahan *timestamp*, klasifikasi status *preventive maintenance*, serta pengelolaan *history maintenance*.

Data hasil pengolahan kemudian diteruskan menuju InfluxDB pada tahap *load* untuk disimpan sebagai data *historys monitoring*. Berikut merupakan data hasil ETL.



Gambar 4. 11 Informasi *Extract*



Gambar 4. 12 *Transform* data

Selain proses ETL, sistem juga melakukan perhitungan parameter *monitoring* yang terdiri dari *cycle shot*, *lifetime shot*, *usage percentage*, *remaining shot*, *maintenance count*, repair *count*, dan status *Operational dies*. Parameter tersebut digunakan untuk menggambarkan kondisi pemakaian *dies* selama proses produksi berlangsung. Sistem selanjutnya mengelompokkan kondisi *dies* ke dalam empat kategori status *preventive maintenance* berdasarkan tingkat pemakaian *dies* terhadap batas *preventive maintenance* yang telah ditentukan. Klasifikasi status *preventive maintenance* dalamTabel 4.4 sebagai berikut.

Tabel 4. 4 Status *Preventive*

Usage percentage	**Status**	**Interpretasi**
< 80%	NORMAL	Kondisi *dies* masih aman
80% – 99%	*WARNING*	Mendekati batas *preventive maintenance*
=100%	*DUE*	Melebihi batas *preventive maintenance*
>100%	*CRITICAL*	Risiko kegagalan *dies* meningkat

Penggunaan klasifikasi status tersebut bertujuan untuk membantu operator dan bagian *maintenance* dalam mengidentifikasi kondisi *dies* secara lebih cepat berdasarkan tingkat pemakaian aktual *dies* selama proses produksi berlangsung. Data hasil pengolahan kemudian disimpan pada InfluxDB sebagai data *history*s *monitoring*. Proses penyimpanan dilakukan secara berkelanjutan sehingga setiap perubahan data *Operational dies* dapat direkam dan digunakan kembali pada proses *monitoring* maupun visualisasi *dahboard*. Menurut Kleppmann (2021), basis data *time-series* memungkinkan penyimpanan data *history*s secara lebih efisien pada sistem *monitoring* yang menghasilkan data berbasis waktu secara berkelanjutan.

Selain digunakan untuk penyimpanan *history monitoring*, data pada InfluxDB juga digunakan sebagai sumber informasi untuk *dahboard monitoring*. Dengan pendekatan tersebut, seluruh aktivitas pemakaian *dies* dapat terdokumentasi dan ditampilkan kembali dalam bentuk informasi yang mendukung kegiatan *preventive maintenance*. Melalui proses pengiriman dan pengolahan data tersebut, sistem *monitoring* tidak hanya berfungsi sebagai media pencatatan aktivitas *dies*, tetapi juga menghasilkan informasi yang mendukung implementasi *Operational Business Intelligence* pada kegiatan *preventive maintenance dies*.

4.3.4 Visualisasi dan Penyajian Informasi (*Output*)

Tahap visualisasi dan penyajian informasi merupakan tahap akhir pada sistem *monitoring* yang berfungsi untuk menyampaikan hasil pengolahan data kepada pengguna dalam bentuk informasi yang lebih mudah dipahami. Informasi yang telah melalui proses pengolahan selanjutnya ditampilkan melalui *dahboard monitoring* untuk mendukung proses pemantauan kondisi *dies*.

Menurut Few (2003), *dahboard monitoring* digunakan untuk menyajikan informasi *Operational* dalam bentuk visual yang ringkas sehingga pengguna dapat memahami kondisi sistem secara lebih cepat dan mendukung proses pengambilan keputusan. Alur visualisasi dan penyajian informasi pada sistem *monitoring* sebagai berikut.



Gambar 4. 13 Alur Visualisasi Informasi *Monitoring*

Berdasarkan Gambar 4.13, data *monitoring* yang telah tersimpan pada basis data digunakan sebagai sumber informasi untuk *dahboard monitoring*. Informasi tersebut ditampilkan dalam bentuk indikator, tabel, dan grafik sehingga pengguna dapat memantau kondisi pemakaian *dies* secara lebih mudah. Informasi yang disajikan pada *dahboard* meliputi identitas *dies* aktif, jumlah *cycle shot*, *lifetime shot*, *usage percentage*, *remaining shot*, status *Operational dies*, *history maintenance*, serta status *preventive maintenance*. Informasi tersebut ditampilkan menggunakan beberapa bentuk visualisasi seperti gauge panel, status indikator, *trend* graph, dan *monitoring* panel untuk mempermudah proses identifikasi kondisi *dies* secara cepat. Penyajian informasi tersebut bertujuan untuk membantu operator dan bagian *maintenance* dalam memahami kondisi aktual *dies* selama proses produksi berlangsung.

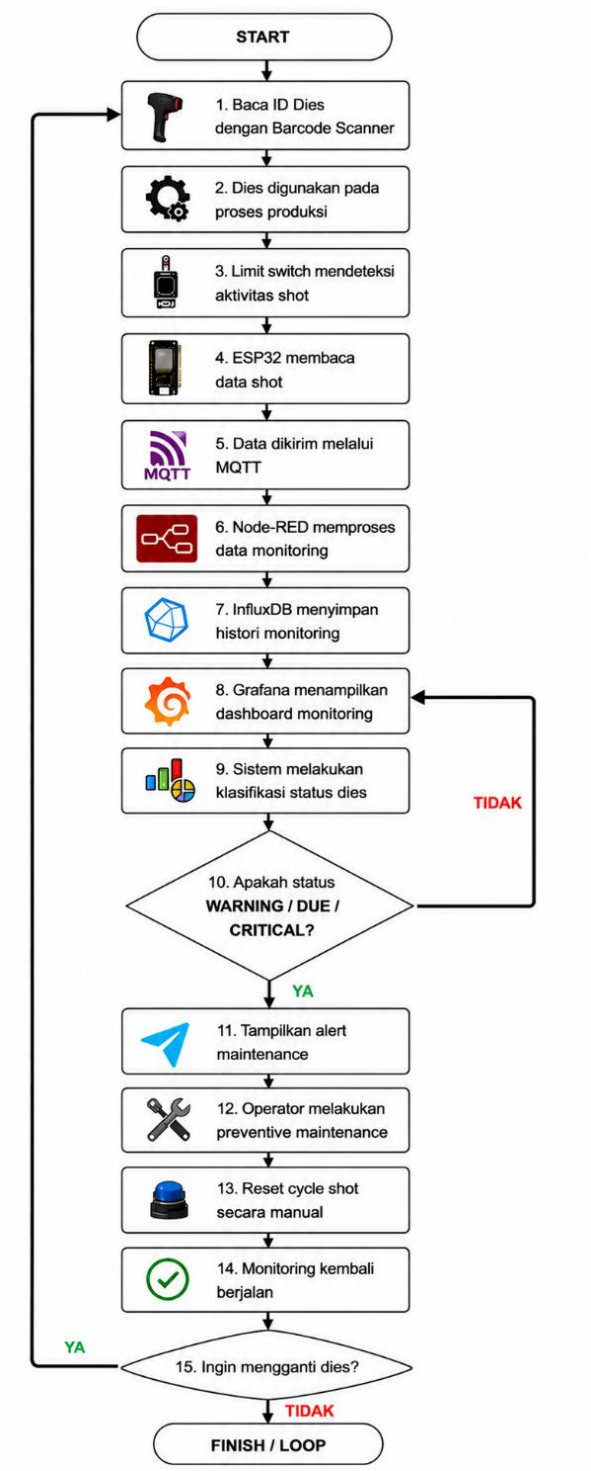
Selain menampilkan kondisi aktual *dies*, *dahboard monitoring* juga digunakan untuk menampilkan data *history*s yang tersimpan pada sistem. Penyajian data *history*s memungkinkan pengguna melakukan penelusuran terhadap riwayat penggunaan *dies* dan aktivitas *maintenance* yang telah dilakukan sebelumnya. Dengan penyajian informasi yang terpusat, proses *monitoring* dapat dilakukan secara lebih efektif dibandingkan pencatatan manual. Informasi yang diperoleh dari data *monitoring* ditampilkan pada *dahboard* secara *near real-time* sehingga pengguna dapat memantau kondisi *dies* berdasarkan data *Operational* terbaruserta mendukung pelaksanaan *preventive maintenance* secara lebih terstruktur.

4.4 Alur Kerja Sistem

Setelah rancangan arsitektur dan mekanisme pengolahan data disusun, tahap selanjutnya adalah perancangan mekanisme *operational* sistem sebagai bagian dari fase *Modeling Quick Design*. Tahap ini bertujuan menggambarkan interaksi antara pengguna dengan sistem *monitoring*, mulai dari proses registrasi *dies*, penggunaan sistem selama produksi, hingga pelaksanaan *preventive maintenance*. Perancangan mekanisme *operational* dilakukan sebagai acuan dalam

implementasi *prototype* sehingga setiap pengguna memiliki alur kerja yang jelas sesuai dengan perannya masing-masing.

Pada penelitian ini, sistem *monitoring* dapat digunakan oleh operator, supervisor, maupun bagian *maintenance* sesuai kebutuhan masing-masing. Menurut Sommerville (2016), perancangan alur kerja sistem diperlukan untuk menjelaskan hubungan antar proses dalam suatu sistem *Operational* sehingga proses *monitoring* dan pengambilan keputusan dapat berjalan secara lebih terstruktur. Alur kerja sistem *monitoring* yang dikembangkan sebagai berikut.



Gambar 4. 14 Alur Sistem Proses *Monitoring*

Berdasarkan Gambar 4.14, proses *monitoring* dimulai dengan operator melakukan pemindaian *barcode* pada *dies* yang akan digunakan pada proses produksi. Hasil pemindaian digunakan untuk mengidentifikasi *dies* aktif sehingga sistem dapat membedakan data *monitoring* untuk setiap *dies* yang digunakan. Setelah *dies* berhasil teridentifikasi, sistem mulai

melakukan *monitoring* aktivitas *shot dies* melalui *limit switch* yang terhubung dengan ESP32. Setiap aktivitas *shot* yang terdeteksi akan dicatat sebagai penambahan jumlah pemakaian *dies* dan dikirim menuju sistem *monitoring* untuk diproses lebih lanjut.

Data hasil pembacaan yang diterima oleh ESP32 untuk dilakukan proses pengolahan awal sebelum dikirim menuju sistem *monitoring* menggunakan protokol MQTT melalui jaringan WiFi. Proses pengiriman data dilakukan secara kontinu selama sistem *monitoring* aktif. Setelah data diterima oleh MQTT *Broker*, Node-RED melakukan proses pengolahan data *monitoring* seperti validasi data, pengelompokan *dies*, pengelolaan *history monitoring*, dan klasifikasi kondisi *preventive maintenance dies*. Data hasil pengolahan kemudian disimpan pada InfluxDB sebagai basis data *time-series* sehingga *history monitoring* dapat disimpan dan digunakan kembali untuk proses *monitoring Operational* maupun evaluasi *history* pemakaian *dies*. Selanjutnya, data *monitoring* ditampilkan pada *dahboard* Grafana dalam bentuk visualisasi *Operational* seperti jumlah *shot*, *usage percentage*, *remaining shot*, dan status kondisi *dies*. *Dahboard monitoring* digunakan untuk membantu operator dan bagian *maintenance* dalam memantau kondisi *dies* secara lebih cepat dan terstruktur.

Pada sistem *monitoring*, klasifikasi kondisi *dies* dilakukan berdasarkan tingkat pemakaian *dies* terhadap batas *preventive maintenance* yang telah ditentukan sebelumnya. Ketika nilai *usage percentage* mencapai batas tertentu, sistem akan menampilkan status *WARNING*, *DUE*, atau *CRITICAL* pada *dahboard monitoring*. Penggunaan *alert monitoring* bertujuan untuk membantu operator mengidentifikasi kondisi *dies* yang mendekati batas *preventive maintenance* sehingga tindakan *maintenance* dapat dilakukan sebelum terjadi risiko kerusakan *dies* yang lebih besar. Selama proses produksi berlangsung, sistem akan terus memperbarui data *monitoring* berdasarkan aktivitas *shot* yang diterima. Apabila *dies* telah mencapai batas *preventive maintenance* yang ditentukan, sistem akan menampilkan status dan indikator yang menunjukkan bahwa *dies* memerlukan tindakan *maintenance*. Sistem *maintenance* berbasis kondisi *Operational* mampu membantu perusahaan mengurangi risiko *downtime* produksi melalui proses *monitoring* kondisi peralatan secara kontinu. (Mobley et al., 2008)

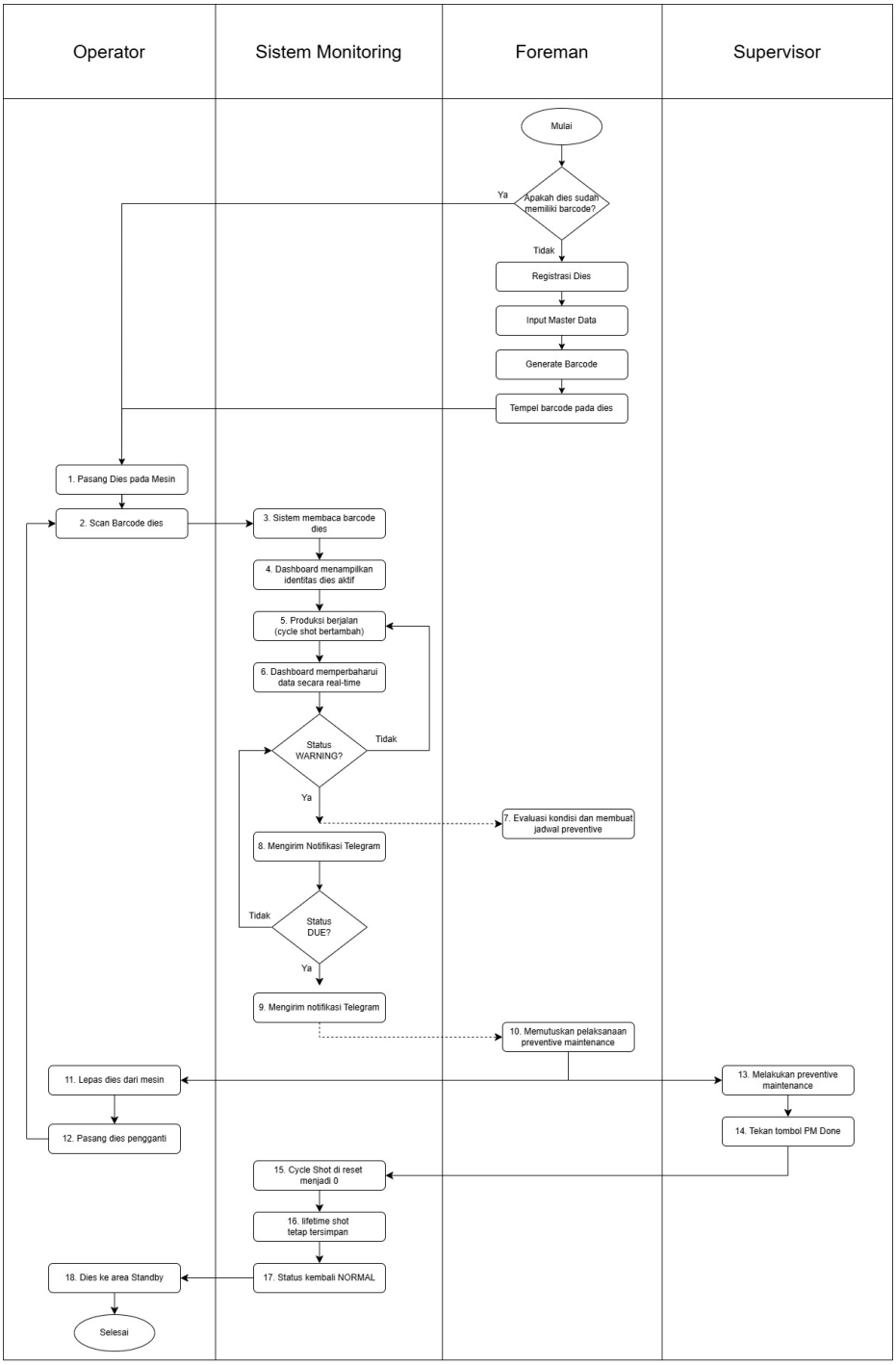
Setelah kegiatan *preventive maintenance* dilakukan, operator dapat melakukan reset *cycle shot* melalui tombol yang tersedia pada sistem. Proses reset dilakukan untuk memulai kembali perhitungan siklus pemakaian *dies* tanpa menghapus data *history* penggunaan *dies* yang telah tersimpan sebelumnya. Selain mendukung proses *preventive maintenance*, sistem juga

memungkinkan pergantian *dies* melalui proses pemindaian *barcode dies* yang baru. Dengan mekanisme tersebut, *monitoring* dapat dilakukan secara berkelanjutan meskipun terjadi pergantian *dies* selama proses produksi berlangsung.

4.4.1 Alur Penggunaan Sistem Pengguna

Activity Diagram adalah *Workflow* dalam operasi sistem dan interaksi pengguna yang berjalan dan bagaimana alur tersebut berhenti. *Activity Diagram* menjelaskan alur dari tampilan dalam sistem. Diagram ini berisi elemen-elemen dengan beberapa bentuk khusus yang dihubungkan satu sama lain oleh panah. Tanda panah menunjukkan alur kerja secara berurutan mulai dari bagian atas ke bawah hingga akhir aktivitas (Wicaksono et al., 2021).

Alur penggunaan sistem menggambarkan tahapan *operational* yang dilakukan oleh setiap pengguna dalam memanfaatkan sistem *monitoring* pemeliharaan *dies*. Berikut merupakan alur yang menunjukkan interaksi antara operator produksi, sistem *monitoring*, *foreman*, dan *supervisor maintenance* mulai dari pemasangan *dies* hingga proses *preventive maintenance* selesai dilakukan.



Gambar 4. 15 *Activity Diagram*

Berdasarkan Gambar 4.15, proses diawali dengan pemeriksaan apakah *dies* yang akan digunakan telah memiliki barcode. Apabila *dies* belum memiliki *barcode*, foreman melakukan registrasi *dies* dengan menginput master data yang meliputi identitas *dies* dan parameter *preventive maintenance*. Selanjutnya sistem menghasilkan *barcode* yang kemudian ditempelkan pada *dies* sebagai identitas unik. Apabila *dies* telah memiliki *barcode*, operator

dapat langsung memasang *dies* pada mesin produksi dan melakukan pemindaian barcode sebelum proses produksi dimulai.

Setelah *barcode* berhasil dipindai, sistem membaca identitas *dies* dan mengambil data master yang telah tersimpan untuk menampilkan informasi *dies* aktif pada *dahboard monitoring*. Selama proses produksi berlangsung, sistem secara otomatis menghitung *cycle shot*, memperbarui data *monitoring* secara *real-time*, serta mengevaluasi jumlah *shot* berdasarkan batas *preventive maintenance* yang telah ditetapkan. Apabila jumlah *cycle shot* belum mencapai batas *warning*, proses produksi tetap berlangsung dan sistem terus melakukan pemantauan. Namun, ketika jumlah *cycle shot* telah mencapai kondisi *warning*, sistem mengirimkan notifikasi melalui Telegram sehingga foreman dapat melakukan evaluasi terhadap kondisi *dies* dan jadwal produksi sebagai dasar penentuan waktu pelaksanaan *preventive maintenance*.

Apabila jumlah *cycle shot* telah mencapai batas *due*, sistem kembali mengirimkan notifikasi kepada *foreman*. Selanjutnya *foreman* memutuskan pelaksanaan *preventive maintenance*, kemudian operator melepas *dies* dari mesin dan menggantinya dengan *dies* lain agar proses produksi tetap berjalan. Setelah itu, *supervisor maintenance* melaksanakan *preventive maintenance* sesuai dengan Instruksi Kerja (IK) yang berlaku. Setelah kegiatan *preventive maintenance* selesai dilakukan, supervisor menekan tombol PM *Done* pada *prototype*. Perintah tersebut membuat sistem mengatur ulang (*reset*) nilai *cycle shot* menjadi nol, sedangkan nilai lifetime *shot* tetap dipertahankan sebagai riwayat penggunaan *dies*. Status *dies* kemudian kembali menjadi NORMAL, dan *dies* yang telah selesai dilakukan *preventive maintenance* ditempatkan pada area stand by sehingga siap digunakan kembali pada proses produksi berikutnya.

.

4.4.2 Ketentuan *Preventive Maintenance*

Pelaksanaan *preventive maintenance* pada penelitian ini mengacu pada Instruksi Kerja (IK) *Preventive maintenance Dies* yang digunakan sebagai pedoman pelaksanaan pemeliharaan *dies* di perusahaan. Sistem *monitoring* yang dikembangkan tidak menggantikan prosedur *preventive maintenance*, melainkan berfungsi sebagai pendukung dalam memberikan informasi kondisi penggunaan *dies* sehingga keputusan pelaksanaan *preventive maintenance* dapat dilakukan sesuai dengan ketentuan yang berlaku.

Tabel 4. 5 Ketentuan *Preventive Maintenance*

Tahapan	Ketentuan IK	Aktivitas pada Sistem ***Monitoring***
Penjadwalan *Preventive*	*Preventive maintenance* disusun berdasarkan jadwal bulanan dan data jumlah *shot dies*.	*Dashboard* menampilkan jumlah *shot* dan status *dies* sebagai dasar penyusunan jadwal PM.
Kondisi *WARNING*	*Dies* mendekati batas *preventive* sehinga diperlukan persiapan *preventive maintenance*.	Sistem mengirim notifikasi. Operator melapor kepada *foreman* untuk persiapan *preventive maintenance*.
Kondisi *DUE*	*Preventive maintenance* dilakukan ketika jumlah *shot* telah mencapai interval *preventive* sesuai ketentuan perusahaan (10.000–15.000 *shot*).	*Foreman* memutuskan pelaksanaan PM sesuai kondisi produksi.
Overshoot	Apabila jumlah *shot* melebihi 15.000 *shot*, maka harus mengikuti prosedur *Overshot Standard Preventive.*	Sistem mengubah status menjadi CRITICAL sebagai penanda pelaksanaan IK *Overshoot*.
Pelaksanaan *Preventive*	*Dies* dilepas dari mesin dan dilakukan proses *preventive maintenance* sesuai IK.	*Supervisor* melakukan *preventive maintenance* sesuai IK.
Setelah *Preventive*	Setelah *preventive* selesai, *dies* ditempatkan pada area *Dies Standby* dan diberi kartu informasi bahwa *dies* siap digunakan kembali.	*Supervisor* menekan tombol PM *done*, *cycle shot* di-*reset* dan status kembali NORMAL

4.4.3 Pengelolaan Master Data

Mekanisme penentuan *preventive maintenance* pada sistem *monitoring* dilakukan dengan membandingkan nilai *cycle shot* terhadap parameter *warning threshold*, *preventive target*, dan *maximum tolerance* yang telah dikonfigurasi pada sistem. Hasil perbandingan tersebut digunakan untuk menentukan status penggunaan *dies* secara otomatis sehingga kondisi pemakaian *dies* dapat ditampilkan pada *dahboard monitoring* secara *near real-time* sebagai dasar dalam mendukung pengambilan keputusan *preventive maintenance*.

Pada *prototype* yang dikembangkan, seluruh *dies* menggunakan parameter *preventive maintenance* yang sama sesuai standar *operational* perusahaan. Oleh karena itu, nilai *warning* threshold, preventive target, dan maximum tolerance dikonfigurasi secara seragam pada Function Node di Node-RED. Konfigurasi tersebut digunakan sebagai acuan sistem dalam menentukan status NORMAL, *WARNING*, *DUE*, dan *CRITICAL*.

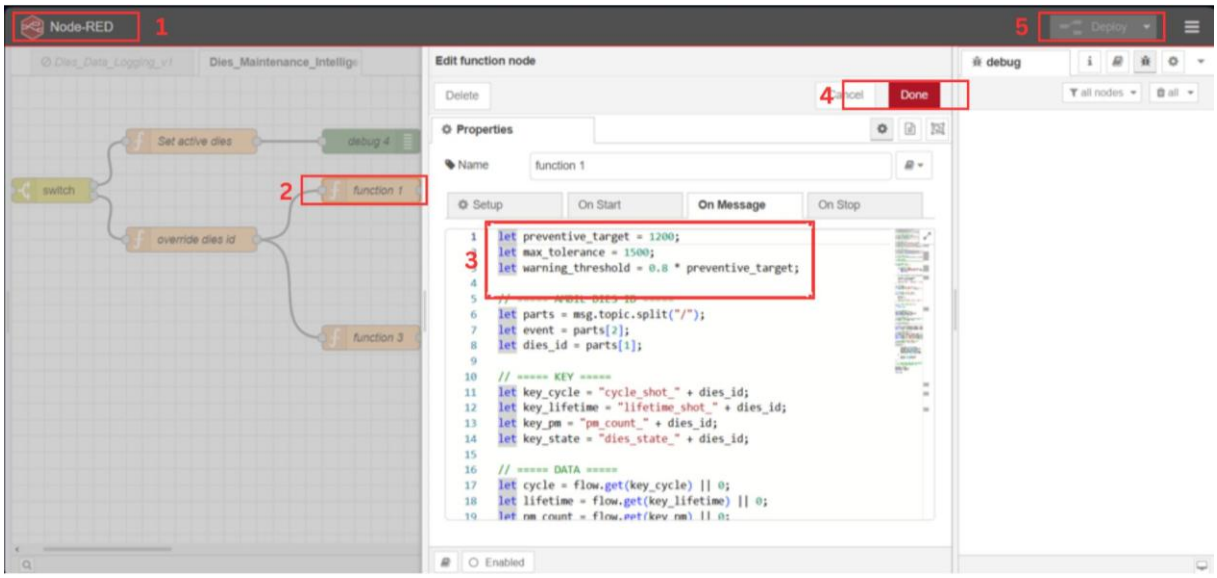
Tabel 4. 6 Acuan Parameter *Preventive Maintenance*

Parameter	Fungsi
Warning Threshold	Menentukan batas awal status *WARNING*.
Preventive Target	Menentukan batas pelaksanaan *preventive maintenance* dan perubahan status menjadi *DUE*.
Maximum Tolerance	Menentukan batas maksimum penggunaan *dies* sebelum status berubah menjadi *CRITICAL*.

Ketiga parameter tersebut menjadi acuan utama dalam proses penentuan status penggunaan *dies* selama sistem *monitoring* berjalan. Setiap perubahan nilai parameter akan diterapkan sebagai konfigurasi terbaru sehingga sistem secara otomatis menggunakan batas *preventive maintenance* yang baru tanpa mengubah mekanisme *monitoring*. Oleh karena itu, apabila perusahaan menetapkan perubahan standar *preventive maintenance*, pengguna yang memiliki hak akses dapat memperbarui parameter tersebut melalui Node-RED dengan tahapan sebagai berikut.

Tabel 4. 7 Tahapan Perubahan Parameter *Preventive maintenance*

Tahapan	Aktivitas
1	Membuka aplikasi Node-RED melalui web browser.
2	Membuka *Function Node* yang berisi konfigurasi parameter *preventive maintenance*.
3	Mengubah nilai *warning* threshold, preventive target, dan maximum tolerance sesuai standar perusahaan.
4	Menekan tombol Deploy untuk menerapkan konfigurasi baru ke sistem *monitoring*.
5	Sistem menggunakan parameter baru untuk menentukan status NORMAL, *WARNING*, *DUE*, dan *CRITICAL* pada seluruh *dies*.



Gambar 4. 16 Tahapan Parameter pada *Function Node*

Perubahan parameter dilakukan tanpa perlu mengubah program pada ESP32 maupun *dahboard* Grafana karena konfigurasi hanya dilakukan pada *Function Node* di Node-RED sebagai pusat pengolahan data. Perubahan konfigurasi hanya memengaruhi parameter penentuan status *preventive maintenance* dan tidak mengubah data histori yang telah tersimpan pada InfluxDB. Dengan demikian, data *lifetime shot* dan riwayat penggunaan *dies* tetap dapat digunakan sebagai data historis meskipun terjadi perubahan standar *preventive maintenance*. Pada penelitian ini, perubahan parameter *preventive maintenance* diterapkan secara seragam untuk seluruh *dies* karena objek pengujian menggunakan standar *preventive maintenance* yang sama sesuai standar *operational* perusahaan.

4.5 Perancangan *Hardware*

Tahap ini merupakan implementasi fase *Construction of Prototype*, yaitu pembangunan *prototype* perangkat keras. Perancangan *hardware* dilakukan untuk mendukung proses akuisisi data, pengiriman data, dan *monitoring* pemakaian *dies* pada sistem yang dikembangkan. Perangkat keras yang digunakan dipilih berdasarkan kebutuhan fungsional sistem sehingga mampu merepresentasikan proses *monitoring preventive maintenance* pada *prototype*.

Menurut Monostori et al. (2021), implementasi sistem *monitoring* berbasis IoT memerlukan integrasi perangkat akuisisi data, perangkat komunikasi, dan perangkat *monitoring* dalam satu kesatuan sistem sehingga proses pengumpulan data operasional dapat berlangsung secara terus-menerus. Pada penelitian ini, perancangan *hardware* difokuskan pada pengembangan *prototype monitoring dies* yang mampu merepresentasikan proses *monitoring*

preventive maintenance pada lingkungan industri manufaktur. Meskipun masih berada pada tahap *prototype*, sistem *hardware* yang dikembangkan tetap disusun menggunakan konsep *monitoring* industri sehingga dapat dikembangkan lebih lanjut menuju implementasi industri aktual.

4.5.1 Komponen *Hardware*

Komponen *hardware* yang digunakan pada penelitian ini dipilih berdasarkan kebutuhan pengembangan *prototype monitoring dies* berbasis IoT. Setiap komponen memiliki fungsi yang saling mendukung dalam proses akuisisi data, komunikasi data, dan *monitoring Operational dies*. Spesifikasi *hardware* yang diterapkan melalui Tabel 4.8.

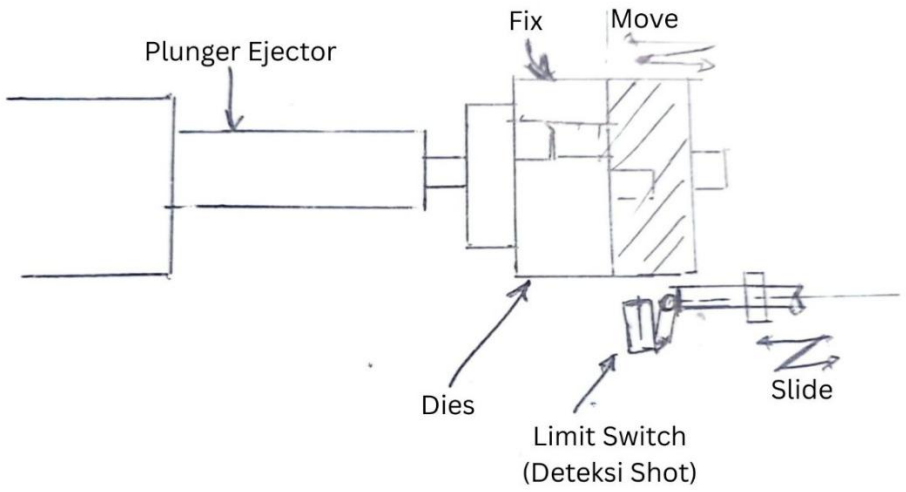
Tabel 4. 8 Spesifikasi *Hardware Prototype*

No	Komponen	Pcs	Spesifikasi	Fungsi
1	ESP32 DevKit V4	1	Dual Core 240 MHz Wifi 2.4 GHz	Akuisisi dan komunikasi data
2	*Limit switch*	1	Mechanical Roller	Deteksi *shot dies*
3	*Push button*	2	Normally Open	*Input preventive maintenance*
4	*Barcode scanner*	1	2D USB *Barcode scanner*	Identifikasi *dies*
5	LED Indikator	3	-	Indikator status *dies*
6	USB Laptop 5V	1	-	Catu daya sistem
7	Kabel Jumper	10	*Male to male*	Koneksi antar perangkat
8	Terminal Block	1	830 *point*	Distribusi koneksi *wiring*
9	Breadboard	1	-	Koneksi sementara *prototype*
10	Wifi	1	-	Komunikasi data *Monitoring*
11	Laptop	1	Asus	Menjalankan Node-Red, InfluxDB, dan Grafana

ESP32 DevKitC V4 digunakan sebagai perangkat utama untuk membaca data dan mengirimkan informasi *monitoring* ke sistem pusat. *Limit switch* digunakan sebagai pendeteksi aktivitas *shot dies* pada *prototype*, sedangkan *push button* digunakan untuk menjalankan fungsi reset *preventive maintenance* secara manual. Selain itu, sistem memanfaatkan jaringan WiFi sebagai media komunikasi data dan laptop sebagai server lokal yang menjalankan MQTT *Broker*, Node-RED, InfluxDB, serta Grafana. Kombinasi perangkat tersebut memungkinkan proses *monitoring* dilakukan secara terintegrasi dalam satu sistem.

4.5.2 Penempatan Sensor Pada *Dies*

Penempatan sensor pada *prototype dies* dirancang untuk mendeteksi aktivitas gerakan mekanisme *dies* yang merepresentasikan proses *shot* pada mesin *die casting*. Pada penelitian ini, *limit switch* ditempatkan pada bagian mekanisme gerak *prototype dies* sehingga sensor dapat aktif setiap kali terjadi pergerakan siklus *dies*. Ilustrasi penempatan sensor pada *prototype dies* dapat dilihat sebagai berikut.



Gambar 4. 17 Penempatan *LimitSwitch Dies*

Berdasarkan Gambar 4.17 menunjukkan Lokasi pemasangan *limit switch* pada mesin *dies. Limit switch* dipasang pada bagian sisi mekanisme gerak *prototype dies* yang mengalami kontak langsung ketika proses simulasi *shot* berlangsung. Ketika mekanisme bergerak dan menyentuh *limit switch*, sensor akan menghasilkan sinyal *digital* yang diteruskan menuju ESP32 sebagai data *trigger shot*.

Penempatan sensor dilakukan dengan mempertimbangkan kemudahan pemasangan, kestabilan pembacaan data, dan kemampuan sensor dalam mendeteksi pergerakan secara konsisten. Dengan penempatan tersebut, proses pencatatan *shot* dapat dilakukan secara otomatis tanpa memerlukan input manual dari operator.

4.5.3 Perbandingan Sistem *Prototype* dan Implementasi Industri

Prototype yang dikembangkan pada penelitian ini dirancang untuk merepresentasikan konsep *monitoring preventive maintenance dies* yang dapat diterapkan pada lingkungan industri. Meskipun terdapat perbedaan pada beberapa komponen fisik, alur *monitoring* dan pengolahan data yang digunakan tetap mengikuti prinsip yang sama dengan implementasi industri.

Tabel 4. 9 Spesifikasi Sistem

No	Aspek Sistem	*Prototype*	Industri
1	Perangkat Kontrol	ESP32 DevKit V4	ESP32 DevKit V4
2	Sensor *Shot*	*Limit switch* / *Push button*	Omron WLCA2 *Limit switch*
3	Komunikasi Data	WiFi dan MQTT	Ethernet WiFi Industrial MQTT
4	Sistem *Monitoring*	Grafana *Dahboard*	Grafana *Dahboard*
5	Pengolahan Data	Node-RED	Node- RED
6	*Database*	InfluxDB	InfluxDB
7	Sumber Daya	USB Laptop 5V	Industrial Power Supply 24V DC
8	Lingkungan Operasi	*Prototype*	Area Produksi
9	Kapasitas *Monitoring*	1 Mesin dan 2 *Dies*	*Multi Dies* dan *Multi* Mesin
10	*Alert Maintenance*	*Dahboard* dan Telegram	*Dahboard* dan Telegram
11	Penyimpanan Data	SSD Laptop	SSD Server
12	*Backup Power*	-	UPS
13	Stabilitas	Skala *Prototype*	Industrial Grade

Berdasarkan Tabel 4.9 menunjukkan bahwa perbandingan antara implementasi *prototype* pada penelitian ini difokuskan untuk memvalidasi proses *monitoring*, komunikasi data, pengolahan data *Operational*, dan visualisasi *dahboard monitoring* sebelum diterapkan pada implementasi di lingkungan industri aktual.

Perbedaan utama terletak pada jenis sensor, sumber daya, dan lingkungan *operational* yang digunakan. Pada *prototype*, aktivitas *shot* dideteksi menggunakan *limit switch* dan sistem memperoleh catu daya dari koneksi USB laptop 5V. Sementara itu, pada implementasi industri sistem dapat menggunakan limit switch industrial dengan sumber daya dari *power supply* 24V DC. Selain itu, UPS dapat ditambahkan sebagai sumber daya cadangan untuk menjaga sistem *monitoring* tetap beroperasi saat terjadi gangguan listrik.

Meskipun demikian, proses akuisisi data, komunikasi MQTT, penyimpanan data *time-series*, serta visualisasi *dahboard* tetap menggunakan prinsip kerja yang sama. Dengan demikian, *prototype* yang dikembangkan mampu merepresentasikan alur *monitoring* yang dapat diterapkan pada lingkungan industri dengan penyesuaian pada aspek perangkat keras.

4.6 Perancangan *Software* dan Data *Flow*

Tahap ini merupakan implementasi perangkat lunak pada fase *Construction of Prototype*. Dalam tahap *Construction of Prototype* dilakukan implementasi perangkat lunak berupa Node-RED, MQTT Broker, InfluxDB, dan Grafana sebagai sistem *monitoring*. Setiap perangkat lunak memiliki fungsi yang berbeda namun saling terintegrasi sehingga membentuk satu kesatuan sistem *monitoring*. Perancangan *software* dilakukan untuk mendukung proses

komunikasi data, pengolahan data *Operational*, penyimpanan data *history*s, serta visualisasi informasi *monitoring* pada sistem yang dikembangkan. Integrasi beberapa *software* diperlukan agar data yang diperoleh dari perangkat IoT dapat diolah dan disajikan dalam bentuk informasi yang mendukung kegiatan *preventive maintenance dies*. Menurut Sommerville (2016), integrasi perangkat lunak dalam suatu sistem diperlukan untuk memastikan setiap fungsi dapat berjalan secara terkoordinasi dan mendukung kebutuhan pengguna secara efektif. Pada penelitian ini, sistem *monitoring* dibangun menggunakan beberapa *software* yang saling terhubung mulai dari proses pengiriman data, pengolahan data, penyimpanan data, hingga visualisasi informasi *monitoring*.

4.6.1 *Software* sistem *monitoring*

Software yang digunakan pada penelitian ini dipilih berdasarkan kebutuhan sistem *monitoring* yang mencakup proses pemrograman perangkat IoT, komunikasi data, pengolahan data, penyimpanan data, dan visualisasi informasi *monitoring*. Setiap *software* memiliki fungsi yang berbeda namun saling terintegrasi dalam mendukung *Operational* sistem *monitoring*. *Software* yang digunakan disajikan dalamTabel 4.10.

Tabel 4. 10 *Software* Sistem *Monitoring*

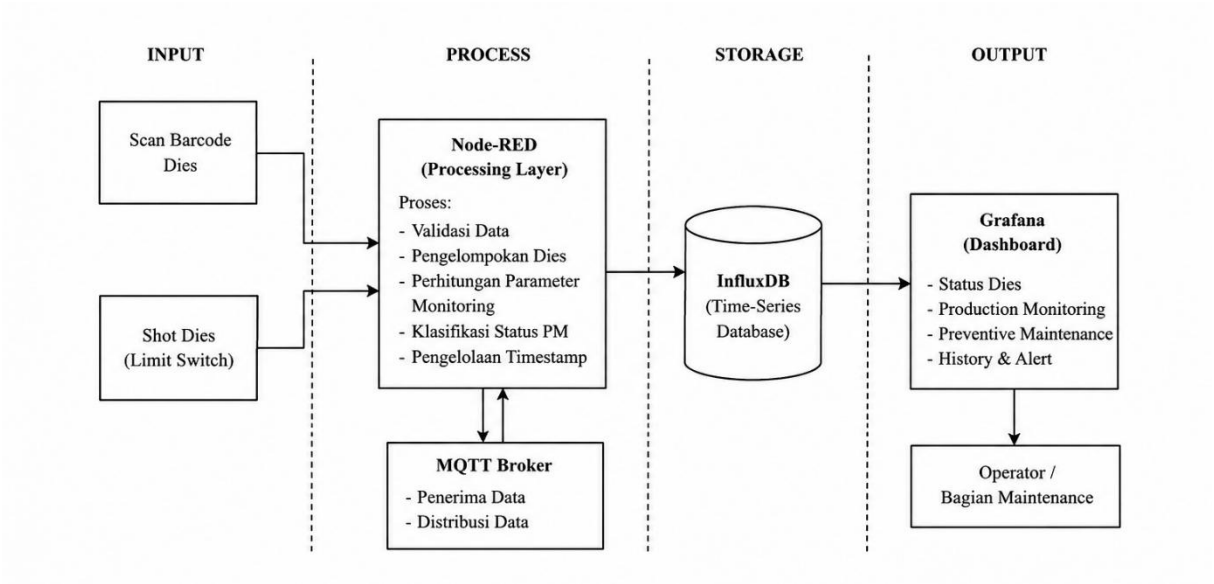
No	*Software*	**Versi**	Fungsi
1	Arduino IDE	2.3.7	Pemrograman dan konfigurasi ESP32
2	MQTT *Broker*	2.1.2	Media komunikasi data *monitoring*
3	Node-RED	4.1.5	Pengolahan dan integrasi data *monitoring*
4	InfluxDB	2.8.0	Penyimpanan data *time-series*
5	Grafana	12.3.3	*Dahboard* visualisasi *monitoring*

Arduino IDE digunakan untuk melakukan pemrograman ESP32 DevKitC V4 yang berfungsi sebagai perangkat akuisisi data pada sistem *monitoring*. Program yang dibuat digunakan untuk membaca aktivitas *shot dies* dan mengirimkan data *monitoring* menuju sistem pengolahan data menggunakan komunikasi MQTT. MQTT *Broker* digunakan sebagai media komunikasi data antar perangkat pada sistem *monitoring*. Mekanisme komunikasi yang digunakan memungkinkan proses pertukaran data dilakukan secara berkelanjutan selama sistem beroperasi. Node-RED digunakan sebagai media integrasi dan pengolahan data *monitoring*. *Software* ini berfungsi untuk menerima data, melakukan pengolahan parameter *monitoring*, serta meneruskan data menuju basis data *monitoring*.

InfluxDB digunakan sebagai basis data utama pada sistem *monitoring* untuk menyimpan data *Operational dies* yang dihasilkan selama proses *monitoring* berlangsung. Data yang tersimpan meliputi informasi penggunaan *dies*, *history maintenance*, dan parameter *monitoring* yang digunakan pada *dahboard* Grafana. Grafana digunakan untuk menampilkan informasi *monitoring* dalam bentuk *dahboard* yang memudahkan pengguna dalam memantau kondisi pemakaian *dies* dan status *preventive maintenance*.

4.6.2 Data *Flow* Sistem *Monitoring*

Data *flow* pada sistem *monitoring* menggambarkan proses perpindahan data mulai dari proses akuisisi data hingga penyajian informasi *monitoring* pada *dahboard*. Pada penelitian ini, aliran data dirancang menggunakan pendekatan *Operational Business Intelligence* agar proses *monitoring* dapat berjalan secara berkelanjutan.



Gambar 4. 18 Data *Flow* Sistem

Proses aliran data dimulai ketika perangkat IoT menghasilkan data *monitoring* berdasarkan aktivitas pemakaian *dies*. Data tersebut kemudian diteruskan menuju sistem komunikasi data untuk dikirim ke platform pengolahan data. Selanjutnya, data *monitoring* diterima oleh sistem pengolahan data untuk dilakukan validasi, pengelolaan parameter *monitoring*, dan pembentukan informasi yang dibutuhkan oleh pengguna. Data hasil pengolahan kemudian disimpan sebagai data *history*s pada basis data *monitoring*.

Data yang telah tersimpan selanjutnya digunakan sebagai sumber informasi pada *dahboard monitoring*. Dengan mekanisme tersebut, setiap perubahan data yang terjadi pada

sistem dapat diteruskan hingga tahap visualisasi informasi. Aliran data yang terintegrasi memungkinkan proses penyimpanan, pengolahan, dan penyajian informasi dilakukan secara berkelanjutan sehingga mendukung kebutuhan *monitoring Operational* berbasis data (Kleppmann, 2021).

4.6.3 Struktur Data *Monitoring*

Struktur data *monitoring* dirancang untuk mendukung proses penyimpanan, pengolahan, dan visualisasi informasi pemakaian *dies* pada sistem *monitoring*. Data yang tersimpan dalam InfluxDB digunakan sebagai dasar dalam proses *monitoring Operational* maupun kegiatan *preventive maintenance*.

Menurut Kleppmann (2021), perancangan struktur data yang baik diperlukan untuk memastikan data dapat disimpan, diakses, dan diolah secara konsisten selama sistem beroperasi. Pada penelitian ini, setiap data *monitoring* disimpan sebagai data *time-series* yang dilengkapi dengan informasi identitas *dies*, parameter pemakaian *dies*, status *Operational*, serta informasi *preventive maintenance*. Berikut merupakan struktur data *monitoring* yang digunakan pada sistem dalam Tabel 4.11 dan 4.12.

Tabel 4. 11 Struktur Data Monitroing

No	Field Data	Tipe Data	Deskripsi
1	*dies*_id	*String*	Identitas *dies*
2	*cycle_shot*	*Integer*	Jumlah *shot* sejak PM terakhir
3	*lifetime_shot*	*Integer*	Total *shot* sepanjang umur *dies* dan tidak pernah di-reset
4	usage_percentage	*Float*	Persentase pemakaian *dies* terhadap batas *preventive maintenance*
5	*remaining_shot*	*Integer*	Sisa *shot* menuju batas PM
6	*maintenance_count*	*Integer*	Jumlah *preventive maintenance* yang telah dilakukan
7	status_*Operational*	*String*	Status *Operational dies* (*RUNNING*, *STANDBY*, PM)
8	*timestamp*	*Datetime*	Waktu *monitoring*
9	pm_status	*String*	Status *preventive maintenance* (NORMAL, *WARNING*, *DUE*, *CRITICAL*)

Tabel 4. 12 Konsep *measurement, tag* dan *field* dalam *influx*

No	***Measurement***	***Tag***	***Field***
1	*cycle_shot*	*dies_id*	*dies_counter*
2	*lifetime_shot*	*dies_id*	*dies_counter*

No	*Measurement*	*Tag*	*Field*
3	usage_percentage	*dies_id*	*dies_counter*
4	*remaining_shot*	*dies_id*	*dies_counter*
5	*maintenance_count*	*dies_id*	*dies_counter*
6	status_*Operational*	*dies_id*	*dies_state*
7	*timestamp*	*dies_id*	*dies_state*
8	pm_status	*dies_id*	*dies_state*

Parameter *lifetime_shot* digunakan untuk mencatat total penggunaan *dies* selama masa *Operational*nya, sedangkan *cycle_shot* digunakan untuk mencatat jumlah *shot* sejak *preventive maintenance* terakhir dilakukan. Pemisahan kedua parameter tersebut memungkinkan sistem membedakan *history* penggunaan *dies* secara keseluruhan dan siklus penggunaan *dies* yang sedang berjalan. Selain parameter pemakaian *dies*, sistem juga menyimpan status *Operational dies* yang digunakan untuk menunjukkan kondisi aktual *dies* selama proses produksi berlangsung. Klasifikasi status *Operational* yang diterapkan dalam sistem disajikan melalui Tabel 4.13.

Tabel 4. 13 Status *Operational Dies*

Variable	Deskripsi
RUNNING	*Dies* sedang digunakan produksi
STANDBY	*Dies* tidak digunakan sementara
PM	*Dies* sedang dalam perbaikan

Data yang tersimpan pada sistem selanjutnya digunakan sebagai sumber informasi untuk *dahboard monitoring*, *history Operational*, serta kebutuhan evaluasi *preventive maintenance dies*.

4.7 Perancangan *Dahboard Monitoring*

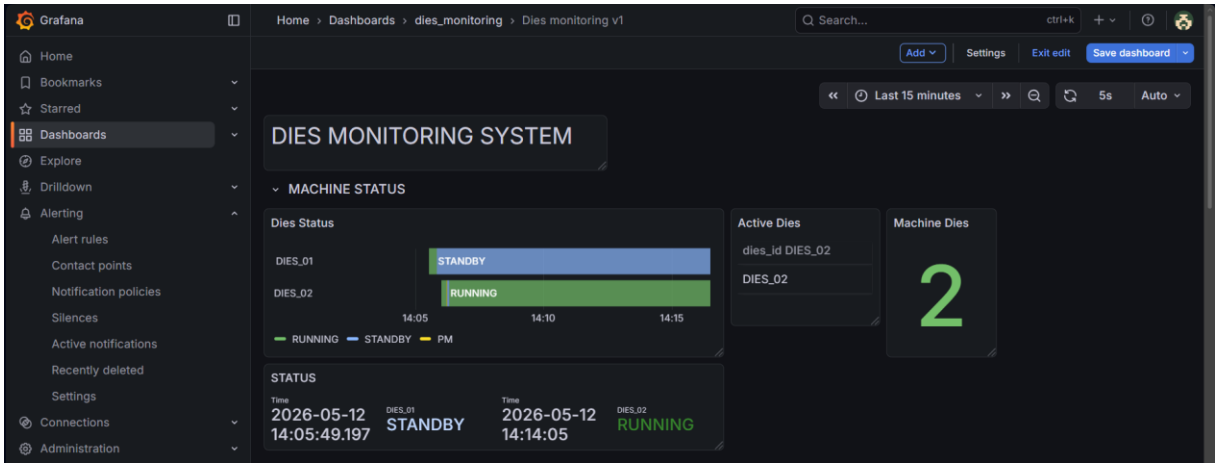
Dahboard monitoring dirancang sebagai media visualisasi informasi untuk menampilkan kondisi pemakaian *dies* secara lebih mudah dan terstruktur. Informasi yang sebelumnya berupa data *Operational* diubah menjadi tampilan visual sehingga dapat membantu operator dan bagian *maintenance* dalam memantau kondisi *dies* selama proses produksi berlangsung. *Dahboard* merupakan media visualisasi yang digunakan untuk menyajikan informasi penting secara ringkas sehingga pengguna dapat memahami kondisi *Operational* dengan lebih cepat dan mendukung proses pengambilan keputusan (Few, 2003).

Pada penelitian ini, *dahboard monitoring* dikembangkan menggunakan Grafana dan terintegrasi dengan basis data *monitoring*. *Dahboard* dirancang untuk menampilkan informasi terkait status *Operational dies*, aktivitas produksi, *preventive maintenance*, serta *history*

monitoring dalam satu tampilan yang terpusat. *Dahboard* yang dikembangkan merupakan hasil implementasi *prototype* perangkat lunak yang berfungsi sebagai media visualisasi informasi hasil pengolahan data *monitoring* secara *near real-time*. Perancangan *dahboard* dibagi menjadi beberapa bagian utama yang terdiri dari *dahboard* status *dies*, *dahboard production monitoring*, *dahboard preventive maintenance*, serta *dahboard history* dan *alert*.

4.7.1 *Dahboard Machine* Status *Dies*

Dahboard machine status digunakan untuk menampilkan kondisi *Operational dies* secara *real-time* selama proses produksi berlangsung. *Dahboard* ini berfungsi untuk membantu operator dan bagian *maintenance* dalam memantau status aktivitas *dies* yang sedang digunakan pada sistem *monitoring*. Informasi yang ditampilkan meliputi status *RUNNING*, *STANDBY*, dan PM pada setiap *dies* yang dimoni*t*or.

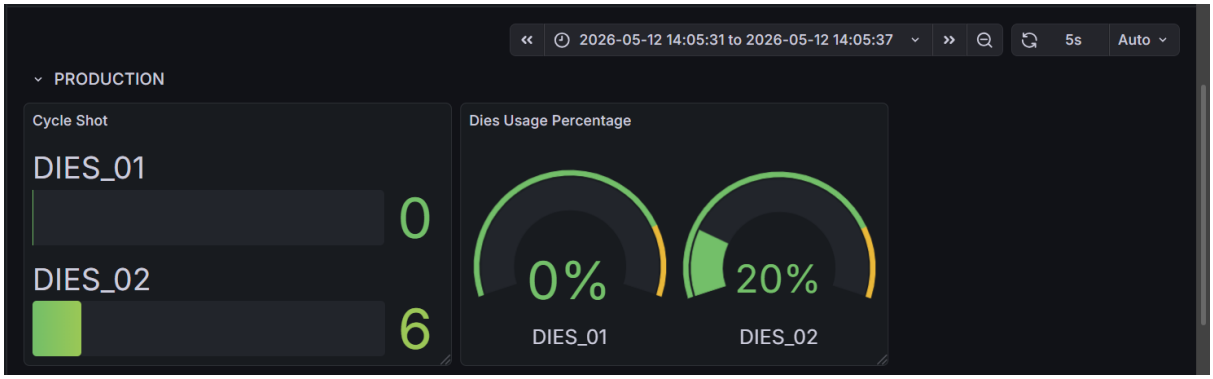


Gambar 4. 19 Status *Dies*

Berdasarkan Gambar 4.19, Informasi yang ditampilkan pada *dahboard* meliputi status *Operational dies*, identitas *dies* aktif, serta jumlah *dies* yang sedang dimonitor oleh sistem. Status *Operational* ditampilkan dalam bentuk indikator visual sehingga pengguna dapat mengetahui kondisi *dies* secara cepat tanpa harus melihat data *monitoring* secara detail. *Dahboard* ini membantu operator maupun bagian *maintenance* dalam mengidentifikasi *dies* yang sedang digunakan, *dies* yang berada pada kondisi *standby*, maupun *dies* yang sedang menjalani proses *maintenance*. Dengan penyajian informasi secara visual, proses pemantauan kondisi *dies* dapat dilakukan dengan lebih mudah dibandingkan pencatatan manual.

4.7.2 *Dahboard Production Monitoring*

Dahboard production *monitoring* digunakan untuk menampilkan informasi pemakaian *dies* selama proses produksi berlangsung seperti jumlah *cycle shot* dan persentase pemakaian *dies* terhadap batas *preventive maintenance*. *Dahboard* ini membantu bagian *maintenance* dalam memantau tingkat pemakaian *dies* serta mengetahui *dies* yang mendekati batas *preventive maintenance*.



Gambar 4. 20 *Monitoring Production*

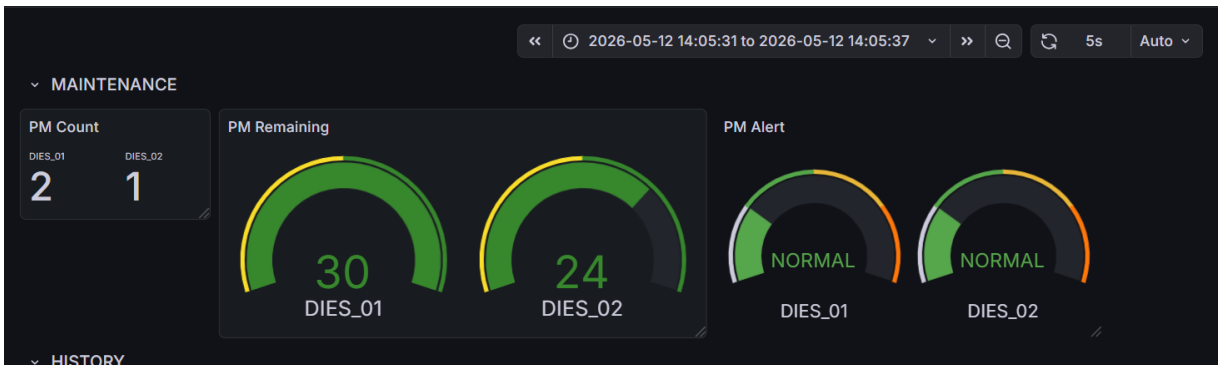
Berdasarkan Gambar 4.20 menunjukkan informasi utama yang ditampilkan pada *dahboard* ini terdiri dari *cycle shot* dan *usage percentage* untuk setiap *dies* yang dimonitor. *Cycle shot* menunjukkan jumlah *shot* yang telah digunakan sejak *preventive maintenance* terakhir dilakukan, sedangkan *usage percentage* menunjukkan tingkat pemakaian *dies* terhadap batas *preventive maintenance* yang telah ditentukan. Penyajian informasi dalam bentuk indikator visual memungkinkan pengguna mengetahui kondisi pemakaian *dies* secara lebih cepat dibandingkan melihat data numerik secara langsung. Selain itu, penggunaan grafik persentase memudahkan pengguna dalam mengidentifikasi *dies* yang mendekati batas *preventive maintenance*.

Selain itu, *dahboard* ini juga mendukung proses pemantauan beberapa *dies* secara bersamaan sehingga kondisi pemakaian masing-masing *dies* dapat dibandingkan dalam satu tampilan. Dengan demikian, operator maupun bagian *maintenance* dapat memperoleh informasi mengenai tingkat pemakaian *dies* secara lebih terstruktur dan mudah dipahami. Informasi yang ditampilkan pada *dahboard* production *monitoring* menjadi salah satu dasar dalam pelaksanaan *preventive maintenance* karena menggambarkan tingkat penggunaan aktual *dies* selama proses produksi berlangsung.

4.7.3 *Dahboard Preventive Maintenance*

Dahboard preventive maintenance digunakan untuk menampilkan informasi yang berkaitan dengan kondisi *preventive maintenance dies* seperti *remaining shot preventive maintenance*, *history maintenance*, dan jumlah *preventive maintenance* yang telah dilakukan. Informasi tersebut digunakan sebagai dasar dalam menentukan prioritas *maintenance dies*.

Dahboard ini dirancang untuk membantu operator dan bagian *maintenance* dalam memantau sisa umur pemakaian *dies* serta mengidentifikasi *dies* yang mendekati batas *preventive maintenance*. Berikut merupakan tampilan *dahboard preventive maintenance*



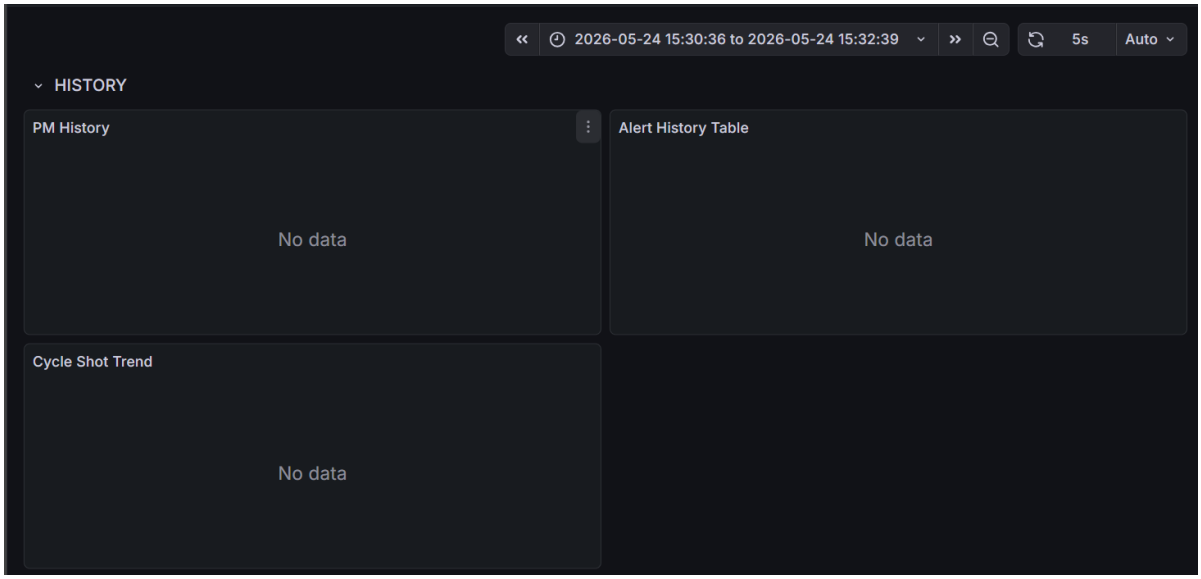
Gambar 4. 21 *Monitoring Maintenance*

Berdasarkan Gambar 4.21, Informasi yang ditampilkan pada *dahboard* ini meliputi PM *Count*, PM *Remaining*, dan PM *Alert*. PM *Count* digunakan untuk menunjukkan jumlah *preventive maintenance* yang telah dilakukan pada setiap *dies*, sedangkan PM *Remaining* menunjukkan sisa kapasitas pemakaian *dies* sebelum mencapai batas *preventive maintenance* yang telah ditentukan.

Selain itu, *dahboard* juga menampilkan PM *Alert* yang berfungsi sebagai indikator kondisi *dies* berdasarkan tingkat pemakaiannya. Informasi tersebut ditampilkan dalam bentuk indikator visual sehingga pengguna dapat mengetahui kondisi *dies* secara lebih cepat tanpa harus melakukan perhitungan secara manual. Penggunaan indikator *preventive maintenance* memungkinkan pengguna melakukan pemantauan kondisi *dies* secara berkelanjutan dan membantu mengurangi risiko keterlambatan *maintenance* akibat kurangnya informasi mengenai tingkat pemakaian *dies*. Dengan demikian, kegiatan *preventive maintenance* dapat direncanakan secara lebih terstruktur sesuai kondisi aktual *dies* yang dimoni*tor*.

4.7.4 *Dahboard History* dan *Alert*

Dahboard history dan *alert* digunakan untuk menampilkan *history monitoring* serta *alert* kondisi *dies* yang memerlukan perhatian dari operator maupun bagian *maintenance* ketika melewati batas *preventive maintenance*. *Dahboard* ini berfungsi sebagai media pemantauan *history*s yang melengkapi informasi *Operational* pada *dahboard monitoring* lainnya. Berikut merupakan tampilan *dahboard history* dan *alert*.



Gambar 4. 22 *Monitoring History* dan *Alert*

Berdasarkan Gambar 4.22, Informasi yang ditampilkan pada *dahboard* ini meliputi *history* pemakaian *dies*, *history* aktivitas *maintenance*, serta status *monitoring* yang tersimpan pada sistem. Data *history*s tersebut memungkinkan pengguna melakukan penelusuran terhadap aktivitas *dies* yang telah terjadi selama periode *Operational* tertentu.

Selain menampilkan *history monitoring*, *dahboard* juga digunakan untuk menyajikan informasi *alert* yang berkaitan dengan kondisi *dies*. Informasi *alert* membantu pengguna mengidentifikasi *dies* yang memerlukan tindakan *maintenance* berdasarkan kondisi pemakaian yang tercatat pada sistem *monitoring*. Penyajian *history* dan *alert* dalam satu *dahboard* memudahkan proses pemantauan karena pengguna dapat melihat kondisi aktual serta riwayat penggunaan *dies* secara bersamaan. Dengan demikian, proses evaluasi kondisi *dies* dapat dilakukan secara lebih terstruktur dan berbasis data *history*s yang tersimpan pada sistem.

BAB V
PENGUJIAN SISTEM DAN PEMBAHASAN

5.1 Pengujian Sistem Data

Tahap ini merupakan implementasi fase terakhir dari metode *prototype* yaitu *Deployment and Feedback,* dengan melakukan pengujian terhadap *prototype* yang telah dibangun. Pengujian dilakukan untuk memverifikasi apakah sistem *monitoring* mampu menjalankan seluruh fungsi secara terintegrasi, mulai dari proses akuisisi data, pengiriman data, penyimpanan data, hingga penyajian informasi pada *dashboard monitoring*. Selain itu, tahapan ini juga bertujuan memastikan setiap fungsi yang telah dirancang pada BAB IV dapat beroperasi sesuai dengan kebutuhan penelitian. Pengujian mencakup beberapa aspek utama, yaitu akuisisi data, komunikasi data, penyimpanan data, visualisasi *dashboard*, klasifikasi status *preventive maintenance*, serta kemampuan sistem dalam melakukan *monitoring* lebih dari satu *dies*. Metode yang digunakan adalah pengujian fungsional (*functional testing*), yaitu dengan memverifikasi dan membandingkan kesesuaian keluaran sistem terhadap fungsi yang telah ditentukan pada tahap perancangan.

5.1.1 Pengujian Akuisisi Data

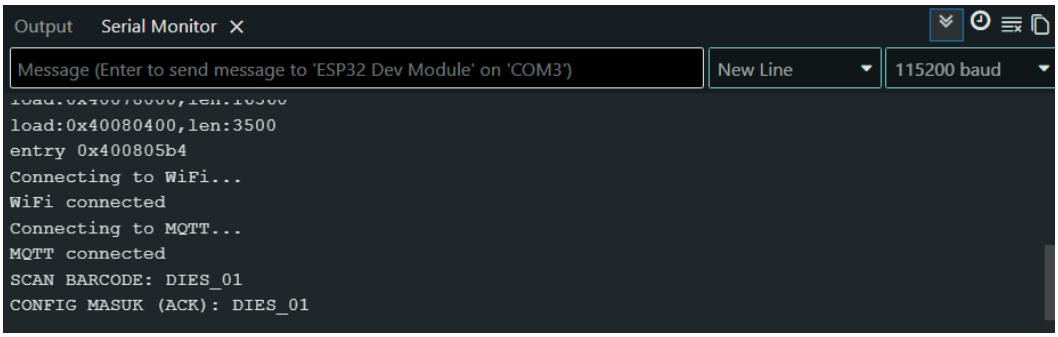
Pengujian akuisisi data bertujuan untuk memaverifikasi *barcode scanner* dan *limit switch* mampu menghasilkan data yang dapat digunakan oleh sistem *monitoring*. Pengujian dilakukan dengan melakukan pemindaian *barcode dies* dan memberikan *trigger* pada *limit switch* untuk mensimulasikan aktivitas *shot dies*.

1. *Scan barcode dies*

   Pengujian scan *barcode* dilakukan untuk memastikan sistem mampu membaca identitas *dies* yang digunakan pada proses produksi. Pengujian dilakukan dengan memindai *barcode dies* menggunakan *barcode scanner* yang terhubung dengan ESP32. Hasil pengujian dan pembacaan *barcode* ditampilkan dalam serial monitor sebagaimana ditunjukkan melalui Gambar 5.1 dan Gambar 5.2



Gambar 5. 1 Scan *Barcode*

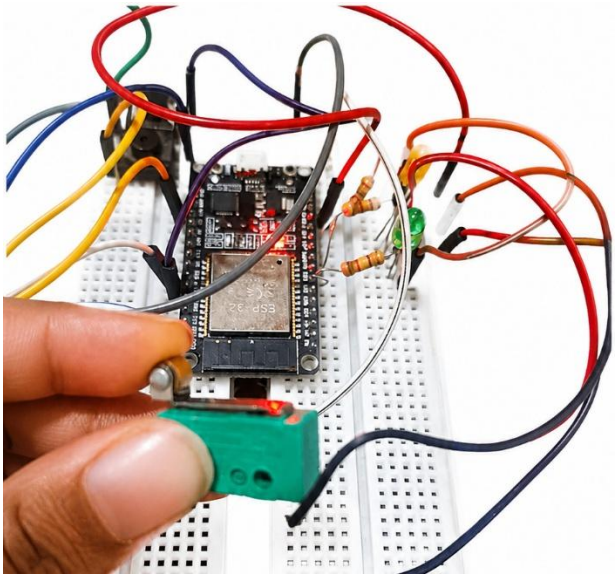


Gambar 5. 2 Hasil serial moni*to*r

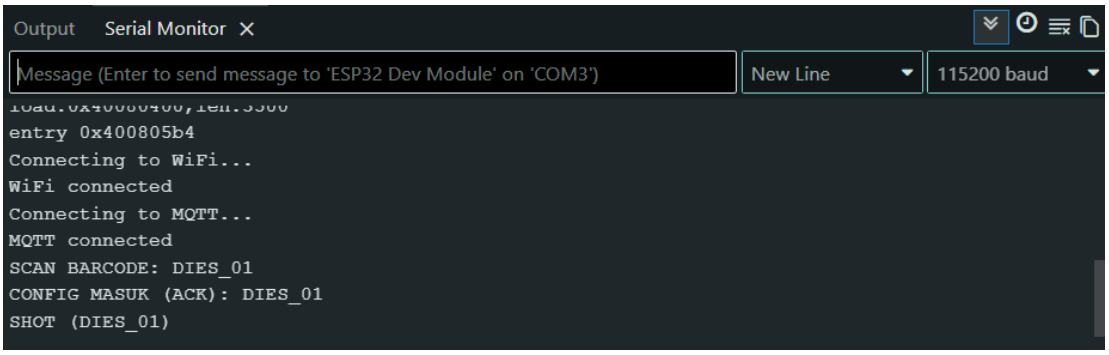
Berdasarkan hasil pengujian, *barcode scanner* berhasil membaca identitas *dies* dan mengirimkan data ke ESP32 yang telah terlihat dalam serial moni*to*r Arduino-IDE tanpa terjadi kesalahan pembacaan. Informasi identitas *dies* yang berhasil dibaca selanjutnya digunakan sebagai referensi *dies* aktif pada sistem *monitoring*. Hasil tersebut menunjukkan bahwa proses identifikasi *dies* telah berjalan sesuai dengan rancangan sistem yang dibangun.

2. *Trigger limit switch* untuk *shot dies*

   Pengujian *limit switch* dilakukan untuk memastikan sensor mampu mendeteksi aktivitas *shot dies* pada *prototype* yang dikembangkan. Pengujian dilakukan dengan memberikan satu kali *Trigger* pada *limit switch* untuk mensimulasikan satu siklus *shot dies*.



Gambar 5. 3 *Trigger limit switch*



Gambar 5. 4 Hasil serial monitor *shot*

Berdasarkan hasil pengujian yang ditampilkan pada serial monitor, setiap *Trigger* dari *limit switch* yang diberikan berhasil dideteksi oleh ESP32 dan dicatat sebagai penambahan jumlah *shot dies*. Nilai *shot* bertambah sesuai dengan jumlah *Trigger* yang dilakukan. Hasil tersebut menunjukkan bahwa *limit switch* mampu berfungsi sebagai perangkat akuisisi data untuk mendeteksi aktivitas *shot* pada sistem *monitoring* yang dikembangkan.

3. *Trigger limit switch* berulang

   Pengujian *Trigger* berulang dilakukan untuk mengevaluasi konsistensi pembacaan sensor selama proses *monitoring* berlangsung. Pengujian dilakukan dengan memberikan beberapa kali *Trigger* secara berurutan pada *limit switch* untuk mensimulasikan aktivitas produksi yang berlangsung secara kontinu.



Gambar 5. 5 Hasil *Trigger* berulang

Hasil pengujian pada Gambar 5.5 menunjukkan bahwa sistem mampu mencatat seluruh aktivitas *shot* yang diberikan tanpa kehilangan data maupun pembacaan ganda. Setiap *Trigger* menghasilkan penambahan nilai *shot* sesuai jumlah aktual yang dilakukan selama pengujian. Hasil tersebut menunjukkan bahwa mekanisme pembacaan sensor dan proses pencatatan data pada ESP32 dapat berjalan secara stabil untuk mendukung kebutuhan *monitoring dies* secara berkelanjutan.

Berdasarkan seluruh pengujian akuisisi data yang telah dilakukan, sistem berhasil memperoleh data *Operational* yang dibutuhkan untuk proses *monitoring dies*. *Barcode scanner* mampu mengidentifikasi *dies* yang digunakan dengan baik, sedangkan *limit switch* mampu mendeteksi aktivitas *shot dies* secara konsisten baik pada pengujian tunggal maupun pengujian berulang.

5.1.2 Pengujian Pengiriman dan Penyimpanan Data

Pengujian pengiriman dan penyimpanan data dilakukan untuk memverifikasi bahwa data hasil pembacaan *dies* dapat dikirim, diterima, diproses, dan disimpan pada sistem *monitoring*. Pengujian dilakukan dengan memberikan *Trigger* pada *limit switch* dan mengamati aliran data mulai dari ESP32 hingga basis data *monitoring*.

1. Pengiriman data ke ESP32

   Pengujian ini bertujuan untuk memastikan ESP32 mampu mengirimkan data *monitoring* menuju sistem pengolahan data menggunakan protokol MQTT melalui

jaringan WiFi. Pengujian dilakukan dengan memberikan *Trigger* pada *limit switch* sehingga menghasilkan perubahan data *monitoring* pada ESP32.



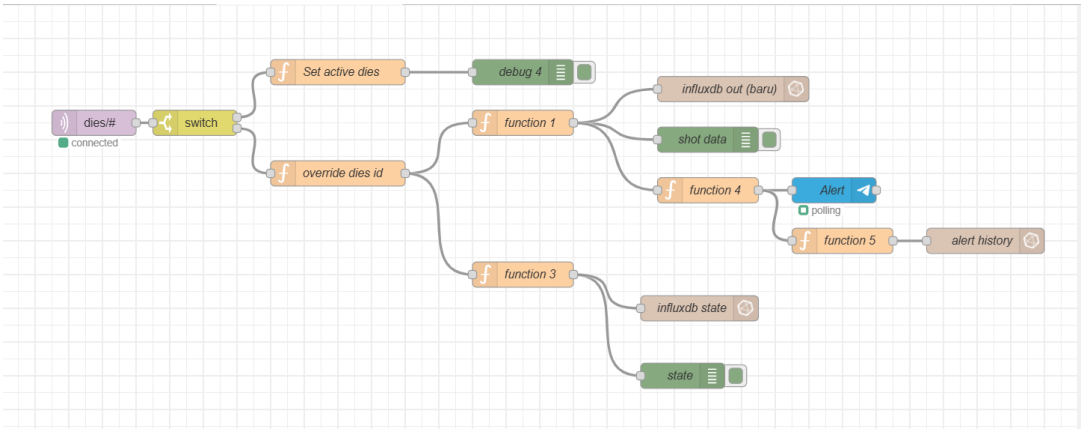
Gambar 5. 6 Hasil data ESP32

Berdasarkan hasil pengujian, data *monitoring* yang terdiri dari identitas *dies* dan informasi *shot* berhasil ditampilkan pada serial monitor. Hasil tersebut menunjukkan bahwa ESP32 mampu membaca data dari perangkat *input* dan melakukan proses publish data menuju MQTT *Broker* sesuai dengan topic yang telah ditentukan.

Keberhasilan proses pengiriman data menunjukkan bahwa perangkat akuisisi data telah mampu menjalankan fungsi komunikasi data secara baik sehingga data *monitoring* dapat diteruskan menuju sistem pengolahan data pada tahap berikutnya.

2. Penerimaan data di Node-RED

Pengujian penerimaan data dilakukan untuk memastikan data yang dikirim oleh ESP32 dapat diterima dan diproses oleh Node-RED. Pengujian dilakukan dengan mengamati status koneksi MQTT serta informasi yang ditampilkan pada panel debug Node-RED.

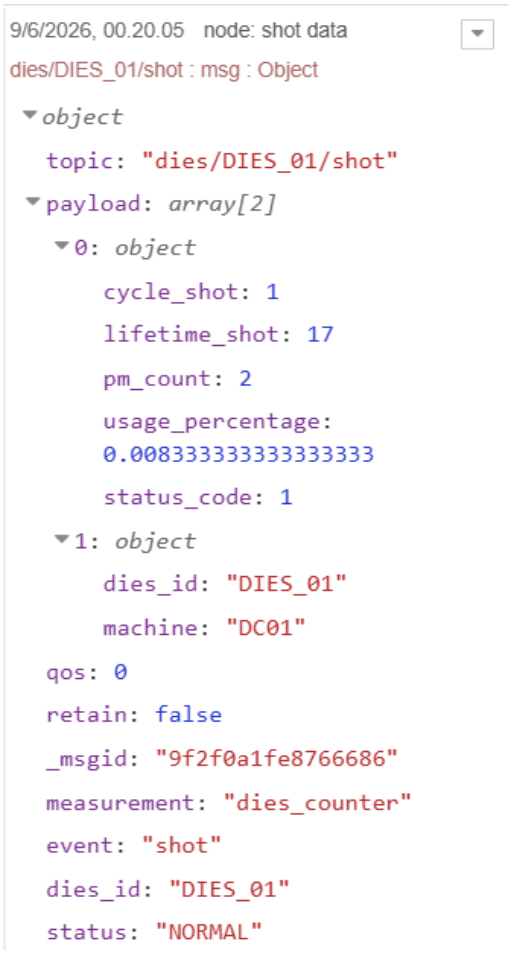


Gambar 5. 7 Node-RED *Connected*



Gambar 5. 8 Informasi *active dies*

Berdasarkan hasil pengujian pada Gambar 5.7, Node-RED yang telah disambungkan dengan beberapa *flow* berhasil terhubung dengan MQTT *Broker* dengan ditandai *Connected* sehingga komunikasi data dapat berlangsung secara normal. Selanjutnya, hasil pengujian pada Gambar 5.8 menunjukkan bahwa sistem mampu mengidentifikasi *dies* aktif berdasarkan data *barcode* yang diterima. Informasi identitas *dies* tersebut digunakan sebagai referensi pada proses *monitoring*.

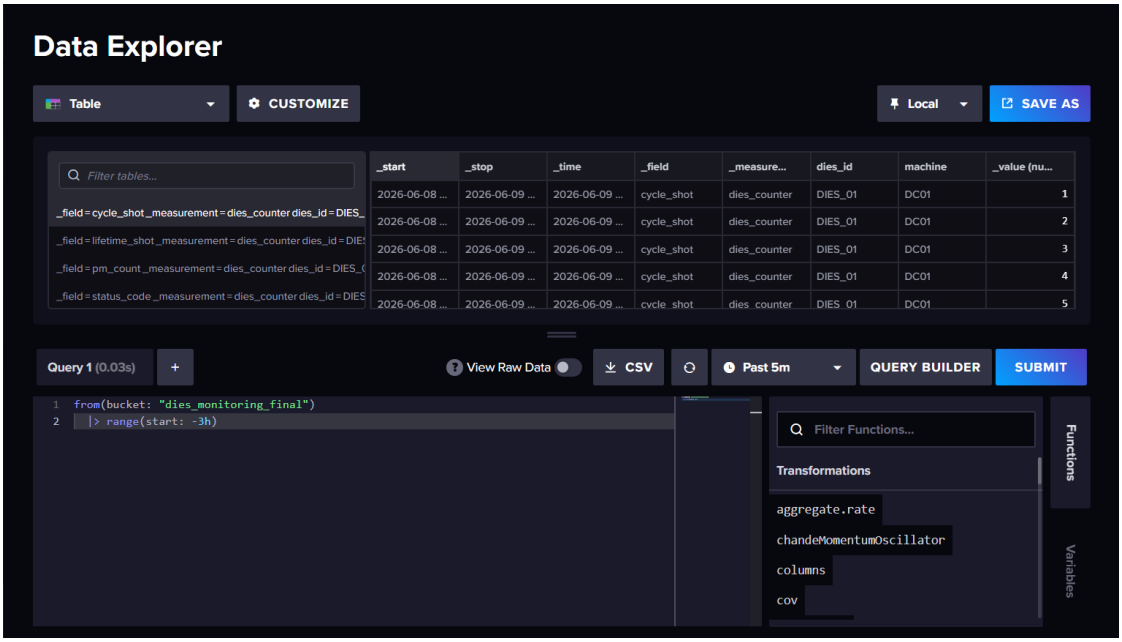


Gambar 5. 9 Informasi data masuk *shot dies*

Selain itu, hasil pengujian pada Gambar 5.9 menunjukkan bahwa data aktivitas *shot* yang dikirim oleh ESP32 berhasil diterima dan ditampilkan pada panel debug Node-RED. Data tersebut selanjutnya diproses untuk menghasilkan parameter *monitoring* seperti *cycle shot* sebanyak 1, *lifetime shot* sebanyak 17, *usage percentage* sebesar 0.008%, status *Operational dies* ditandai dengan nomor 1 yaitu NORMAL, serta informasi *preventive maintenance* yang digunakan pada *dahboard monitoring*.

3. Penyimpanan data di InfluxDB

Pengujian penyimpanan data dilakukan untuk memastikan data hasil pengolahan yang berasal dari Node-RED dapat tersimpan pada basis data InfluxDB. Pengujian dilakukan dengan mengamati data *monitoring* yang tersimpan pada *Data Explorer* InfluxDB.



Gambar 5. 10 Data Explorer InfluxDB

Berdasarkan hasil pengujian pada gambar 5.10, data *monitoring* berhasil tersimpan pada InfluxDB dalam bentuk data *time-series* dengan nama *database* "*dies_monitoring*_final". Data yang tersimpan meliputi identitas *dies*, *cycle shot*, *lifetime shot*, *usage percentage*, status *preventive maintenance*, serta informasi *monitoring* lainnya yang dibutuhkan oleh sistem yang dimana dari data yang tercantum terlihat bahwa dalam fields *cycle shot* menampilkan informasi seperti *time*, *dies*_id yaitu *DIES*_01, *machine* yaitu DC01 serta value sebanyak 5 kali *shot*.

Hasil tersebut menunjukkan bahwa proses *load* pada mekanisme ETL telah berjalan dengan baik sehingga seluruh data *monitoring* dapat disimpan sebagai data *history*s. Data *history*s tersebut selanjutnya digunakan sebagai sumber data utama pada *dahboard* Grafana untuk kebutuhan *monitoring* dan analisis kondisi *dies*.

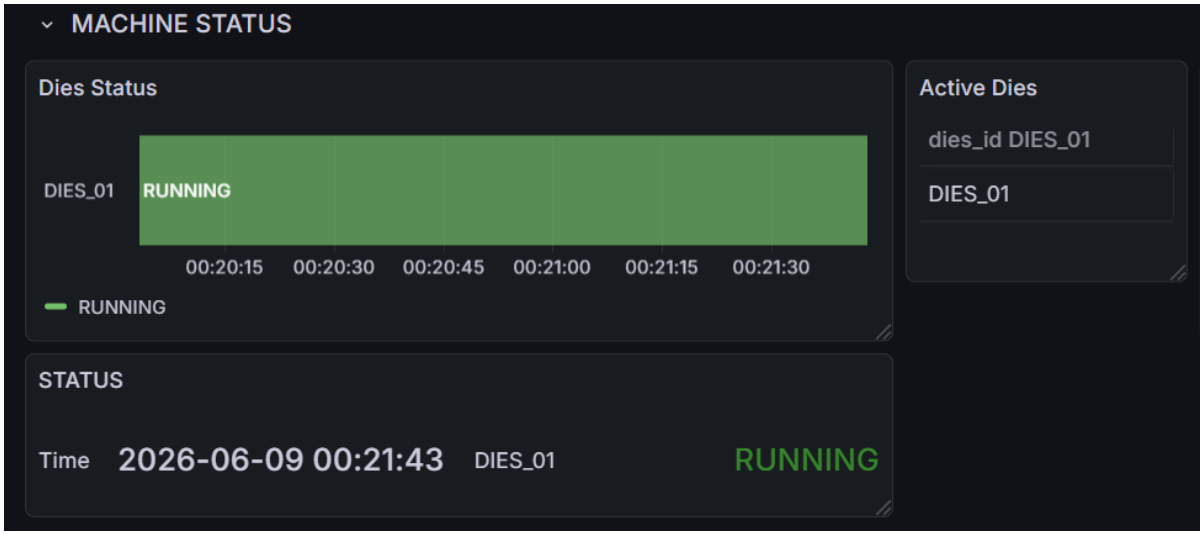
5.1.3 Pengujian *Dahboard Monitoring*

Pengujian *dahboard monitoring* dilakukan untuk memastikan seluruh informasi *monitoring* yang tersimpan pada InfluxDB dapat ditampilkan dengan benar pada *dahboard* Grafana. Pengujian dilakukan dengan membandingkan data yang tersimpan pada basis data dengan informasi yang ditampilkan pada *dahboard monitoring* secara *real-time*.

Dahboard yang diuji terdiri dari *dahboard* status *dies*, *dahboard* produksi, *dahboard preventive maintenance*, serta *dahboard history monitoring* dan *alert*. Pengujian dilakukan

untuk memastikan setiap panel mampu menampilkan informasi sesuai fungsi yang telah dirancang pada sistem *monitoring*.

1. Pengujian *Dahboard* Status *Dies*



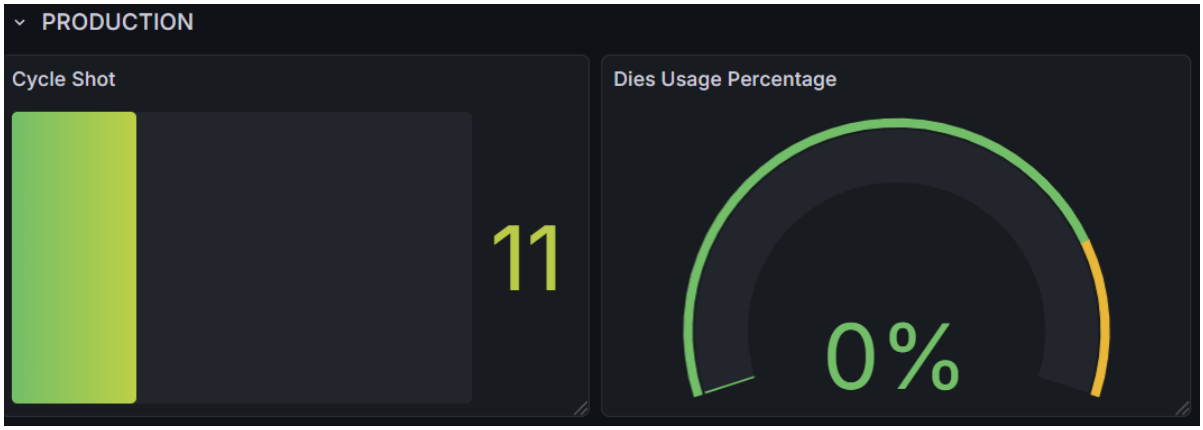
Gambar 5. 11 *Monitoring* Status *Dies*

Berdasarkan Gambar 5.11, *dahboard* berhasil menampilkan informasi status *Operational dies* secara *real-time*. Pada panel *Dies* Status, sistem menunjukkan bahwa *DIES*_01 berada pada kondisi *RUNNING* selama proses pengujian berlangsung. Status tersebut diperoleh dari hasil pembacaan data *monitoring* yang dikirimkan oleh ESP32 melalui MQTT dan diproses oleh Node-RED sebelum disimpan ke dalam InfluxDB.

Selanjutnya, panel *Active Dies* menampilkan identitas *dies* aktif yaitu *DIES*_01. Informasi ini menunjukkan bahwa proses pemilihan *dies* menggunakan *barcode scanner* berhasil diterima oleh sistem dan digunakan sebagai referensi *monitoring* selama proses produksi berlangsung.

Dan yang terakhir ada panel Status, *dahboard* menampilkan waktu pembaruan terakhir yaitu 2026-06-09 00:21:43 dengan kondisi *RUNNING*. Informasi tersebut menunjukkan bahwa data *monitoring* masih diterima dan diproses oleh sistem secara normal sehingga operator dapat mengetahui kondisi *dies* yang sedang digunakan secara langsung. Berdasarkan hasil pengujian, *dashboard* berhasil menyajikan informasi identitas *dies* aktif dan status operasionalnya secara *real-time* sesuai dengan kondisi aktual pada saat pengujian dilakukan.

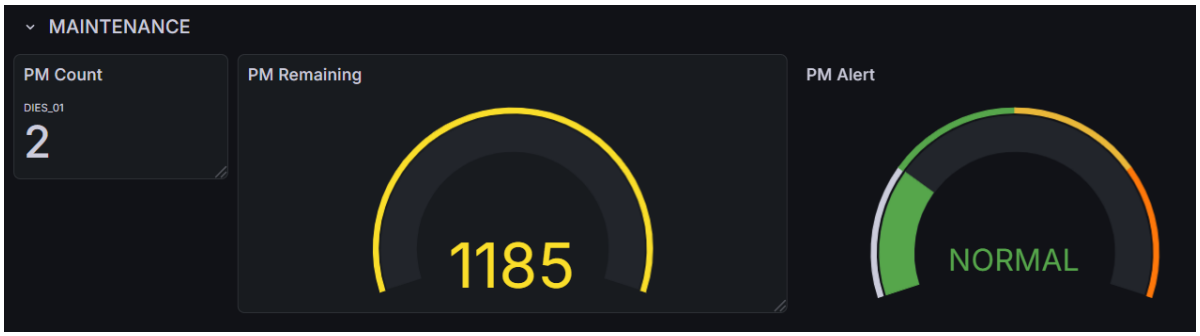
2. Pengujian *Dahboard* Production *Dies*.



Gambar 5. 12 *Monitroing Production Dies*

Berdasarkan Gambar 5.12, *dahboard* berhasil menampilkan informasi jumlah *cycle shot* dan persentase penggunaan *dies* selama proses produksi berlangsung. Pada panel *Cycle shot*, nilai yang ditampilkan adalah 11 *shot*. Nilai tersebut menunjukkan bahwa *limit switch* telah mendeteksi sebanyak sebelas aktivitas *shot* dan seluruh data berhasil dikirim ke sistem *monitoring*. Selanjutnya, pada panel *Dies Usage percentage*, *dahboard* menampilkan nilai penggunaan *dies* sebesar 0%. Nilai ini menunjukkan bahwa penggunaan *dies* masih berada jauh di bawah batas *preventive maintenance* yang telah ditetapkan sehingga kondisi *dies* masih berada pada kategori aman untuk digunakan. Data *cycle shot* dan persentase penggunaan tersebut diperbarui secara otomatis setiap kali terjadi aktivitas *shot* pada *dies*. Dengan demikian operator dapat memantau tingkat penggunaan *dies* tanpa melakukan pencatatan secara manual.

3. Pengujian *Dahboard Preventive maintenance*



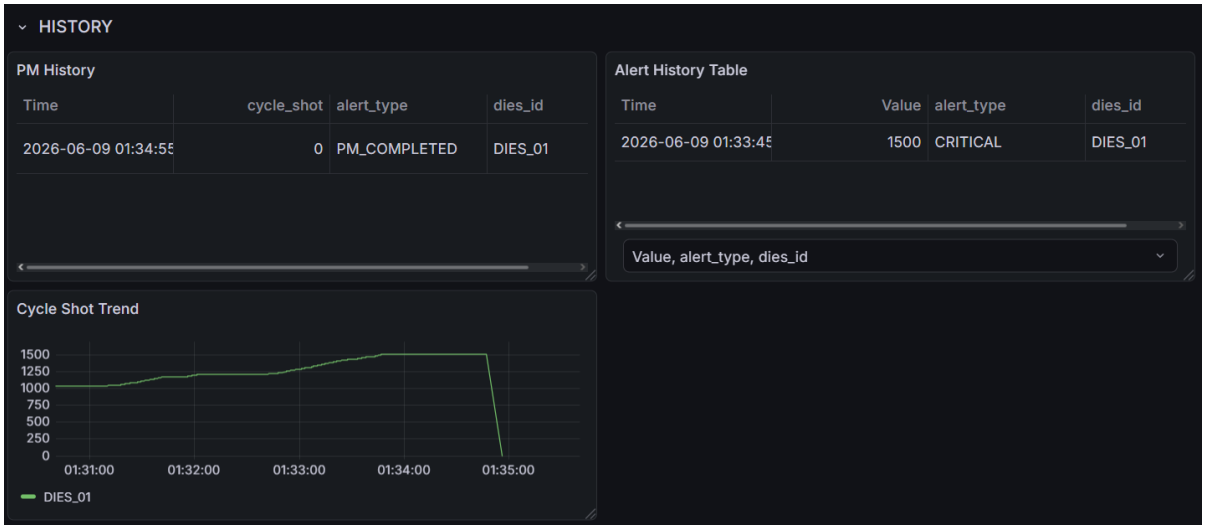
Gambar 5. 13 *Monitoring* PM

Berdasarkan Gambar 5.13, *dahboard* berhasil menampilkan informasi terkait aktivitas *preventive maintenance* yang dilakukan pada *dies*. Pada panel PM *Count*, sistem menunjukkan nilai 2, yang berarti *preventive maintenance* telah dilakukan sebanyak dua kali selama proses pengujian. Nilai tersebut akan bertambah secara otomatis setiap kali operator menjalankan

fungsi reset *preventive maintenance* melalui sistem. Selanjutnya, pada panel PM *Remaining*, *dahboard* menampilkan nilai 1185. Nilai tersebut menunjukkan jumlah *cycle shot* yang masih tersisa sebelum *dies* mencapai target *preventive maintenance* berikutnya. Informasi ini membantu operator dalam menentukan waktu pelaksanaan *maintenance* secara lebih terencana.

Kemudian panel PM *Alert* merupakan sistem yang menampilkan status NORMAL, *WARNING*, *DUE*, dan *CRITICAL*. Data yang ditampilkan menunjukkan status NORMAL, kondisi tersebut menunjukkan bahwa jumlah *cycle shot* yang tercatat masih berada di bawah batas *warning* maupun batas *preventive maintenance* yang telah ditentukan. Hasil uji *dashboard* mampu menyajikan bahwa *dahboard* mampu memberikan informasi kondisi *preventive maintenance* secara *real-time* sehingga pengguna dapat memanfaatkan informasi tersebut sebagai dasar dalam menentukan tindakan pemeliharaan *dies*.

4. Pengujian *Dahboard History* dan *Alert*



Gambar 5. 14 *Monitoring History* dan *Alert*

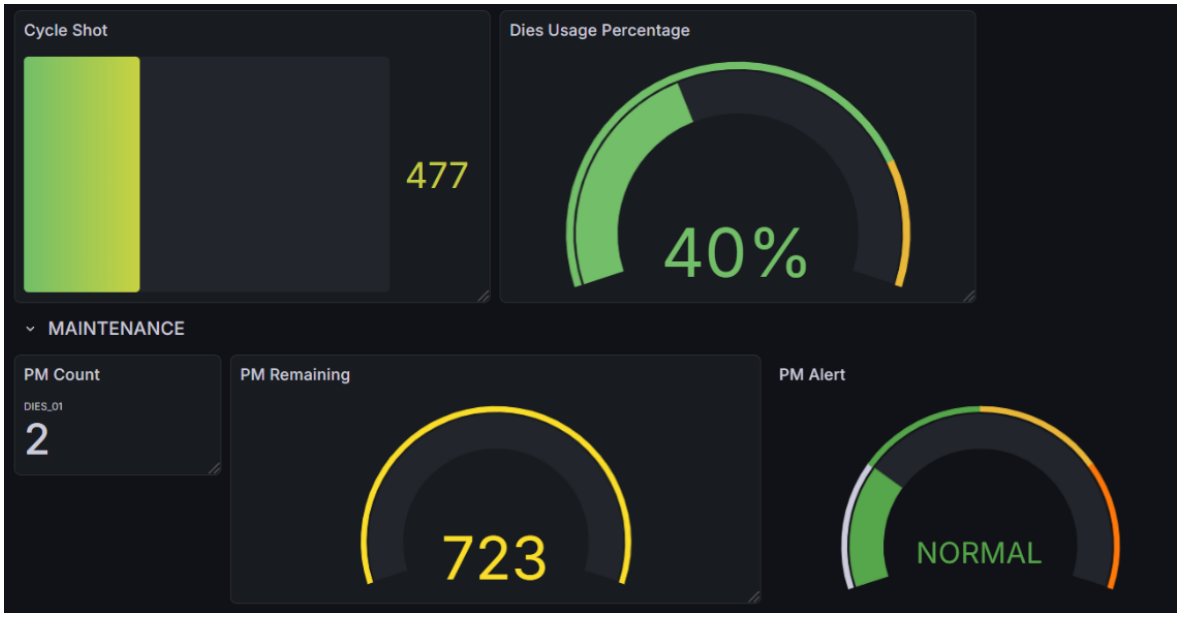
Berdasarkan Gambar 5.14, *dahboard* berhasil menampilkan *history* aktivitas *maintenance* dan *history alert* yang terjadi selama proses *monitoring*. Pada panel PM *History*, tercatat aktivitas *preventive maintenance* yang dilakukan pada *DIES*_01 dengan jenis aktivitas PM_*COMPLETED* dan nilai *cycle shot* kembali menjadi 0 setelah proses *maintenance* selesai dilakukan. Data tersebut tersimpan secara otomatis ke dalam *database* sebagai *history maintenance*.

Selanjutnya, pada panel *Alert History Table*, sistem menampilkan *history alert* dengan kategori *CRITICAL* pada nilai *cycle shot* 1500. Informasi ini menunjukkan bahwa sistem berhasil mendeteksi ketika jumlah *cycle shot* telah mencapai batas maksimum yang ditentukan

dan secara otomatis menghasilkan notifikasi peringatan kepada operator. Terakhir, terdapat panel *Cycle shot Trend* menampilkan grafik perubahan jumlah *cycle shot* terhadap waktu. Grafik menunjukkan bahwa jumlah *shot* meningkat secara bertahap hingga mencapai sekitar 1500 *shot*, kemudian kembali ke nilai 0 setelah dilakukan *preventive maintenance*. Perubahan tersebut sesuai dengan kondisi aktual pengujian dan menunjukkan bahwa proses reset *cycle shot* berjalan dengan baik.

5.1.4 Pengujian *Alert Preventive Maintenance*

Pengujian *preventive maintenance* dan *alert maintenance* bertujuan untuk mengevaluasi kemampuan sistem dalam mengklasifikasikan kondisi *dies* berdasarkan jumlah penggunaan yang tercatat. ada pengujian ini dilakukan simulasi peningkatan *cycle shot* hingga mencapai batas *warning*, *due maintenance*, dan *critical*. Selain itu, pengujian juga dilakukan untuk memverifikasi kemampuan sistem dalam menghasilkan notifikasi otomatis melalui Telegram, menampilkan status *dies* pada *dahboard* Grafana, serta melakukan reset *cycle shot* setelah *preventive maintenance* selesai dilakukan.



Gambar 5. 15 Kondisi Normal

Berdasarkan Gambar 5.15, sistem menunjukkan kondisi *dies* masih berada pada status NORMAL. Nilai *cycle shot* yang tercatat sebesar 477 *shot* dengan persentase penggunaan *dies* sebesar 40%. Nilai PM *Remaining* menunjukkan angka 723, yang berarti *dies* masih dapat digunakan sebanyak 723 *shot* sebelum mencapai target *preventive maintenance*. Pada kondisi

ini sistem belum menghasilkan notifikasi peringatan karena penggunaan *dies* masih berada di bawah batas *warning* yang telah ditentukan.



Gambar 5. 16 Kondisi *Warning*

Selanjutnya pada Gambar 5.16, jumlah *cycle shot* meningkat menjadi 1035 *shot* dengan persentase penggunaan sebesar 86%. Nilai PM *Remaining* tersisa 165 *shot*. Pada kondisi ini sistem secara otomatis mengubah status *dies* menjadi *WARNING* sebagai indikasi bahwa *dies* telah mendekati batas *preventive maintenance*. Perubahan status ditampilkan pada *dahboard* Grafana dan menjadi dasar untuk menghasilkan notifikasi peringatan kepada operator.



Gambar 5. 17 Kondisi *Due Maintenance*

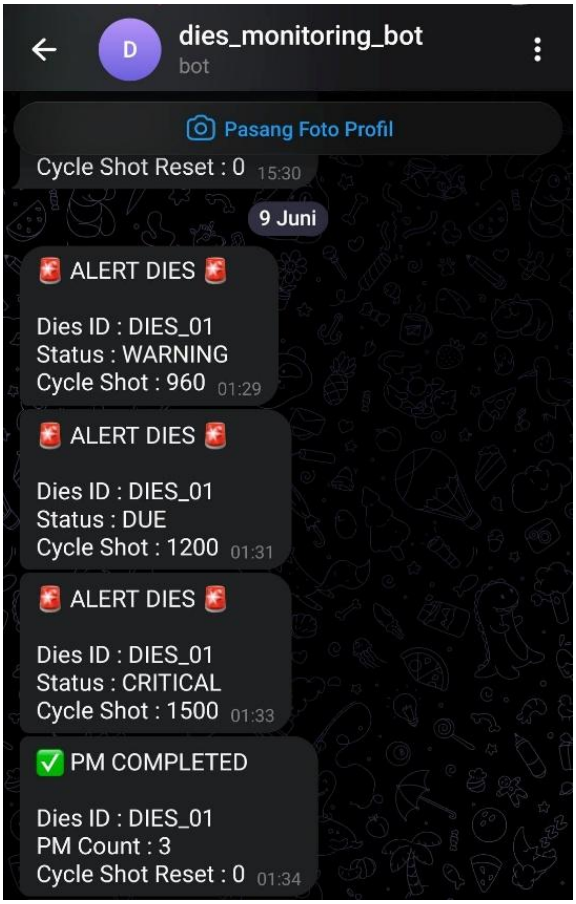
Pada Gambar 5.17, jumlah *cycle shot* mencapai 1205 *shot* sehingga melewati target *preventive maintenance* yang ditetapkan sebesar 1200 *shot*. *Dahboard* menampilkan persentase penggunaan sebesar 100% dan nilai PM *Remaining* sebesar -5. Sistem secara otomatis

mengubah status *dies* menjadi *DUE* yang menunjukkan bahwa *preventive maintenance* sudah harus dilakukan untuk mencegah potensi kerusakan *dies* akibat penggunaan berlebih.



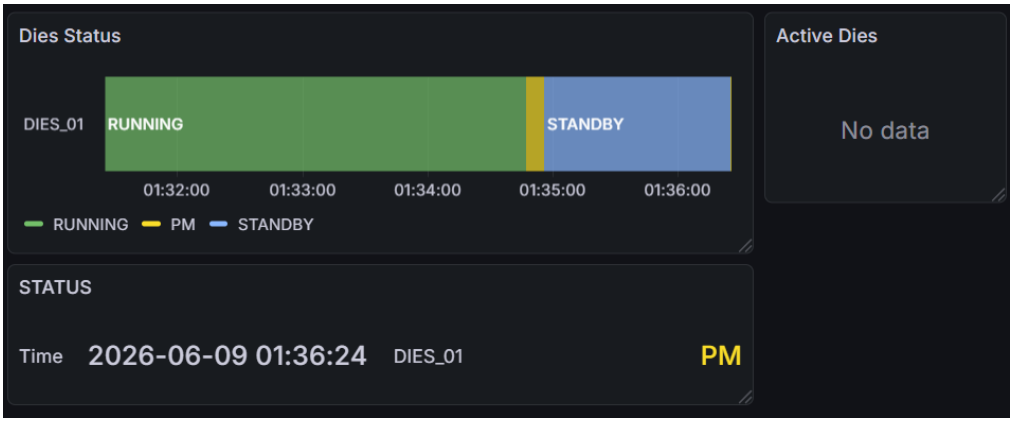
Gambar 5. 18 Kondisi *Critical*

Kemudian Gambar 5.18, jumlah *cycle shot* mencapai 1508 *shot* sehingga telah melampaui batas maksimum penggunaan *dies*. Nilai PM *Remaining* menunjukkan angka -308 dan status *dies* berubah menjadi *CRITICAL*. Kondisi ini menunjukkan bahwa *dies* telah digunakan melebihi batas aman yang ditetapkan sehingga berpotensi meningkatkan risiko kerusakan *dies* apabila tetap digunakan dalam proses produksi.

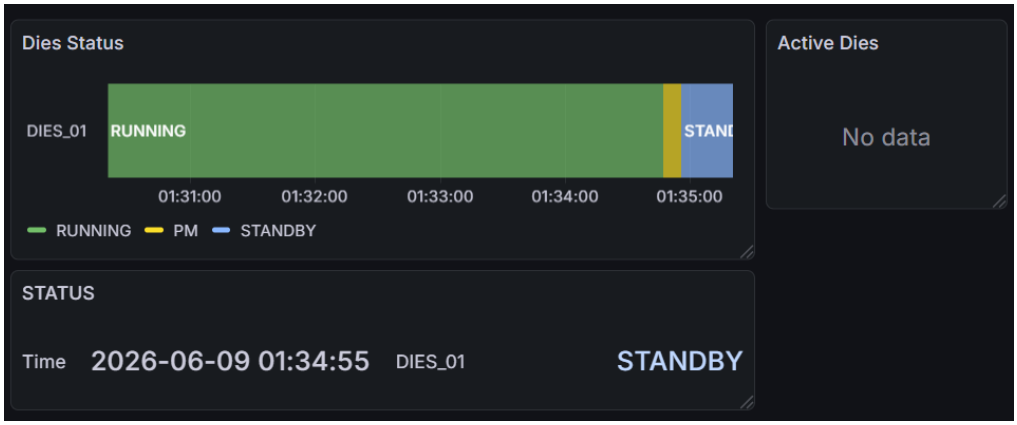


Gambar 5. 19 Notifikasi Telegram *Alert Maintenance*

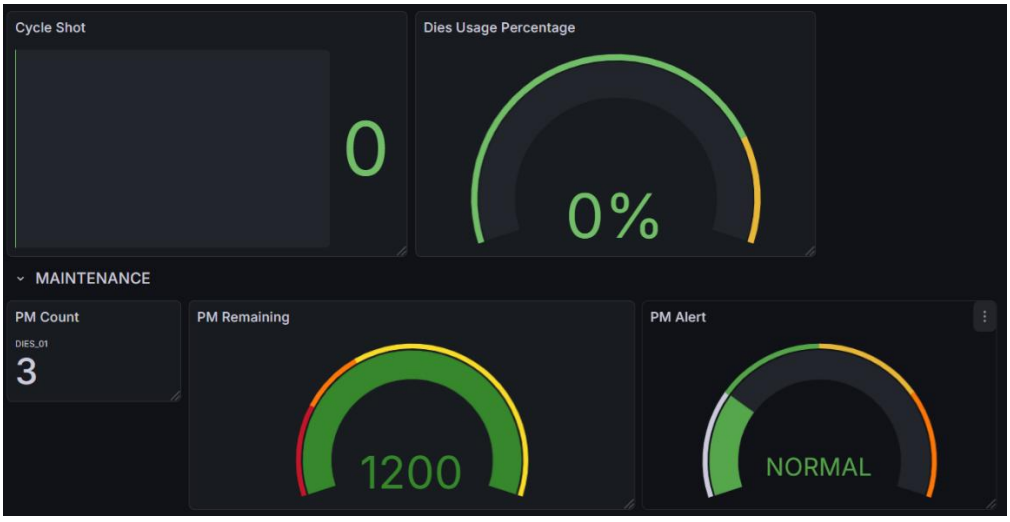
Visualisasi pada Gambar 5.19, menunjukkan mekanisme pengiriman notifikasi secara otomatis melalui Telegram setiap kali status *dies* berubah menjadi *WARNING*, *DUE*, dan *CRITICAL*. Informasi yang dikirim meliputi identitas *dies*, status *maintenance*, dan jumlah *cycle shot* saat notifikasi dihasilkan. Pada kondisi *WARNING*, sistem mengirimkan notifikasi ketika *cycle shot* mencapai 960. Selanjutnya notifikasi *DUE* dikirim saat *cycle shot* mencapai 1200, sedangkan notifikasi *CRITICAL* dikirim ketika *cycle shot* mencapai 1500. Hasil pengujian menunjukkan bahwa mekanisme *alert* mampu berjalan secara otomatis dan memberikan informasi kepada operator secara *real-time*.



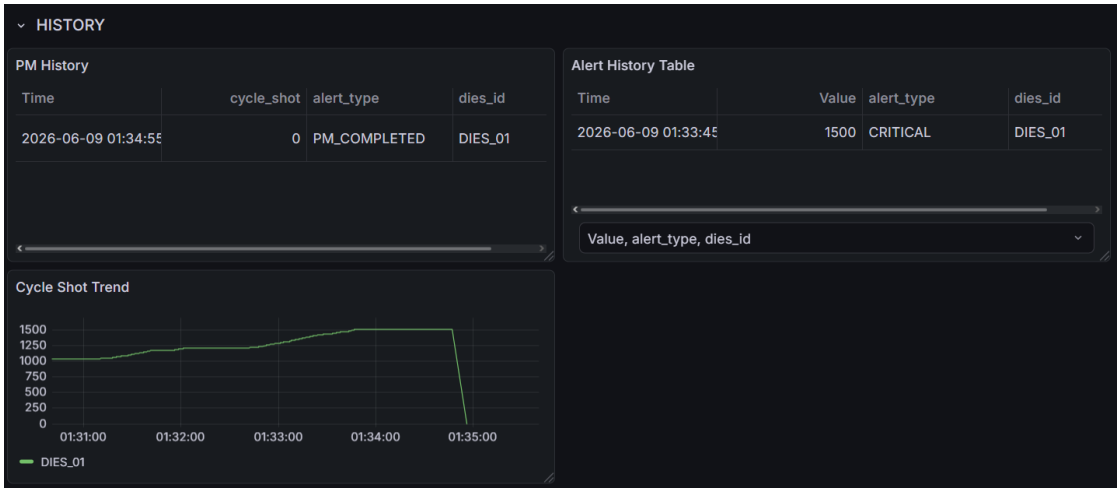
Gambar 5. 20 Status *Dies* Saat *Maintenance*



Gambar 5. 21 Status *Dies* Setelah *Maintenance*



Gambar 5. 22 *Dahboard* Setelah PM Selesai



Gambar 5. 23 *History Maintenance* dan *Alert*

Pada Gambar 5.20 hingga Gambar 5.23 menunjukkan status *dies* saat dilakukan pm sampai setelah melakukan pm, yang dimana operator melakukan *preventive maintenance* setelah menerima notifikasi dari sistem. Saat *maintenance* berlangsung, status *dies* berubah menjadi PM dan ditampilkan pada *dahboard monitoring*. Perubahan status tersebut menunjukkan bahwa *dies* sedang berada dalam proses perawatan sehingga aktivitas produksi dihentikan sementara. Setelah *preventive maintenance* selesai dilakukan, operator menjalankan fungsi PM Done sehingga sistem melakukan reset *cycle shot* menjadi 0 dan menambah nilai PM *Count* dari 2 menjadi 3. Pada saat yang sama status *dies* berubah menjadi *STANDBY* sebagai tanda bahwa *dies* siap digunakan kembali dalam proses produksi.

Tabel 5. 1 Hasil Pengujian Status PM

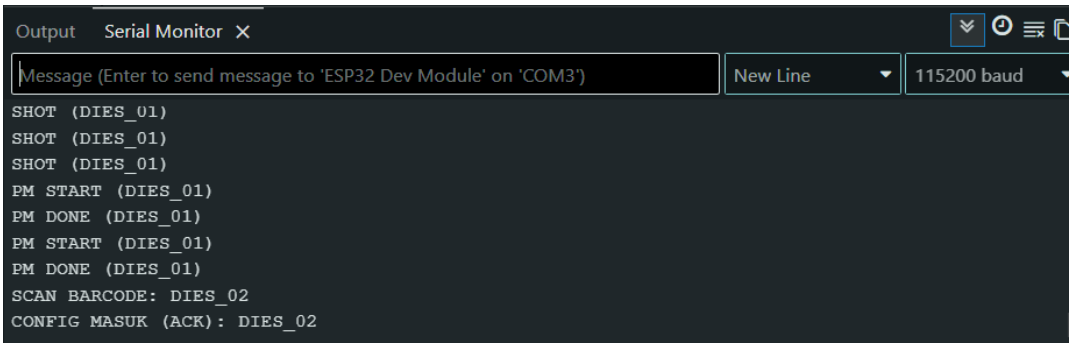
Cycle shot	**Usage (%)**	**PM *Remaining***	**Status**
477	40	723	NORMAL
1035	86	165	*WARNING*
1205	100	-5	*DUE*
1508	100	-308	*CRITICAL*

Berdasarkan Tabel 5.1, sistem berhasil mengklasifikasikan kondisi *dies* menjadi empat kategori yaitu NORMAL, *WARNING*, *DUE*, dan *CRITICAL* berdasarkan jumlah *cycle shot* yang tercatat. Perubahan status terjadi secara otomatis sesuai dengan batas *preventive maintenance* yang telah ditentukan sehingga operator dapat mengetahui kondisi *dies* secara *real-time*.

5.1.5 Pengujian *Monitoring Multi Dies*

Pengujian *monitoring multi-dies* dilakukan untuk memastikan sistem mampu mengidentifikasi, menyimpan, dan menampilkan data dari lebih dari satu *dies* secara bersamaan. Pada pengujian ini digunakan dua *dies*, yaitu *DIES*_01 dan *DIES*_02. Setiap *dies* memiliki identitas, jumlah *cycle shot*, status operasi, dan data *maintenance* yang tersimpan secara terpisah. Hasil pengujian diamati melalui Serial Monitor, Node-RED, serta *dahboard* Grafana.

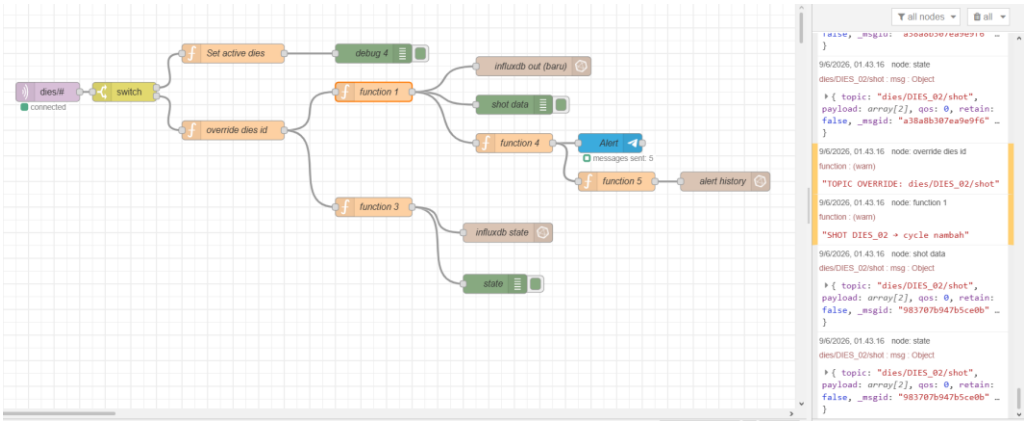
1. Pergantian *Dies* Dengan Scan *Barcode*



Gambar 5. 24 Pergantian *Dies*

Berdasarkan Gambar 5.24, proses pergantian *dies* diawali dengan pemindaian *barcode* pada *dies* yang akan digunakan. Hasil pembacaan *barcode* ditampilkan pada Serial Monitor dengan informasi SCAN *BARCODE*: *DIES*_02. Setelah *barcode* berhasil dibaca, ESP32 mengirimkan data konfigurasi ke sistem dan memperoleh konfirmasi berupa CONFIG MASUK (ACK): *DIES*_02. Hasil tersebut menunjukkan bahwa identitas *dies* berhasil diperbarui dari *DIES*_01 menjadi *DIES*_02 sehingga seluruh data *monitoring* berikutnya akan dikaitkan dengan *dies* yang aktif.

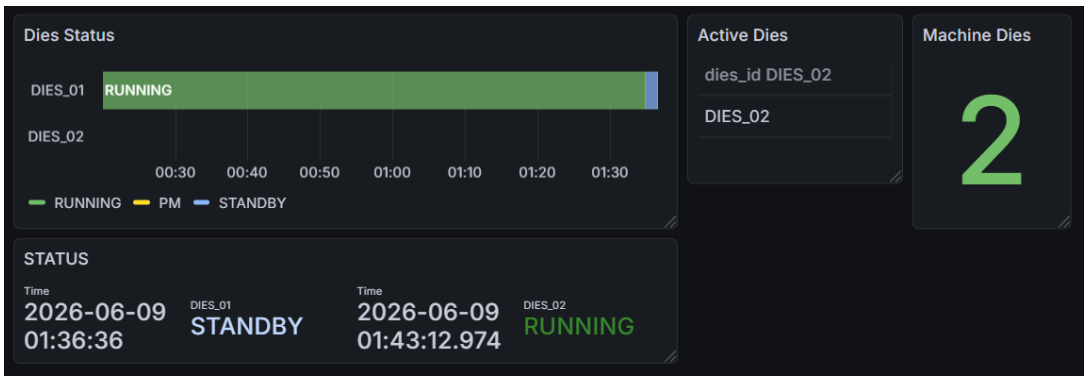
2. Pemrosesan Data *Multi-Dies*



Gambar 5. 25 Proses *Multi dies* pada Node-RED

Berdasarkan Gambar 5.25, Node-RED berhasil menerima data dari MQTT dan melakukan proses override identitas *dies* sesuai hasil pemindaian *barcode*. Pada debug panel terlihat pesan *TO*PIC OVERRIDE: *dies*/*DIES*_02/*shot* yang menunjukkan bahwa setiap sinyal *shot* yang diterima sistem telah dikaitkan dengan *DIES*_02. Selain itu muncul pesan *SHOT DIES*_02 → *cycle* nambah yang menandakan bahwa proses pencatatan *cycle shot* dilakukan pada *dies* yang sesuai. Hasil ini menunjukkan bahwa mekanisme identifikasi *dies* dan pemisahan data *monitoring* berjalan dengan baik.

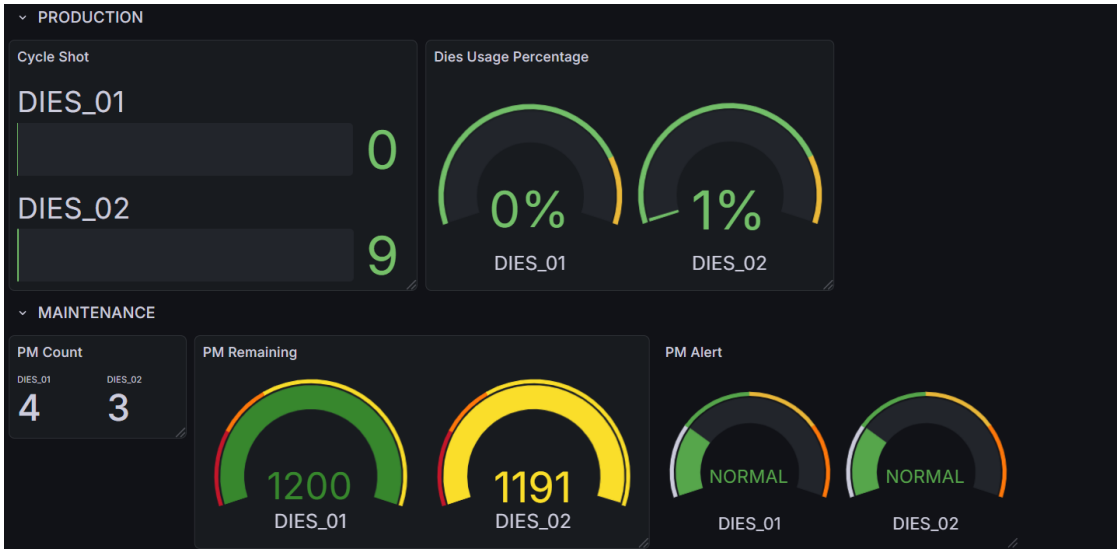
3. Status Operasi *Multi-Dies*



Gambar 5. 26 Status *Multi Dies*

Berdasarkan Gambar 5.26, *dahboard Machine* Status menampilkan status masing-masing *dies* secara terpisah. *DIES*_01 berada pada kondisi *STANDBY* dengan waktu pembaruan terakhir pukul 01:36:36, sedangkan *DIES*_02 berada pada kondisi *RUNNING* dengan waktu pembaruan terakhir pukul 01:43:12. Panel *Active Dies* menunjukkan bahwa *dies* yang sedang aktif digunakan adalah *DIES*_02. Selain itu, panel *Machine Dies* menampilkan nilai 2 yang menunjukkan bahwa sistem berhasil mengenali dua identitas *dies* yang tersimpan pada *database monitoring*. Hasil tersebut menunjukkan bahwa sistem mampu membedakan status operasi setiap *dies* secara independen sehingga aktivitas masing-masing *dies* dapat dipantau secara *real-time*.

4. *Monitoring* Production dan *Maintenance Multi-Dies*



Gambar 5. 27 Hasil *Monitoring Multi-Dies*

Berdasarkan Gambar 5.27, *dahboard* menampilkan data produksi dan *maintenance* dari dua *dies* secara bersamaan. Pada panel *Cycle shot*, *DIES*_01 memiliki nilai *cycle shot* sebesar 0, sedangkan *DIES*_02 memiliki nilai *cycle shot* sebesar 9. Nilai tersebut menunjukkan bahwa aktivitas produksi saat pengujian hanya terjadi pada *DIES*_02. Selanjutnya, dilanjutkan pada panel *Dies Usage percentage*, *DIES*_01 menunjukkan tingkat penggunaan sebesar 0%, sedangkan *DIES*_02 menunjukkan tingkat penggunaan sebesar 1%. Persentase tersebut dihitung berdasarkan jumlah *cycle shot* terhadap target *preventive maintenance* yang telah ditetapkan.

Kemudian, pada panel PM *Count* terlihat bahwa *DIES*_01 telah menjalani *preventive maintenance* sebanyak 4 kali, sedangkan *DIES*_02 telah menjalani *preventive maintenance* sebanyak 3 kali. Informasi ini menunjukkan bahwa *history maintenance* setiap *dies* tersimpan secara terpisah dalam *database*. Serta panel PM *Remaining* menunjukkan nilai 1200 untuk *DIES*_01 dan 1191 untuk *DIES*_02. Nilai tersebut menunjukkan sisa jumlah *shot* yang dapat digunakan sebelum mencapai target *preventive maintenance* berikutnya. Sementara itu, panel PM *Alert* menunjukkan status NORMAL pada kedua *dies* karena jumlah penggunaan masih berada jauh di bawah batas *warning*.

Seluruh rangkaian pengujian menunjukkan bahwa sistem berhasil mengindentifikasi setiap *dies* secara terpisah melalui proses pembacaan *barcode* dan pengolahan data pada Node-RED. Setiap data *cycle shot*, status operasi, *history preventive maintenance*, dan status *alert* tersimpan secara terpisah berdasarkan identitas *dies* yang aktif. Selain itu, *dahboard* Grafana

mampu menampilkan informasi *monitoring* dari lebih dari satu *dies* secara bersamaan tanpa terjadi pencampuran data antar *dies*. Berikut merupakan hasil keseluruhan pengujian *monitoring multi-dies*.

Tabel 5. 2 Hasil Pengujian *Multi-Dies*

Parameter	***DIES*_01**	***DIES*_02**
Status Operasi	*STANDBY*	*RUNNING*
Cycle shot	0	9
Usage percentage	0%	1%
PM *Count*	4	3
PM *Remaining*	1200	1191
Status *Maintenance*	NORMAL	NORMAL

Berdasarkan Tabel 5.2, sistem berhasil menyimpan dan menampilkan data *monitoring* untuk masing-masing *dies* secara terpisah. Perbedaan nilai *cycle shot*, status operasi, *history maintenance*, serta sisa *preventive maintenance* menunjukkan bahwa mekanisme identifikasi *dies* menggunakan *barcode* dan proses pengolahan data pada Node-RED telah berjalan dengan baik. Hasil pengujian ini membuktikan bahwa sistem mampu mendukung *monitoring* lebih dari satu *dies* dalam satu *dahboard* secara *real-time*.

5.1.6 Pengujian *Latency* Sistem *Monitoring*

Pengujian *latency* dilakukan untuk mengetahui waktu respon sistem *monitoring* sejak aktivitas *shot* terdeteksi oleh *limit switch* hingga informasi ditampilkan pada *dahboard* Grafana. Pengujian dilakukan dengan metode observasi menggunakan *stopwatch* untuk mengukur selang waktu antara simulasi *shot* dan perubahan nilai *monitoring* pada *dahboard*. Pengukuran dilakukan sebanyak lima kali percobaan untuk memperoleh gambaran waktu respon sistem secara keseluruhan.

Tabel 5. 3 Uji *Latency*

Trial	***Delay* (detik)**
1	02.05
2	03.19
3	03.11
4	02.97
5	03.83
Rata-rata	**03.03**

Berdasarkan hasil pengujian yang disajikan Tabel 5.3, diperoleh rata-rata waktu respons sistem sebesar 3,03 detik. Waktu tersebut merupakan waktu yang dibutuhkan data untuk melalui seluruh proses pada sistem, mulai dari pembacaan oleh ESP32, pengiriman melalui MQTT, pengolahan pada Node-RED, penyimpanan ke InfluxDB, hingga ditampilkan pada *dahboard* Grafana. Perubahan data hasil aktivitas *shot* dapat ditampilkan oleh sistem dalam waktu yang cukup cepat setelah aktivitas *shot* terjadi.

Berdasarkan hasil pengumpulan data selama penelitian, waktu yang dibutuhkan untuk satu siklus produksi berada pada rentang 55–75 detik per *shot*. Oleh karena itu, waktu respon sistem sebesar 3,03 detik masih jauh lebih kecil dibandingkan waktu satu siklus produksi sehingga informasi yang ditampilkan pada *dahboard* tetap dapat digunakan untuk mendukung proses *monitoring* dan penentuan *preventive maintenance*.

Pengukuran pada penelitian ini dilakukan menggunakan *stopwatch* secara manual sehingga hasil yang diperoleh masih dapat dipengaruhi oleh waktu reaksi pengguna saat memulai dan menghentikan pengukuran. Selain itu, waktu respon sistem juga dapat dipengaruhi oleh kondisi jaringan WiFi yang digunakan, proses komunikasi data melalui MQTT, serta interval pembaruan data pada *dahboard* Grafana. Meskipun demikian, hasil pengujian menunjukkan bahwa sistem mampu menampilkan informasi *monitoring* dalam beberapa detik setelah aktivitas *shot* terjadi sehingga kondisi *dies* masih dapat dipantau dengan baik melalui *dahboard monitoring*.

5.2 Pembahasan

Pembahasan dilakukan untuk menganalisis hasil pengujian sistem yang telah dilakukan dengan meninjau kinerja setiap fungsi yang dikembangkan. Selain mengevaluasi keberhasilan implementasi sistem, pembahasan juga menguraikan kontribusi sistem terhadap proses *monitoring dies*, pelaksanaan *preventive maintenance*, serta penerapan konsep *Internet of Things* (IoT) dan *Operational Business Intelligence* pada lingkungan manufaktur.

Hasil pengujian yang telah diperoleh selanjutnya dianalisis untuk mengetahui kesesuaian sistem dengan tujuan penelitian serta mengidentifikasi kelebihan dan keterbatasan sistem yang dikembangkan.

5.2.1 Analisis Kinerja Sistem *Monitoring* berbasis IoT

Pengujian yang telah dilakukan menunjukkan bahwa seluruh fungsi utama pada sistem *monitoring dies* berbasis IoT telah berjalan sesuai dengan rancangan penelitian. Sistem mampu melakukan identifikasi *dies* melalui *barcode scanner*, melakukan pencatatan jumlah *shot* menggunakan *limit switch*, mengirimkan data melalui protokol MQTT, menyimpan data ke dalam *database* InfluxDB, serta menampilkan informasi *monitoring* secara *real-time* pada *dahboard* Grafana. Penggunaan *barcode scanner* memungkinkan setiap *dies* memiliki identitas yang berbeda sehingga data *monitoring* dapat disimpan dan dikelola secara terpisah. Hasil pengujian menunjukkan bahwa setelah *barcode* dipindai, sistem mampu mengubah identitas *dies* aktif secara otomatis dan seluruh data *shot* berikutnya akan dikaitkan dengan *dies* yang sesuai. Mekanisme ini memungkinkan proses *monitoring* dilakukan terhadap lebih dari satu *dies* tanpa terjadi pencampuran data.

Pada sisi akuisisi data, *limit switch* berfungsi sebagai sensor penghitung *shot* yang memberikan sinyal setiap kali terjadi simulasi proses produksi. Data yang diterima oleh ESP32 kemudian diproses dan dikirimkan ke MQTT *Broker*. Hasil pengujian menunjukkan bahwa setiap penambahan *shot* berhasil diterima oleh Node-RED dan tersimpan ke dalam *database* tanpa kehilangan data selama proses komunikasi berlangsung. Sistem juga mampu menampilkan kondisi *dies* secara *real-time* melalui *dahboard* Grafana. Informasi yang ditampilkan meliputi status operasi *dies*, jumlah *cycle shot*, persentase penggunaan *dies*, jumlah *preventive maintenance*, sisa *shot* menuju *preventive maintenance* berikutnya, *history* penggunaan *dies*, serta *history alert maintenance*. Tampilan *dahboard* tersebut memberikan kemudahan bagi operator untuk mengetahui kondisi *dies* tanpa harus melakukan pemeriksaan secara langsung.

Berdasarkan pengujian yang dilakukan, sistem mencatat rata-rata waktu respons sebesar 3,03 detik sejak aktivitas *shot* terjadi hingga data ditampilkan pada *dahboard*. Waktu respon tersebut masih lebih kecil dibandingkan waktu siklus produksi yang berada pada rentang 55–75 detik per *shot* sehingga informasi *monitoring* tetap dapat digunakan secara efektif untuk mendukung proses pemantauan *dies*. Berdasarkan hasil pengujian secara keseluruhan, sistem yang dikembangkan telah mampu memenuhi tujuan penelitian yaitu menghasilkan sistem *monitoring dies* berbasis IoT yang dapat digunakan untuk memantau penggunaan *dies* secara *real-time* serta mendukung pelaksanaan *preventive maintenance* berdasarkan jumlah *cycle shot* yang tercatat.

5.2.2 Analisis Implementasi IoT dan *Business Intelligence*

Sistem yang dikembangkan dalam penelitian menerapkan konsep IoT untuk menghubungkan perangkat akuisisi data dengan sistem *monitoring* secara *real-time*. Implementasi IoT dimulai dari proses pengambilan data pada sisi perangkat keras menggunakan ESP32 yang terhubung dengan *barcode scanner* dan *limit switch*. *Barcode scanner* digunakan untuk mengidentifikasi *dies* yang sedang digunakan, sedangkan *limit switch* digunakan untuk mendeteksi aktivitas *shot* selama proses produksi berlangsung.

Setelah data berhasil diperoleh dari perangkat input, ESP32 mengirimkan informasi tersebut melalui jaringan menggunakan protokol MQTT. Protokol ini dipilih karena memiliki kebutuhan *bandwidth* yang rendah, ringan untuk diimplementasikan pada perangkat mikrokontroler, serta mampu mendukung komunikasi data secara *real-time*. Selama proses pengujian, seluruh data yang dikirimkan oleh ESP32 berhasil diterima oleh MQTT Broker dan diteruskan ke Node-RED tanpa ditemukan kehilangan data.

Node-RED berperan sebagai pusat pengolahan data dalam sistem. Pada tahap ini dilakukan proses *Extract*, *Transform*, dan *Load* (ETL) sebelum data disimpan ke dalam *database*. Tahap *extract* dilakukan dengan menerima data dari MQTT *Broker*. Kemudian tahap *transform* mengolah data mentah menjadi informasi yang lebih bermakna, seperti *cycle shot*, *usage percentage*, PM *remaining*, PM *count*, serta klasifikasi status *dies*. Setelah proses pengolahan selesai, data dimuat (*load*) ke dalam InfluxDB sebagai media penyimpanan untuk mendukung proses *monitoring* dan visualisasi.

InfluxDB digunakan sebagai *time-series database* karena mampu menyimpan data *monitoring* berdasarkan urutan waktu secara efisien. Penggunaan *database* ini memungkinkan sistem menyimpan *history* penggunaan *dies*, *history preventive maintenance*, serta *history alert* yang dapat digunakan untuk analisis kondisi *dies* dalam jangka waktu tertentu. Hasil pengujian menunjukkan bahwa seluruh data *monitoring* berhasil tersimpan dan dapat ditampilkan kembali pada *dahboard* sesuai waktu kejadian.

Konsep *Business Intelligence* pada penelitian ini diimplementasikan melalui *dahboard* Grafana yang berfungsi mengubah data hasil *monitoring* menjadi informasi yang mudah dipahami oleh pengguna. *Dahboard* tidak hanya menampilkan data mentah, tetapi juga menyajikan berbagai indikator kinerja seperti jumlah *cycle shot*, persentase penggunaan *dies*, status operasi *dies*, status *preventive maintenance*, *history alert*, dan *history maintenance*.

Penyajian informasi dalam bentuk visual memungkinkan operator memperoleh gambaran kondisi *dies* secara cepat tanpa harus melakukan analisis data secara manual.

Berdasarkan hasil implementasi tersebut, dapat disimpulkan bahwa integrasi ESP32, MQTT *Broker*, Node-RED, InfluxDB, dan Grafana berhasil membentuk suatu sistem *monitoring* berbasis IoT dan BI. Melalui integrasi tersebut, data operasional yang diperoleh dari proses produksi dapat diolah menjadi informasi secara *real-time* sehingga dapat dimanfaatkan sebagai dasar dalam pengambilan keputusan *maintenance*.

5.2.3 Analisis *Preventive* dan *Alert Maintenance*

Salah satu kemampuan utama yang menjadi fokus penelitian ini adalah penerapan *preventive maintenance* berdasarkan jumlah *cycle shot dies*. Pendekatan tersebut berbeda dengan metode konvensional yang masih mengandalkan pencatatan manual maupun pengalaman operator dalam menentukan waktu pemeliharaan. Melalui sistem yang dikembangkan, penggunaan *dies* dapat dipantau secara otomatis, disertai pemberian peringatan ketika jumlah *cycle shot* telah mendekati batas *preventive maintenance* yang telah ditetapkan.

Pada penelitian ini, target *preventive maintenance* ditetapkan pada 1200 *shot*, sedangkan batas maksimum penggunaan *dies* ditetapkan pada 1500 *shot*. Berdasarkan parameter tersebut, sistem melakukan klasifikasi kondisi *dies* ke dalam empat kategori yaitu NORMAL, *WARNING*, *DUE*, dan *CRITICAL*. Kategori NORMAL menunjukkan bahwa *dies* masih berada pada kondisi penggunaan yang aman. Kategori *WARNING* menunjukkan bahwa *dies* telah mendekati batas *preventive maintenance* sehingga operator perlu mulai mempersiapkan kegiatan *maintenance*. Kategori *DUE* menunjukkan bahwa *dies* telah mencapai batas *preventive maintenance* yang ditentukan. Sementara itu, kategori *CRITICAL* menunjukkan bahwa *dies* telah melewati batas penggunaan yang direkomendasikan sehingga risiko kerusakan *dies* menjadi lebih tinggi.

Hasil pengujian menunjukkan bahwa perubahan status *dies* berjalan sesuai dengan logika yang telah dirancang pada sistem. Ketika jumlah *cycle shot* masih berada jauh di bawah target *preventive maintenance*, *dahboard* menampilkan status NORMAL dengan indikator berwarna hijau. Saat jumlah *cycle shot* mendekati target *preventive maintenance*, status berubah menjadi *WARNING* dan ditampilkan dengan indikator berwarna kuning. Setelah jumlah *shot* mencapai batas *preventive maintenance*, sistem mengubah status menjadi *DUE*. Selanjutnya

ketika jumlah *shot* telah melewati batas maksimum yang ditentukan, status berubah menjadi *CRITICAL* dengan indikator berwarna merah.

Selain menampilkan status pada *dahboard*, sistem juga menghasilkan notifikasi otomatis melalui Telegram. Notifikasi dikirim setiap kali terjadi perubahan status sehingga operator dapat segera mengetahui kondisi *dies* tanpa harus membuka *dahboard monitoring*. Informasi yang dikirim meliputi identitas *dies*, status *maintenance*, dan jumlah *cycle shot* saat notifikasi dihasilkan. Hasil pengujian menunjukkan bahwa notifikasi *WARNING*, *DUE*, dan *CRITICAL* berhasil dikirim sesuai kondisi aktual *dies* yang sedang dimoni*to*r.

Tabel 5. 4 Klasifikasi Status *Dies*

Status	**Kondisi**	**Tindakan**
NORMAL	*Cycle shot* < 960	Operasi normal
WARNING	*Cycle shot* ≥ 960	Persiapan *maintenance*
DUE	*Cycle shot* ≥ 1200	*Maintenance* harus dilakukan
CRITICAL	*Cycle shot* ≥ 1500	*Maintenance* segera dilakukan

Berdasarkan Tabel 5.3, sistem menerapkan mekanisme klasifikasi kondisi *dies* berdasarkan jumlah *cycle shot* yang tercatat. Klasifikasi tersebut digunakan sebagai dasar dalam pemberian notifikasi serta rekomendasi tindakan *maintenance* yang perlu dilakukan oleh operator. Untuk mendukung pengambilan keputusan *maintenance*, sistem juga menghitung nilai PM *Remaining* yang menunjukkan sisa jumlah *shot* sebelum *preventive maintenance* berikutnya perlu dilakukan. Informasi ini memberikan gambaran kepada operator mengenai umur penggunaan *dies* yang masih tersedia sehingga kegiatan *maintenance* dapat direncanakan dengan lebih baik. Ketika *preventive maintenance* telah selesai dilakukan, operator dapat menjalankan fungsi PM *Done* sehingga sistem secara otomatis melakukan reset *cycle shot* menjadi nol dan menambah nilai PM *Count* sebagai *history* pelaksanaan *maintenance*.

Berdasarkan hasil pengujian, mekanisme *preventive maintenance* yang diterapkan pada sistem mampu menyajikan informasi kondisi *dies* secara *real-time*, mengirimkan peringatan otomatis sesuai status penggunaan, serta mendokumentasikan *history maintenance* sebagai bahan evaluasi di masa mendatang. Dengan kemampuan tersebut, sistem tidak hanya berfungsi sebagai alat *monitoring*, tetapi juga dapat digunakan sebagai pendukung pengambilan keputusan *maintenance* berdasarkan kondisi aktual *dies*.

5.2.4 Analisis *Monitoring Multi-Dies*

Sistem *monitoring* terhadap penggunaan *dies* yang dikembangkan pada penelitian ini juga dirancang untuk mendukung *monitoring* lebih dari satu *dies* dalam satu platform *monitoring*. Kemampuan tersebut menjadi penting karena dalam lingkungan produksi, satu mesin dapat menggunakan beberapa *dies* yang berbeda sesuai kebutuhan produksi. Oleh karena itu, sistem harus mampu membedakan identitas setiap *dies* agar data *monitoring* yang dihasilkan tetap akurat dan tidak tercampur dengan *dies* lainnya.

Pada penelitian ini, identifikasi *dies* dilakukan menggunakan *barcode scanner*. Setiap *dies* memiliki *barcode* yang berbeda sehingga sistem dapat mengenali *dies* yang sedang digunakan sebelum proses produksi dimulai. Hasil pengujian menunjukkan bahwa *barcode scanner* mampu membaca identitas *dies* dan mengirimkan informasi tersebut ke ESP32. Selanjutnya informasi identitas *dies* digunakan sebagai referensi pada seluruh proses *monitoring*, mulai dari pencatatan *shot*, perhitungan *usage percentage*, pencatatan *preventive maintenance*, hingga penyimpanan *history* data.

Hasil pengujian *monitoring multi-dies* menunjukkan bahwa sistem mampu membedakan data *DIES*_01 dan *DIES*_02 secara terpisah. Ketika *DIES*_02 dipilih sebagai *dies* aktif melalui proses pemindaian *barcode*, seluruh aktivitas *shot* yang terjadi akan tercatat pada *DIES*_02 tanpa memengaruhi data yang dimiliki oleh *DIES*_01. Begitu pula ketika dilakukan pergantian *dies*, sistem akan mengubah referensi *monitoring* sesuai *barcode* yang dibaca sehingga pencatatan data tetap sesuai dengan identitas *dies* yang digunakan.

Kemampuan pemisahan data tersebut juga terlihat pada *dahboard* Grafana. Setiap *dies* memiliki informasi *cycle shot*, *usage percentage*, PM *count*, PM *remaining*, dan status operasi yang tersimpan secara independen. Dengan demikian pengguna dapat mengetahui kondisi masing-masing *dies* tanpa perlu melakukan pencatatan secara manual. Selain menyajikan kondisi terkini, sistem juga menyimpan riwayat penggunaan (*history usage*) dan riwayat pemeliharaan (*history maintenance*) untuk masing-masing *dies* berdasarkan identitasnya, sehingga informasi tersebut dapat diakses kembali apabila diperlukan.

Dari sisi pengelolaan *maintenance*, *monitoring multi-dies* memberikan keuntungan karena setiap *dies* memiliki riwayat penggunaan yang berbeda. Sistem dapat menghitung jumlah *shot* dan jadwal *preventive maintenance* berdasarkan kondisi aktual masing-masing *dies*. Dengan cara tersebut, kegiatan *maintenance* dapat dilakukan secara lebih tepat karena

keputusan *maintenance* tidak didasarkan pada estimasi atau pencatatan manual, melainkan berdasarkan data penggunaan yang tercatat pada sistem.

Hasil pengujian *monitoring multi-dies* juga menunjukkan bahwa sistem memiliki potensi untuk diterapkan pada lingkungan produksi dengan jumlah *dies* yang lebih banyak. Pada penelitian ini pengujian dilakukan menggunakan dua *dies*, yaitu *DIES*_01 dan *DIES*_02, dan seluruh data *monitoring* berhasil disimpan, diproses, serta ditampilkan secara terpisah sesuai identitas masing-masing *dies*. Keberhasilan tersebut menunjukkan bahwa mekanisme identifikasi *dies* menggunakan *barcode*, pengelolaan data pada Node-RED, serta penyimpanan data pada InfluxDB mampu menangani lebih dari satu objek *monitoring* dalam satu sistem.

Dari hasil tersebut, sistem secara konseptual dapat diperluas untuk memoni*to*r jumlah *dies* yang lebih banyak tanpa mengubah prinsip kerja utama yang telah diterapkan. Setiap *dies* hanya perlu memiliki identitas *barcode* yang berbeda sehingga sistem dapat membedakan dan mengelola data masing-masing *dies* secara independen. Selain itu, penggunaan identitas *dies* dan identitas mesin pada *database* memungkinkan data *monitoring* tetap terorganisir meskipun jumlah *dies* yang dipantau bertambah.

Keberhasilan implementasi *monitoring multi-dies* juga menunjukkan bahwa arsitektur sistem memiliki potensi untuk dikembangkan menjadi sistem *monitoring multi*-mesin. Jika pada penelitian ini sistem berhasil mengelola beberapa *dies* dalam satu *dahboard monitoring*, maka prinsip yang sama dapat diterapkan pada beberapa mesin dengan menambahkan identitas mesin sebagai parameter *monitoring*. Dengan demikian data dari beberapa mesin dapat dikumpulkan dan dipantau melalui satu *dahboard* yang terpusat.

Implementasi dan pengujian yang telah dilakukan menunjukkan bahwa sistem dapat mengelola lebih dari satu *dies* dalam satu *dashboard monitoring* tanpa mengurangi konsistensi maupun akurasi data. Setiap informasi yang diterima tetap diproses berdasarkan identitas *dies* masing-masing sehingga hasil *monitoring* dapat ditampilkan secara terpisah. Fitur ini tidak hanya mendukung *monitoring dies* yang digunakan secara bergantian dalam proses produksi, tetapi juga membuka peluang pengembangan sistem menuju *monitoring multi-dies* dan *multi*-mesin pada skala yang lebih luas.

5.2.5 Implikasi Hasil Penelitian pada Sistem *Monitroring Dies*

Hasil implementasi sistem *monitoring dies* yang dikembangkan pada penelitian ini memperlihatkan bahwa penerapan IoT dan *Operational* BI memberikan kemudahan dalam

proses *monitoring* serta pelaksanaan *preventive maintenance* dibandingkan metode pencatatan manual. Informasi penggunaan *dies* dapat diperbarui secara *real-time*, tersimpan secara otomatis, dan disajikan melalui satu *dashboard* yang terintegrasi.

Tabel 5. 5 Improvisasi Penelitian

Sebelum Sistem	**Sesudah Sistem**
Pencatatan manual	*Monitoring digital*
Tidak ada *history* otomatis	*History* tersimpan otomatis
Maintenance berdasarkan pencatatan manual	*Maintenance* berdasarkan *cycle shot*
Tidak ada *alert*	Ada *alert* Telegram
Sulit mengetahui kondisi *dies*	*Dahboard real-time*

Berdasarkan perbandingan dalam tabel 5.4, implementasi sistem *monitoring dies* memberikan perbaikan pada proses *monitoring* dan pemeliharaan *dies* dibandingkan metode yang digunakan sebelumnya. Perbaikan tersebut terlihat pada proses pencatatan data yang telah dilakukan secara otomatis, penyediaan informasi kondisi *dies* secara *real-time*, penyimpanan *history monitoring*, serta pemberian *alert maintenance* secara otomatis. Selain itu, sistem juga mampu mengelola *monitoring* lebih dari satu *dies* dalam satu *dahboard* sehingga proses pemantauan dapat dilakukan secara lebih terstruktur dan efisien. Beriku merupakan Implikasi hasil penelitian yang diperoleh antara lain.

1. *Monitoring* penggunaan *dies* dapat dilakukan secara *real-time* melalui *dahboard* Grafana sehingga operator dapat mengetahui kondisi *dies* tanpa harus melakukan pemeriksaan secara manual.
2. Data penggunaan *dies* tersimpan secara otomatis dalam *database* sehingga *history cycle shot*, *history maintenance*, dan *history alert* dapat ditelusuri kembali apabila diperlukan.
3. Informasi *usage percentage*, PM *remaining*, dan status *maintenance* dapat digunakan sebagai dasar dalam menentukan waktu pelaksanaan *preventive maintenance* berdasarkan kondisi aktual *dies*.
4. Notifikasi Telegram memungkinkan informasi kondisi *dies* diterima lebih cepat ketika status berubah menjadi *WARNING*, *DUE*, maupun *CRITICAL*.
5. Keberhasilan pengujian *monitoring multi-dies* menunjukkan bahwa sistem mampu mengelola lebih dari satu *dies* dalam satu *dahboard monitoring* tanpa terjadi pencampuran data antar *dies*.
6. Arsitektur sistem yang berhasil mengelola dua *dies* secara terpisah menunjukkan bahwa sistem memiliki potensi untuk dikembangkan pada jumlah *dies* yang lebih banyak.

Selain itu, penggunaan identitas mesin pada *database* juga memberikan peluang pengembangan menuju sistem *monitoring multi*-mesin dalam satu platform *monitoring* terpusat.

Informasi yang dihasilkan sistem berfungsi sebagai pendukung *preventive maintenance*. Penentuan waktu pemeliharaan dilakukan berdasarkan kondisi aktual *dies* yang ditunjukkan melalui parameter *cycle shot, usage percentage*, PM *remaining*, dan status *maintenance*. Pada implementasinya, keputusan tetap berada pada *foreman* atau bagian *maintenance*, sedangkan sistem berperan sebagai penyedia informasi yang mendukung proses pengambilan keputusan tersebut.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan proses perancangan, implementasi, dan pengujian sistem yang telah dilakukan, maka dapat diperoleh kesimpulan sebagai berikut.

1. Pendekatan *Business Intelligence Operational* berhasil diterapkan pada sistem *monitoring* pemeliharaan *dies* berbasis *Internet of things* (IoT) melalui integrasi ESP32, MQTT *Broker*, Node-RED, InfluxDB, dan Grafana. Sistem mampu mengumpulkan, mengolah, menyimpan, dan menyajikan data penggunaan *dies* dalam bentuk *dahboard monitoring* sehingga dapat mendukung proses pengambilan keputusan *preventive maintenance* berdasarkan data aktual penggunaan *dies*.
2. Sistem yang dikembangkan berhasil menyajikan informasi penggunaan *dies* secara *real-time* berupa *cycle shot*, *usage percentage*, PM *remaining*, PM *count*, status *dies*, *history maintenance*, dan *history alert*. Selain itu, sistem juga mampu memberikan peringatan otomatis melalui *dahboard* Grafana dan Telegram ketika kondisi *dies* mencapai kategori *WARNING*, *DUE*, dan *CRITICAL*. Hasil pengujian menunjukkan bahwa sistem mampu memantau lebih dari satu *dies* secara terpisah sehingga dapat membantu proses pemeliharaan, pemantauan, dan penentuan waktu *preventive maintenance* secara lebih mudah, terstruktur, dan tepat waktu.

6.2 Saran

Sebagai upaya pengembangan sistem *monitoring dies* yang telah dirancang terdapat beberapa rekomendasi yang dapat dijadikan sebagai pengembangan pada penelitian selanjutnya, yaitu sebagai berikut.

1. Penelitian selanjutnya disarankan mengimplementasikan sistem secara langsung pada mesin *die casting* di lingkungan produksi sehingga data penggunaan *dies* dapat diperoleh dari kondisi *Operational* yang sebenarnya.
2. Sistem dapat dikembangkan dengan menambahkan jumlah *dies* dan jumlah mesin yang dimonitor dalam satu *dahboard* untuk menguji kemampuan sistem dalam mengelola

data *monitoring* pada skala yang lebih besar dan terpusat untuk beberapa mesin produksi sekaligus.

3. *Dahboard monitoring* dapat dikembangkan dengan menambahkan fitur analisis data *history*s penggunaan *dies* sehingga dapat memberikan informasi yang lebih mendukung proses perencanaan dan pengambilan keputusan *maintenance*.

DAFTAR PUSTAKA

Anggara, W. E. F., Yuana, H., & Puspitasari, W. D. (2023). Rancang Bangun Alat Monitor Ketinggian Air Berbasis Internet of Things (IoT) Menggunakan ESP32 Dan Framework Blynk. In *Jurnal Mahasiswa Teknik Informatika* (Vol. 7, Number 5).

Ardiansyah, M. F., & Widjajati, E. P. (2021). *Penjadwalan Preventive Maintenance Pada Mesin Mixing Dalam Produksi Brick Batu Tahan Api Dengan Menggunakan Metode Age Replacement Pada PT. Loka Refractories Wira Jatim*.

Banks, A., & Gupta, R. (2015). *mqtt-v3.1.1Plus Errata 01*. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/errata01/os/mqtt-v3.1.1-errata01-os-complete.doc

Bristy, I. J., Tabassum, M., Islam, M. I., & Hasan, Md. N. (2025). IoT-Driven Predictive Maintenance Dashboards in Industrial Operations. *Saudi Journal of Engineering and Technology*, *10*(09), 457–466. https://doi.org/10.36348/sjet.2025.v10i09.009

Burggräf, P., Bergweiler, G., Kehrer, S., Krawczyk, T., & Fiedler, F. (2024). Mega-casting in the automotive production system: Expert interview-based impact analysis of large-format aluminium high-pressure die-casting (HPDC) on the vehicle production. *Journal of Manufacturing Processes*, *124*, 918–935. https://doi.org/10.1016/j.jmapro.2024.06.028

Chaudhuri, S., Dayal, U., & Narasayya, V. (2011). An overview of business intelligence technology. In *Communications of the ACM* (Vol. 54, Number 8, pp. 88–98). https://doi.org/10.1145/1978542.1978562

Chen, H., Chiang, R. H. L., Storey, V. C., & Robinson, J. M. (2012). *Special Issue: Business Intelligence Research And Analytics: From Big Data To Big Impact*. http://misq.umn.edu/misq/article-pdf/36/4/1165/5436/8_si_chenintroduction.pdf

Elkateb, S., Métwalli, A., Shendy, A., & Abu-Elanien, A. E. B. (2024). Machine learning and IoT – Based predictive maintenance approach for industrial applications. *Alexandria Engineering Journal*, *88*, 298–309. https://doi.org/10.1016/j.aej.2023.12.065

Elradi, M. D. (2025). Prometheus & Grafana: A Metrics-focused Monitoring Stack. *Journal of Computer Allied Intelligence*, *3*(3), 28–39. https://doi.org/10.69996/jcai.2025015

Fatchurramadhani. (2022). *Implementasi Kecerdasan Bisnis Sebagai Sistem Pendukung*.

Few, S. (2003). *Information Dashboard Design*. www.PerceptualEdge.com

Firmansyah, H., & Indrasari, W. (2025). Internet of Things-Based Real-Time Thermal Monitoring: Utilizing the Affordable MLX90640 Thermal Camera and the Industrial MAX6675 Thermocouple Module with a Linear Regression Characterization Method. *POSITRON*, *15*(1), 81–91. https://doi.org/10.26418/positron.v15i1.89878

Garcia, A., Oregui, X., Franco, J., & Arrieta, U. (2022). Edge Containerized Architecture for Manufacturing Process Time Series Data Monitoring and Visualization. *IN4PL - Proceedings of the International Conference on Innovative Intelligent Industrial Production and Logistics*, 145–152. https://doi.org/10.5220/0011574500003329

GMM Mould Technology. (2024). *Die Casting Mould Design: Complete Guide for Aluminum, Zinc & Magnesium*. https://www.gmmoldtech.com/die-casting-mould-design-complete-guide-for-aluminum-zinc-amp-magnesium.html?utm_source=chatgpt.com

Guerbaoui, M., El Faiz, S., Ed-Dahhak, A., Lachhab, A., Benhala, B., Bakziz, Z., Ichou, I., & Selmani, A. (2025). From Data to Decisions: A Smart IoT and Cloud Approach to

Environmental Monitoring. *E3S Web of Conferences*, *601*. https://doi.org/10.1051/e3sconf/202560100008

Hidayar, R. N., & Kardiman, M. (2024). *Proses Produksi Crank Case Dengan Metode HPDC (High Pressure Die Casting)*.

Hidayat, D., & Sari, I. (2021). Monitoring suhu dan kelembaban berbasis Internet of Things (IoT). *Jurnal Teknologi Dan Ilmu Komputer Prima (JUTIKOMP)*, *4*(1), 525–530.

Inmon, W. (2002). *Building the Data Warehouse Third Edition*.

Islamy, I., & Wisudawati, L. M. (2023). Sistem Monitoring Smart Garden Tanaman Cabai Berbasis IoT Menggunakan Protokol MQTT, Node Red, dan Telegram Bot. *Jurnal Teknotan*, *17*(3), 197. https://doi.org/10.24198/jt.vol17n3.6

Kaswadi, A., Wahyu Triantoro, V., Ari Wibowo, E., Produksi dan Proses Manufaktur, T., & Astra, P. (2024). *ANALISIS PROSEDUR PERAWATAN CYLINDER COMP DIES UNTUK MENURUNKAN DOWNTIME DIES FLASH DI PT XYZ DENGAN METODE EIGHT STEPS* (Vol. 3, Number 2). http://ejournal.unimugo.ac.id/JITIN

Katta, S. R. (2025). Leveraging Power BI for Data-Driven Decision-Making in Pharma Maintenance Operations. *Journal of Research in Business and Management*, *13*(2), 61–70. https://doi.org/10.35629/3002-13026170

Keller, S., & Owen, A. (2025). *A Comprehensive Cost-Benefit Analysis of Preventive Maintenance Versus Corrective Maintenance: Assessing the Financial Impact and Operational Benefits in Engineering*.

Kimball, R., & Ross, M. (2002). *The Data Warehouse Toolkit Second Edition The Complete Guide to Dimensional Modeling*.

Kleppmann, Martin. (2021). *Designing Data-Intensive Applications*. Upfront Books.

Kurniawan, I., Dhoni, R., Saputra, D. A., Wahyudin, F. R., Mubais, A., Lunak, R. P., Balekambang, P., Rekayasa, J., Lunak, P., Bisnis, J. A., Politeknik, I., & Jepara, B. (2023). *Implementasi Metode Prototype Pada Sistem Informasi Digital Angkringan Berbasis Web di Kecamatan Mayong*. https://journal.unisnu.ac.id/JISTER/

Lakeshore Die Cast. (2026). *Die Casting Cost: Everything You Need to Know*. https://lakeshorediecast.com/die-casting-cost/

Mizgan, H., & Ganea, M. (2022). Optimization of aluminium die-casting process through predictive maintenance and parameter traceability systems. *IOP Conference Series: Materials Science and Engineering*, *1256*(1), 012028. https://doi.org/10.1088/1757-899x/1256/1/012028

Mobley, R. Keith., Higgins, L. R. ., & Wikoff, D. J. . (2008). *Maintenance engineering handbook*. McGraw-Hill.

Nizam, M., Yuana, H., & Wulansari, Z. (2022). Mikrokontroler ESP32 Sebagai Alat Monitoring Pintu Berbasis Web. In *Jurnal Mahasiswa Teknik Informatika)* (Vol. 6, Number 2).

Noor Dhiana, F. A., Supriyono, Abid Yanuar Badharudin, & Achmad Fauzan. (2025). Implementasi Sistem Pengelolaan Program Mentoring Mahasiswa Baru Menggunakan Metode Prototyping. *Bulletin of Computer Science Research*, *5*(4), 287–297. https://doi.org/10.47065/bulletincsr.v5i4.538

North American Die Casting Association. (2021). *NADCA Product Specification Standards for Die Castings*.

Nuraeni, S. (2024). *Analisis Visualisasi Pendapatan Perusahaan Sektor Industri Manufaktur Menggunakan Tableau Public* (Vol. 3).

Panda, S. P., & Padhy, A. (2025). Business Intelligence with Power BI and Tableau: Cloud-Based Data Warehousing, Predictive Analytics, and Artificial Intelligence-Driven

Decision Support. In *Business Intelligence with Power BI and Tableau: Cloud-Based Data Warehousing, Predictive Analytics, and Artificial Intelligence-Driven Decision Support*. Deep Science Publishing. https://doi.org/10.70593/978-93-7185-811-3

Ramadhan, H. F., Fauzi, A., Rupelu, C. N., Aprillia, D. P., Anjani, N. D., & Halimatusadiah. (2022). *Pengaruh Business Intelligence Terhadap PErusahaan Dalam Pengambilan Keputusan Business Intelligence, Arsitektur BI Dan Data Warehouse (Kajian Studi Business Intelligence)*.

Ramadhan, I., & Fitriani, R. (2025). Optimalisasi Efektivitas Preventive Maintenance Berbasis Usage-Based Maintenance untuk Mengurangi Downtime di PT PQR. *Journal of Integrated System*, *7*(2), 166–183. https://doi.org/10.28932/jis.v7i2.9760

Sacu, C., & Spruit, M. (2010). BIDM - The business intelligence development model. *ICEIS 2010 - Proceedings of the 12th International Conference on Enterprise Information Systems*, *1 DISI*, 288–293. https://doi.org/10.5220/0002967402880293

Saory, F., Jaman, J. H., & Author, C. (2025). Sistem Monitoring Server Menggunakan Prometheus Dan Grafana DenganIntegrasi Slack Di PT. Atlas LIntas Indonesia. *Jurnal Informatika Dan Teknik Elektro Terapan*, *13*(3S1). https://doi.org/10.23960/jitet.v13i3S1.8024

Saputra, M. D., & Marzuki, K. (2025). *Desain dan Implementasi Monitoring Infrastruktur Perangkat Jaringan dan Server Menggunakan Grafana dan Prometheus* (Vol. 12, Number 3). http://jurnal.mdp.ac.id

Shapouri, F. (2020). *Association for Information Systems Association for Information Systems Critical Success Factors in Implementing Business Intelligence Critical Success Factors in Implementing Business Intelligence Systems Systems*. https://aisel.aisnet.org/amcis2020/org_transformation_is/org_transformation_is/11

Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, *3*(5), 637–646. https://doi.org/10.1109/JIOT.2016.2579198

Sommerville, Ian. (2016). *Software engineering*. Pearson.

Soori, M., Arezoo, B., & Dastres, R. (2023). Internet of things for smart factories in industry 4.0, a review. In *Internet of Things and Cyber-Physical Systems* (Vol. 3, pp. 192–204). KeAi Communications Co. https://doi.org/10.1016/j.iotcps.2023.04.006

Sumasto, F., Ramadhani Jiwanto, R., Handoko Purwojatmiko, B., Rizki Pratama, I., Studi Teknik Industri Otomotif, P., STMI Jakarta, P., & Penulis, K. (2023). *Implementasi Penjadwalan Preventive Maintenance Untuk Meningkatkan Nilai Efektivitas Mesin Pada Mesin CNC Milling VL-10i*.

Syarif, M., & Risdiansyah, D. (2024). Pemanfaatan Metode Prototype Dalam Perancangan Sistem Informasi Penjualan Berbasis Website. In *Jurnal Mahasiswa Teknik Informatika* (Vol. 8, Number 4).

Systems, E. (2021). *ESP32 Series Datasheet*. Espressif Systems Shanghai, China.

Watson, H. J., & Wixom, B. H. (2004). *The Current State of Business Intelligence*. www.tdwi.org

Wicaksono, M. A., Rudianto, C., & Tanaem, P. F. (2021). Rancang Bangun Sistem Informasi Arsip Surat Menggunakan Metode Prototype. *Jurnal Teknik Informatika Dan Sistem Informasi*, *7*(2). https://doi.org/10.28932/jutisi.v7i2.3664

Xu, L. Da, Xu, E. L., & Li, L. (2018). Industry 4.0: state of the art and future trends. *International Journal of Production Research*, *56*(8), 2941–2962.

Yudha, H., Munthe, P., Farizi Hernando, M., Amalia, D., Indra Martadinata, M., Soleh, A. M., & Kurniadi, D. (2023). Optimalisasi Monitoring Peralatan Genset Manajemen

Pemeliharaan, IoT, dan Desain. In *Hal : xx-xxx Journal of Engineering and Transportation (JET)* (Vol. 1, Number 1).

Yuli Setiawannie, & Nita Marikena. (2022). Perencanaan Penjadwalan Preventive Maintenance Mesin Pounch dengan Critical Path Method di PT. Grafika Nusantara. *INSOLOGI: Jurnal Sains Dan Teknologi*, *1*(1), 01–10. https://doi.org/10.55123/insologi.v1i1.105

LAMPIRAN

A-Coding ESP32 Arduino IDE

```
#include <WiFi.h>
#include <PubSubClient.h>

// ===== PIN INPUT =====
#define PIN_SHOT      18
#define PIN_PM_START  19
#define PIN_PM_DONE   23

// ===== PIN LED =====
#define LED_RUNNING 25
#define LED_PM      26
#define LED_ERROR   27

// ===== WIFI =====
const char* ssid = "HIFIAIR_9B13";
const char* password = "71021013";
const char* mqtt_server = "LAPTOP-9BQO1U7R.local";

// ===== MQTT =====
WiFiClient espClient;
PubSubClient client(espClient);

// ===== STATE =====
int lastShot, lastPMStart, lastPMDone;
String currentState = "STANDBY";
String DIES_ID = "NONE";

// ===== BUILD TOPIC =====
String buildTopic(String event) {
  return "dies/" + DIES_ID + "/" + event;
}

```

A-Coding ESP32 Arduino IDE (Lanjutan)

```
// ===== BARCODE INPUT =====
void readBarcodeFromSerial() {
  if (Serial.available()) {
    String input = Serial.readStringUntil('\n');
    input.trim();

    if (input.length() > 0) {
      Serial.print("SCAN BARCODE: ");
      Serial.println(input);

      // langsung update dies
      DIES_ID = input;

      // kirim ke Node-RED
      client.publish("dies/config", input.c_str());

      // reset state
      currentState = "STANDBY";
      updateLED();
    }
  }
}

// ===== MQTT CALLBACK =====
void callback(char* topic, byte* payload, unsigned int length) {

  String message = "";

  for (int i = 0; i < length; i++) {
    message += (char)payload[i];
  }

```

A-Coding ESP32 Arduino IDE (Lanjutan)

```
  if (String(topic) == "dies/config") {
    Serial.println("CONFIG MASUK (ACK): " + message);
  }
}

// ===== MQTT RECONNECT =====
void reconnectMQTT() {
  while (!client.connected()) {
    Serial.println("Connecting to MQTT...");

    if (client.connect("ESP32Client")) {
      Serial.println("MQTT connected");
      client.subscribe("dies/config");
    } else {
      delay(2000);
    }
  }
}

// ===== UPDATE LED =====
void updateLED() {
  if (currentState == "RUNNING") {
    digitalWrite(LED_RUNNING, HIGH);
    digitalWrite(LED_PM, LOW);
    digitalWrite(LED_ERROR, LOW);
  }
  else if (currentState == "PM") {
    digitalWrite(LED_RUNNING, LOW);
    digitalWrite(LED_PM, HIGH);
    digitalWrite(LED_ERROR, LOW);
  }
```

A-Coding ESP32 Arduino IDE (Lanjutan)

```
  else if (currentState == "ERROR") {
    digitalWrite(LED_RUNNING, LOW);
    digitalWrite(LED_PM, LOW);
    digitalWrite(LED_ERROR, HIGH);
  }
  else { // STANDBY
    digitalWrite(LED_RUNNING, LOW);
    digitalWrite(LED_PM, LOW);
    digitalWrite(LED_ERROR, LOW);
  }
}

// ===== SETUP =====
void setup() {
  Serial.begin(115200);

// INPUT
  pinMode(PIN_SHOT, INPUT_PULLUP);
  pinMode(PIN_PM_START, INPUT_PULLUP);
  pinMode(PIN_PM_DONE, INPUT_PULLUP);

// LED OUTPUT
  pinMode(LED_RUNNING, OUTPUT);
  pinMode(LED_PM, OUTPUT);
  pinMode(LED_ERROR, OUTPUT);

 // UNIT LED OFF
  digitalWrite(LED_RUNNING, LOW);
  digitalWrite(LED_PM, LOW);
  digitalWrite(LED_ERROR, LOW);
```

A-Coding ESP32 Arduino IDE (Lanjutan)

```
  lastShot     = digitalRead(PIN_SHOT);
  lastPMStart  = digitalRead(PIN_PM_START);
  lastPMDone   = digitalRead(PIN_PM_DONE);

  Serial.println("Connecting to WiFi...");
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }

  Serial.println("WiFi connected");

  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

// ===== LOOP =====
void loop() {

  if (!client.connected()) {
    reconnectMQTT();
  }

  client.loop();

  // ===== BARCODE =====
  readBarcodeFromSerial();
```

A-Coding ESP32 Arduino IDE (Lanjutan)

```
157   if (DIES_ID == "NONE") return;
158
159   // ===== SHOT =====
160   int currShot = digitalRead(PIN_SHOT);
161   if (lastShot == 1 && currShot == 0) {
162
163     currentState = "RUNNING";
164
165     client.publish(buildTopic("shot").c_str(), "1");
166
167     Serial.println("SHOT (" + DIES_ID + ")");
168
169     updateLED();
170     delay(120);
171   }
172   lastShot = currShot;
173
174   // ===== PM START =====
175   int currPMStart = digitalRead(PIN_PM_START);
176   if (lastPMStart == 1 && currPMStart == 0) {
177
178     currentState = "PM";
179
180     client.publish(buildTopic("pm_start").c_str(), "1");
181
182     Serial.println("PM START (" + DIES_ID + ")");
183
184     updateLED();
185     delay(120);
186   }
187   lastPMStart = currPMStart;
```

A-Coding ESP32 Arduino IDE (Lanjutan)

```
  /// ===== PM DONE =====
  int currPMDone = digitalRead(PIN_PM_DONE);
  if (lastPMDone == 1 && currPMDone == 0) {

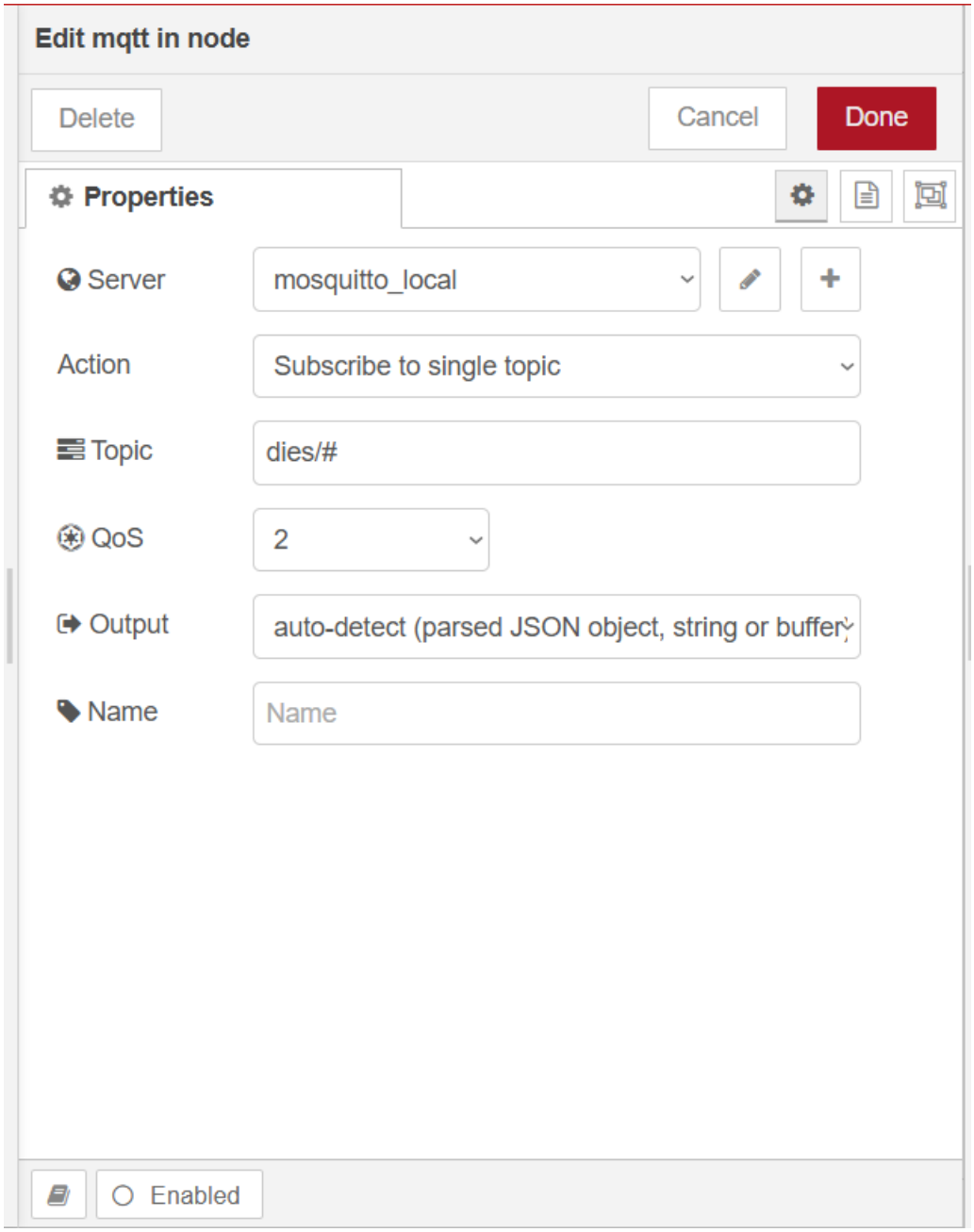
    currentState = "STANDBY";

    client.publish(buildTopic("pm_done").c_str(), "1");

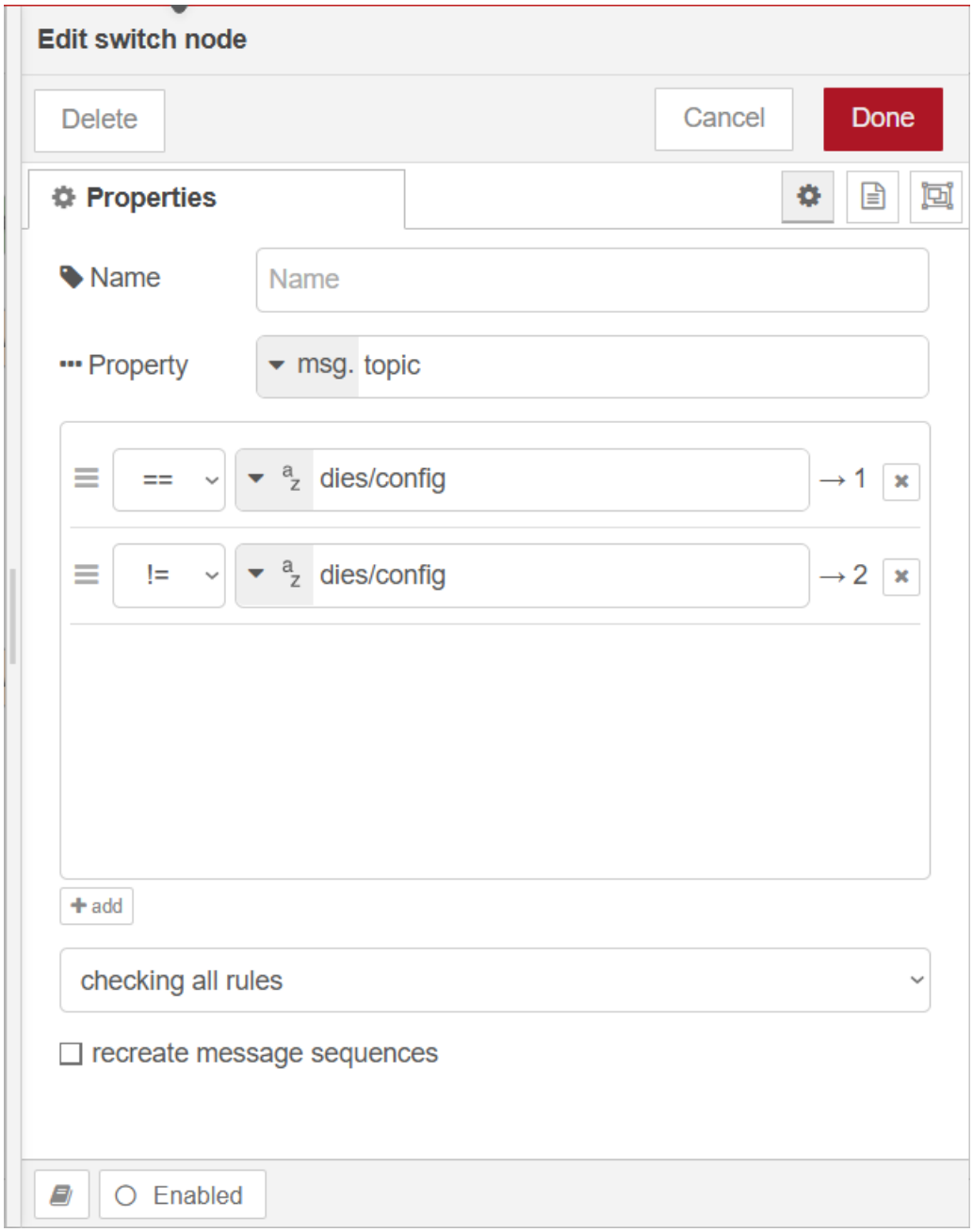
    Serial.println("PM DONE (" + DIES_ID + ")");

    updateLED();
    delay(120);
  }
  lastPMDone = currPMDone;
}
```

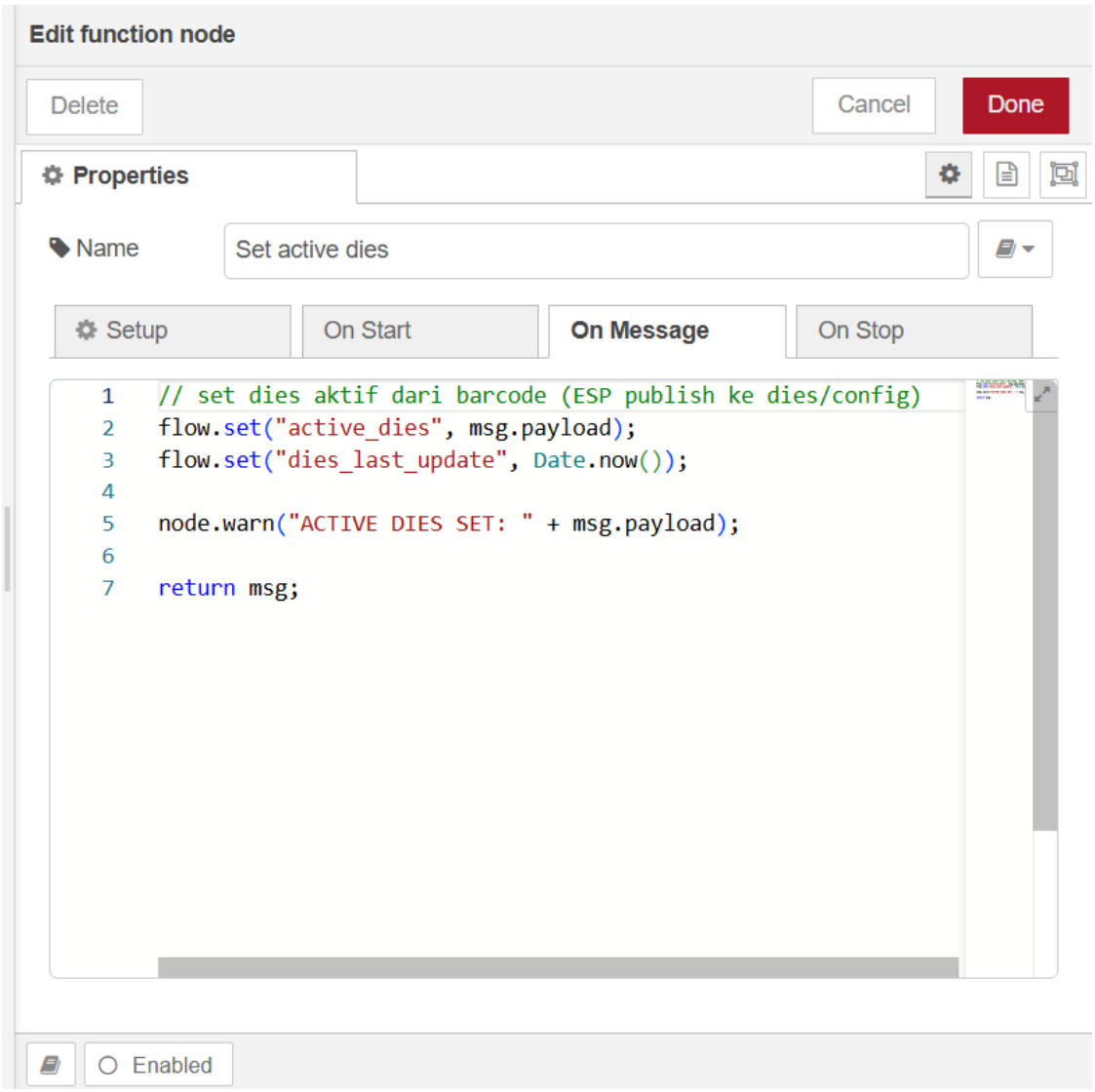
B-Properties Node-RED



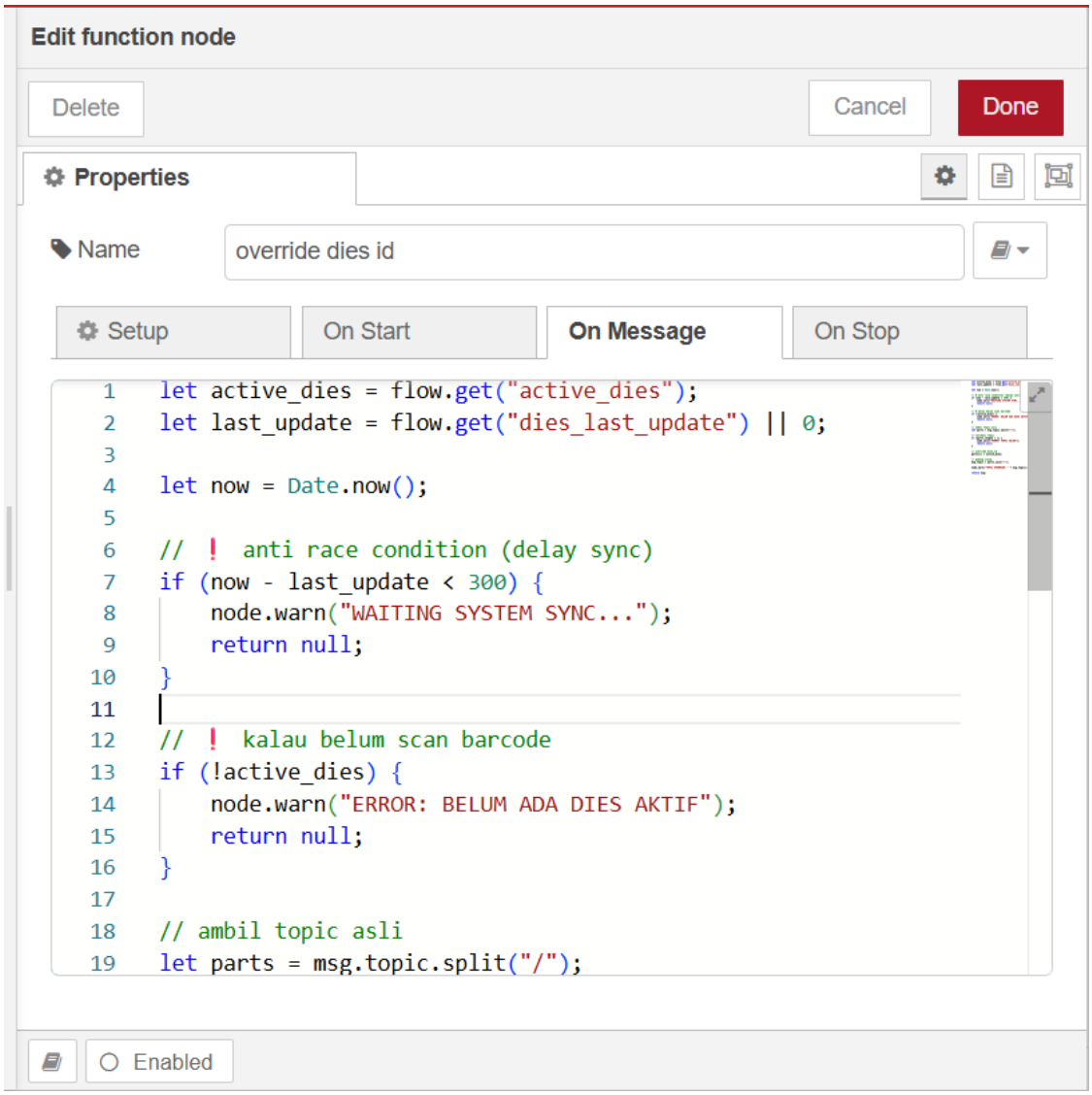
B-Properties Node-RED (Lanjutan)



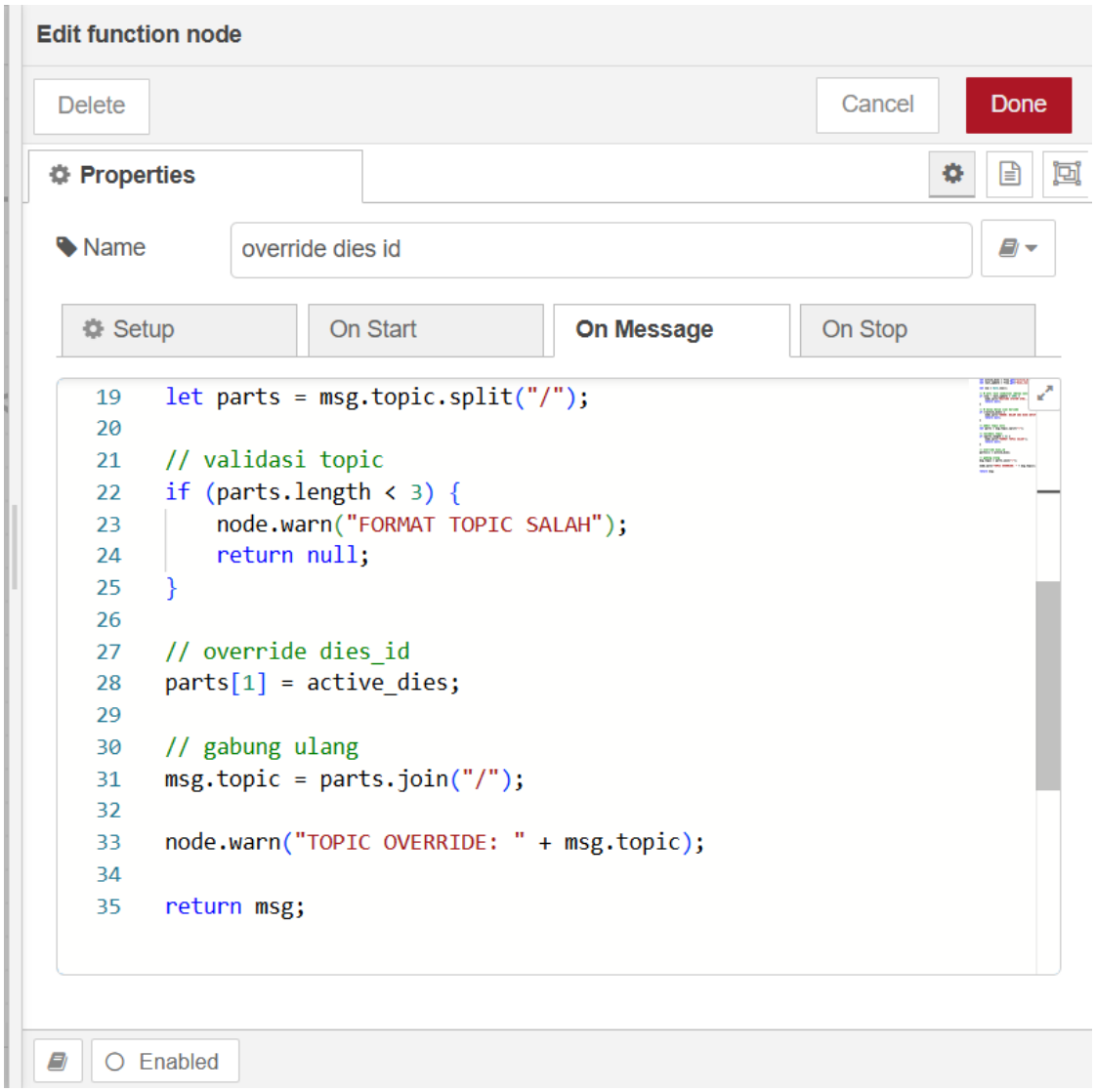
B-Properties Node-RED (Lanjutan)



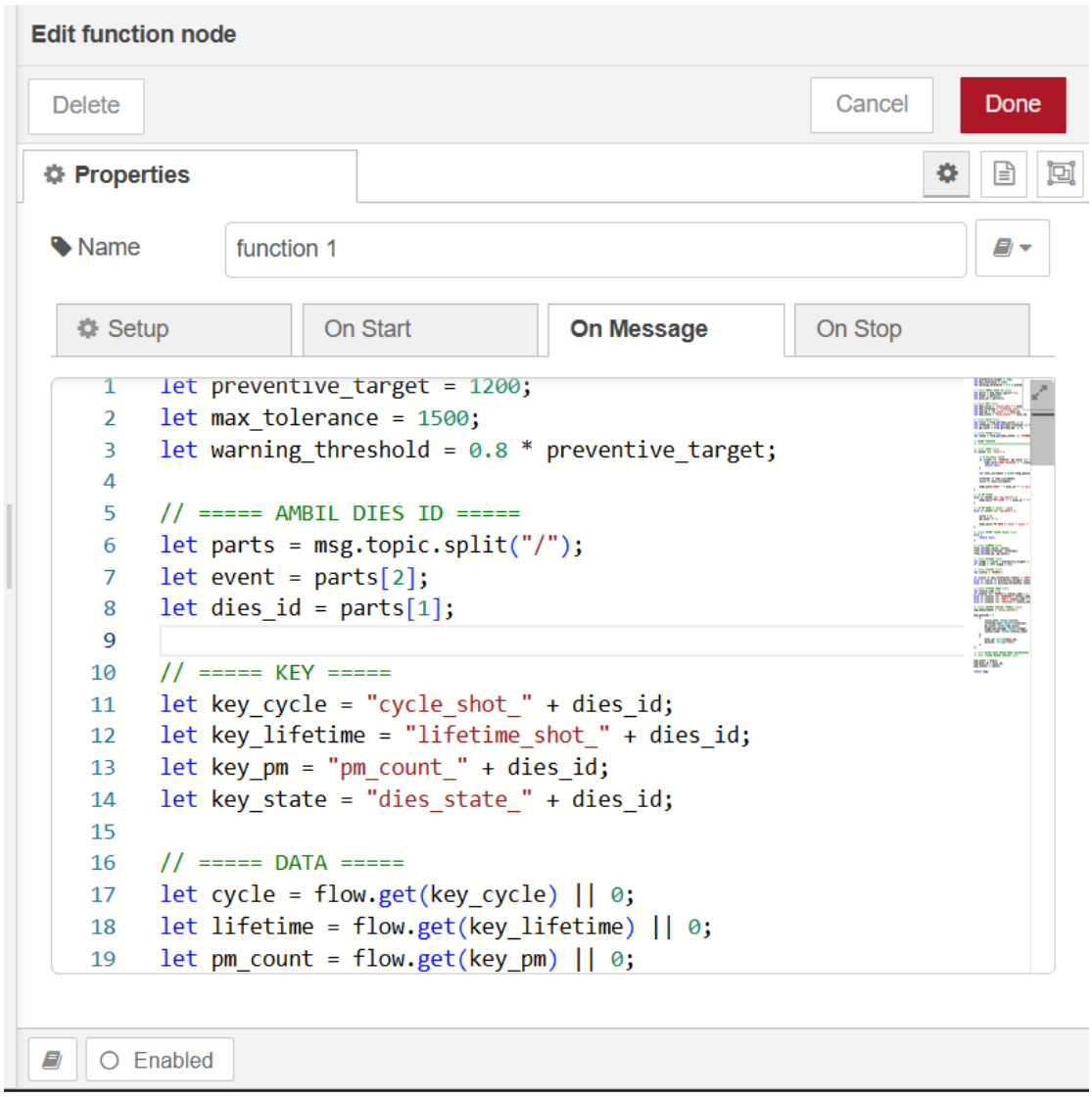
B-Properties Node-RED (Lanjutan)



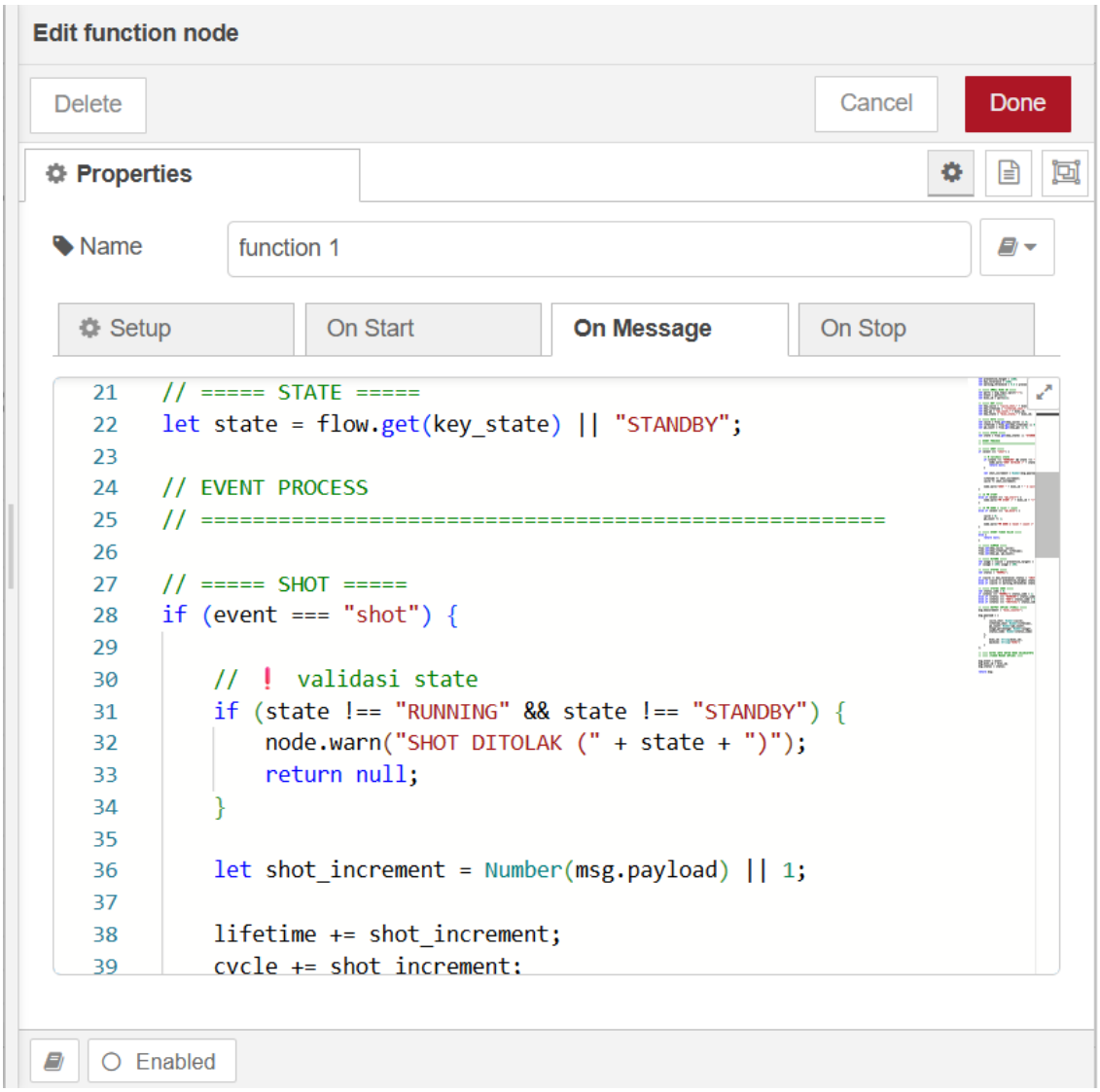
B-Properties Node-RED (Lanjutan)



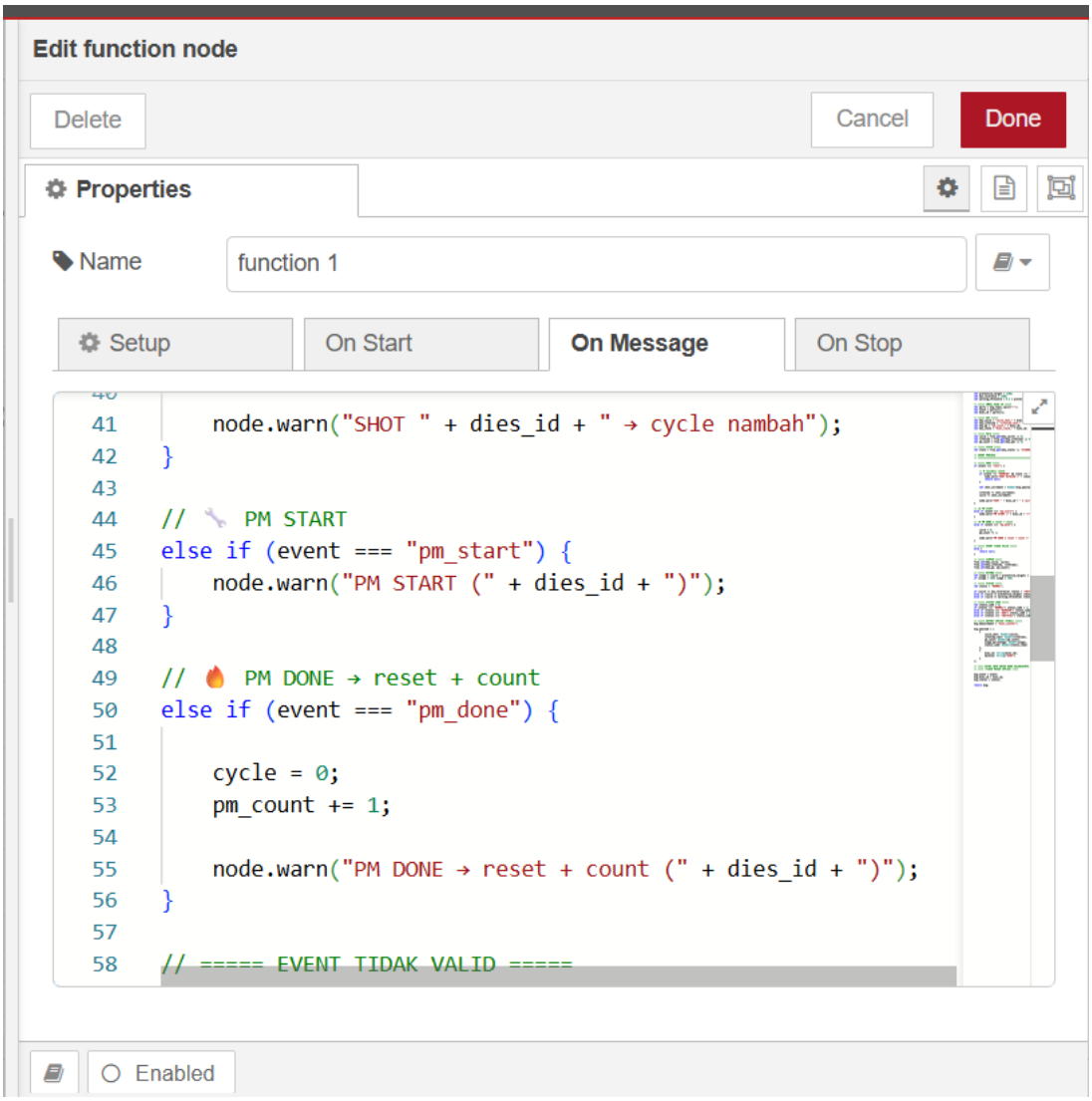
B-Properties Node-RED (Lanjutan)



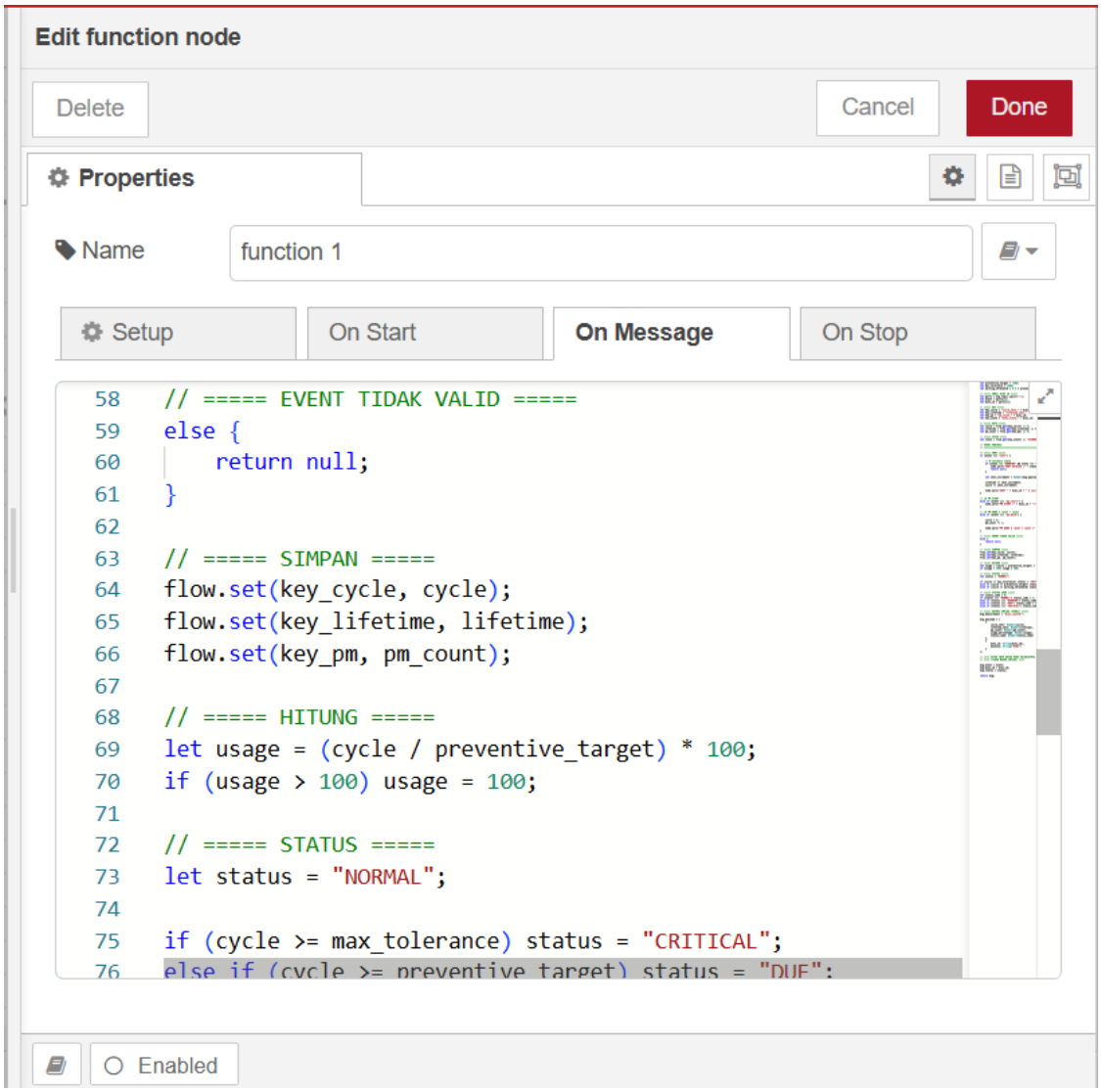
B-Properties Node-RED (Lanjutan)



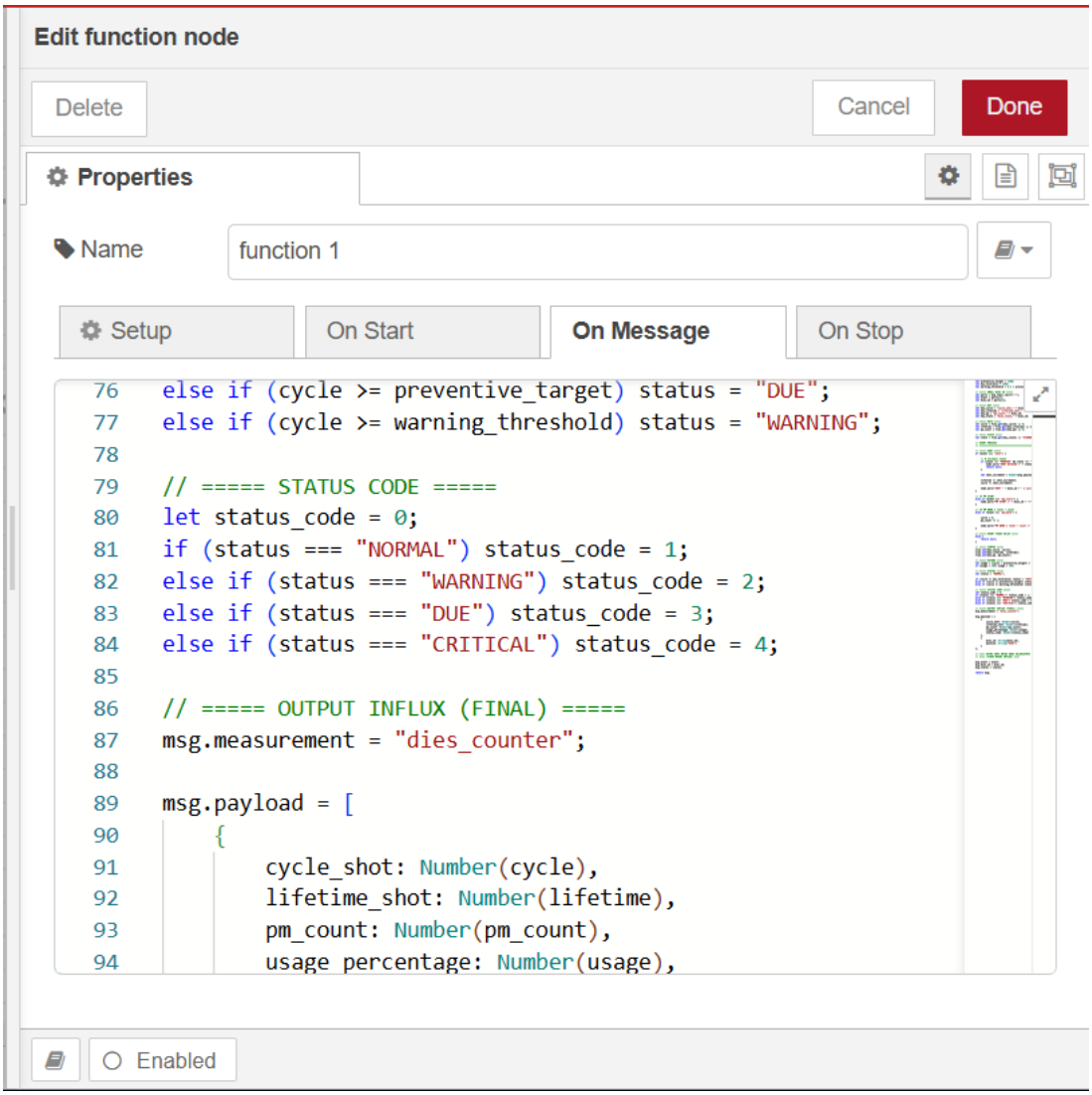
B-Properties Node-RED (Lanjutan)



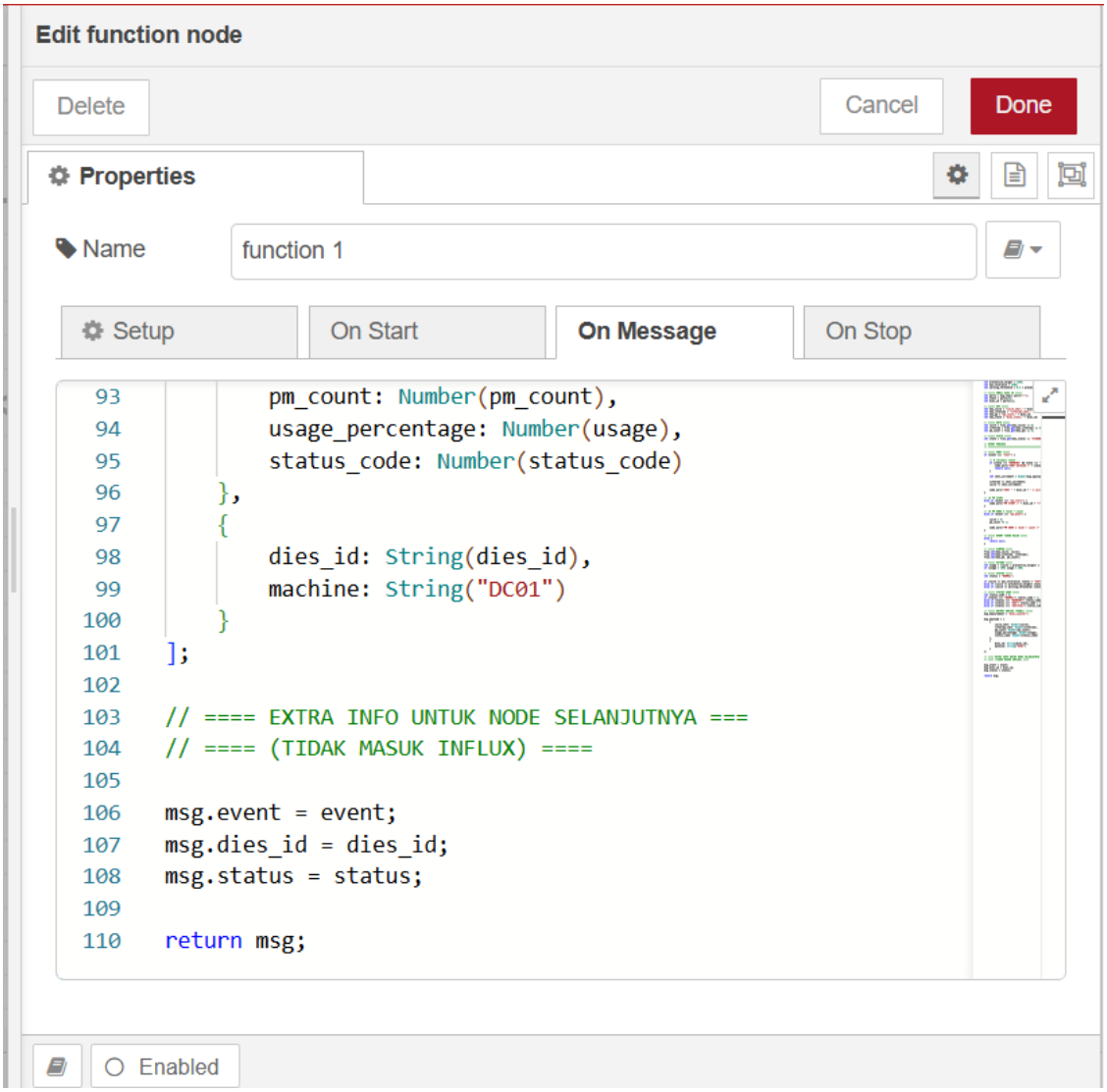
B-Properties Node-RED (Lanjutan)



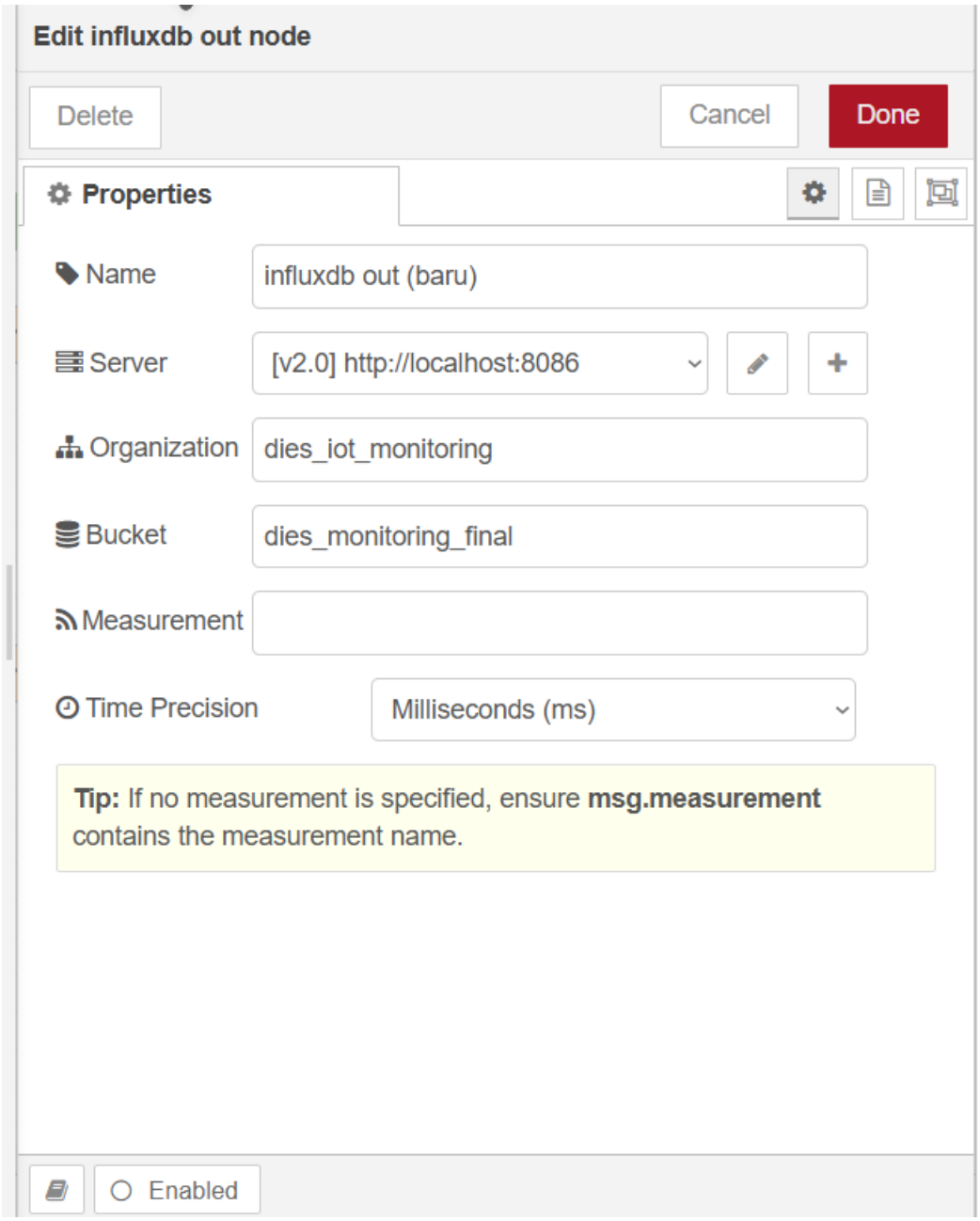
B-Properties Node-RED (Lanjutan)



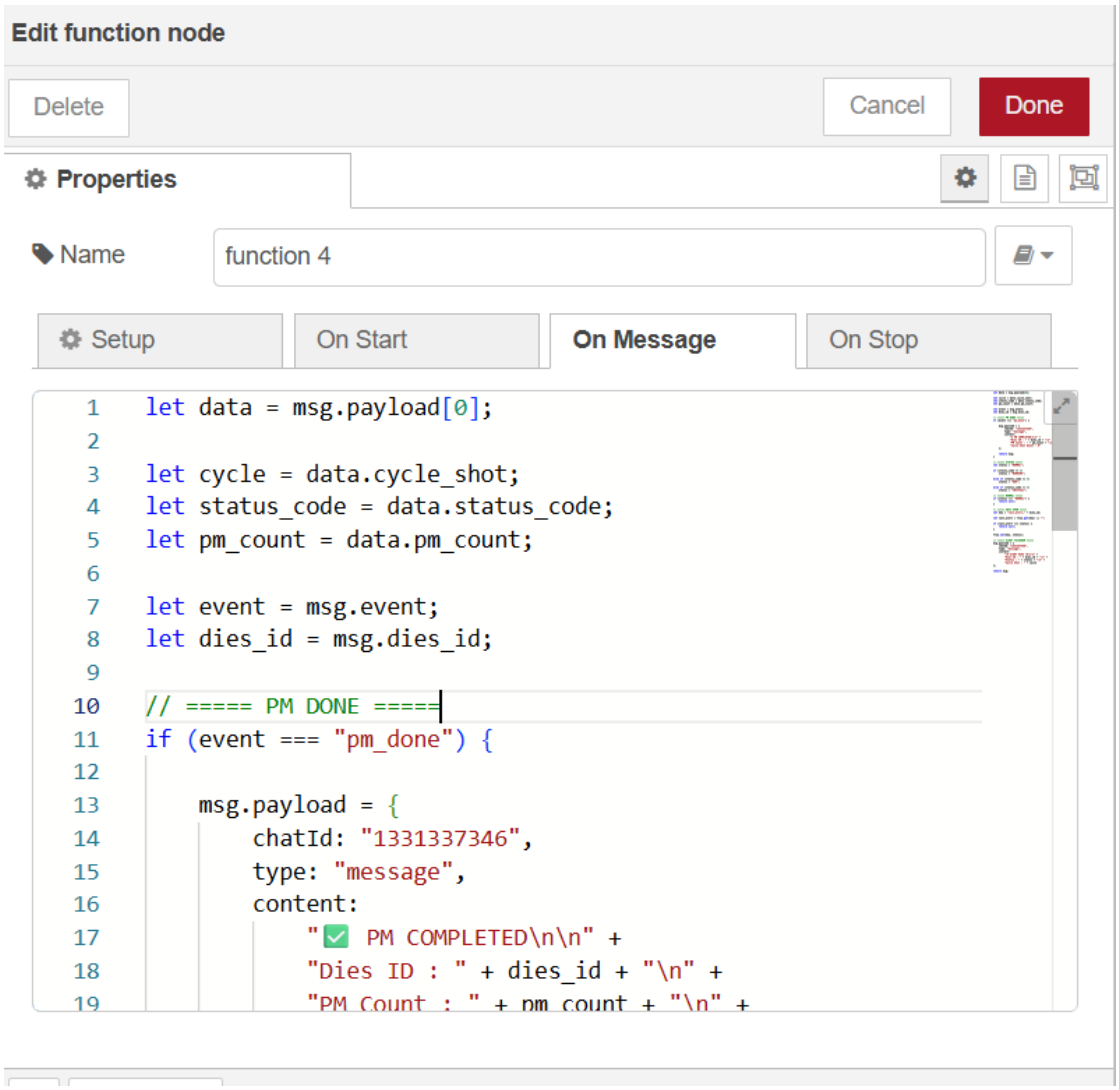
B-Properties Node-RED (Lanjutan)



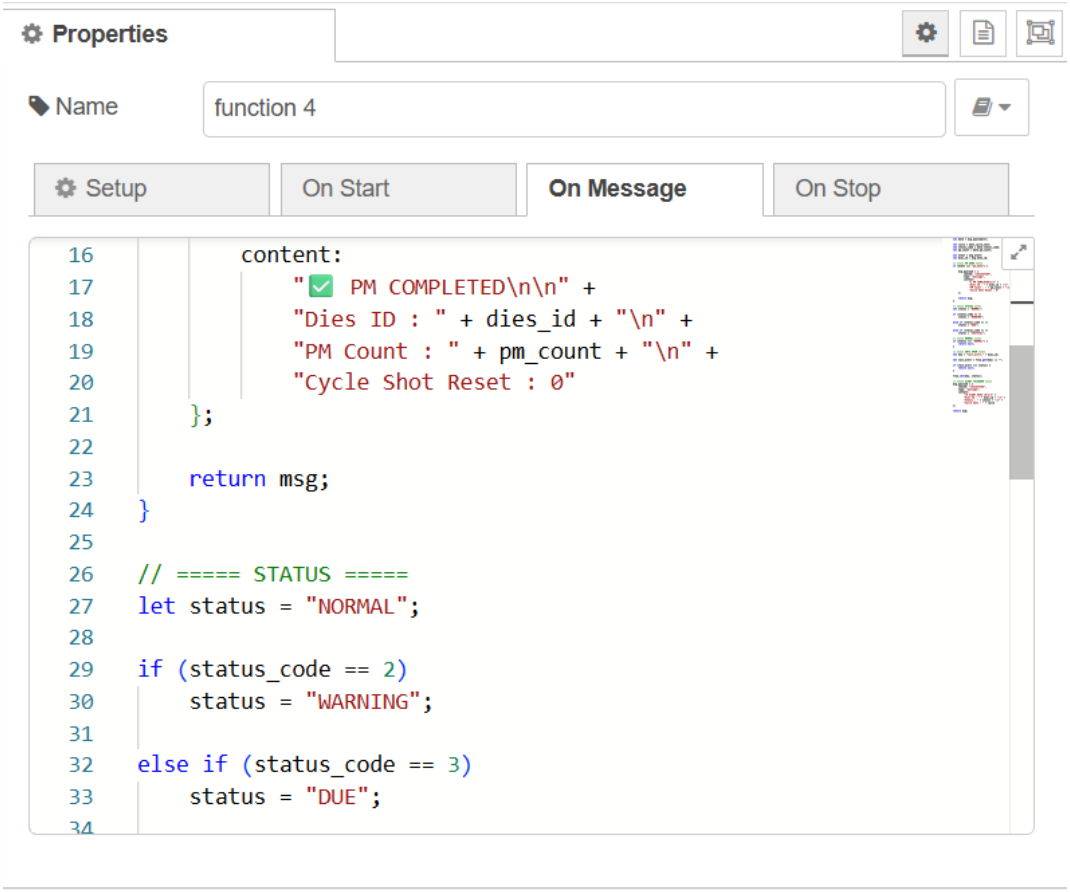
B-Properties Node-RED (Lanjutan)



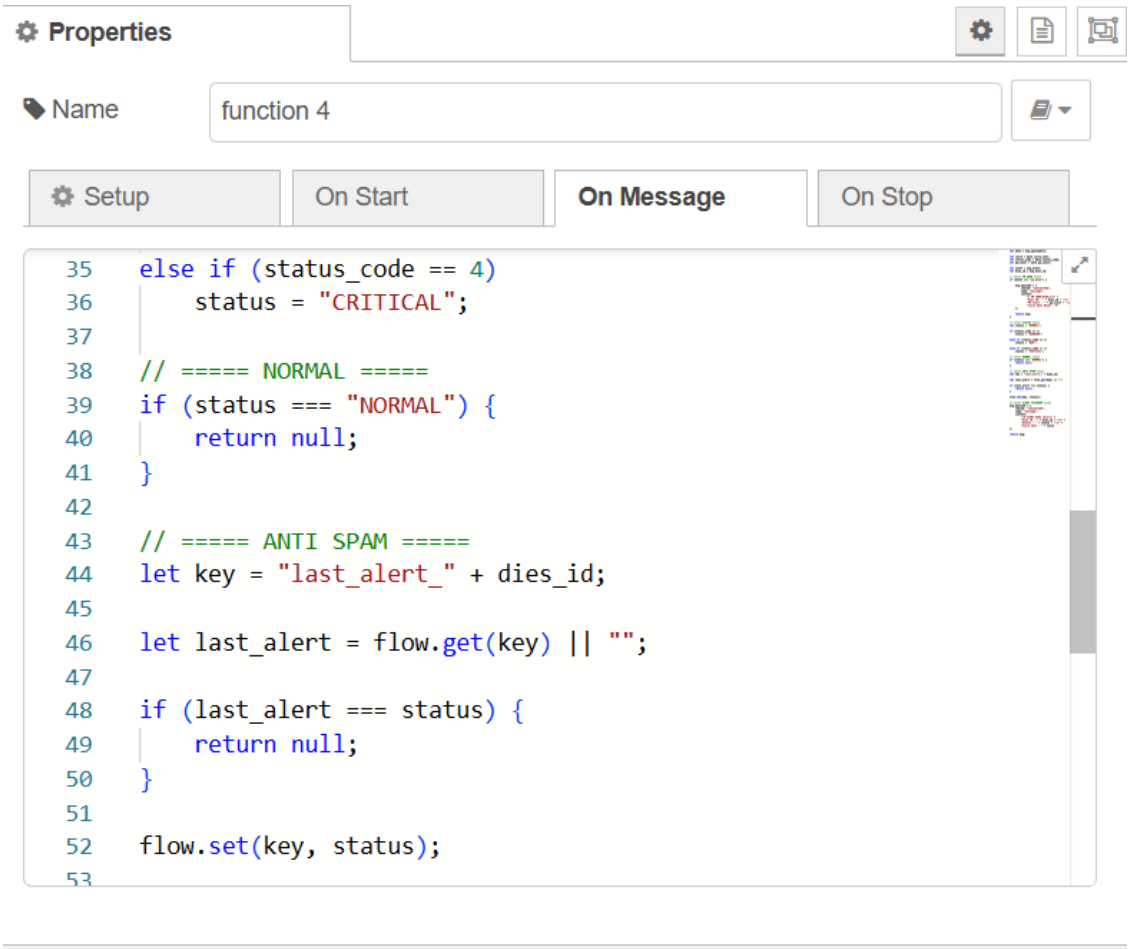
B-Properties Node-RED (Lanjutan)



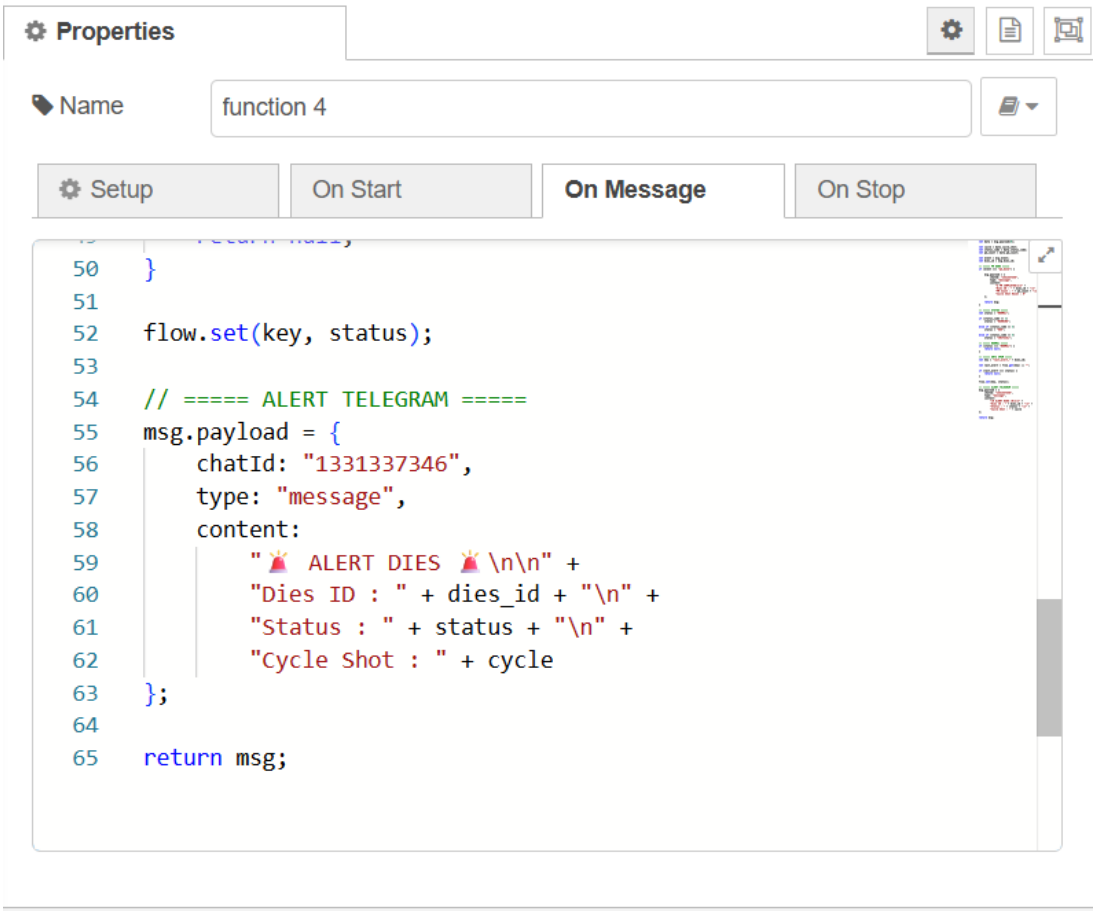
B-Properties Node-RED (Lanjutan)



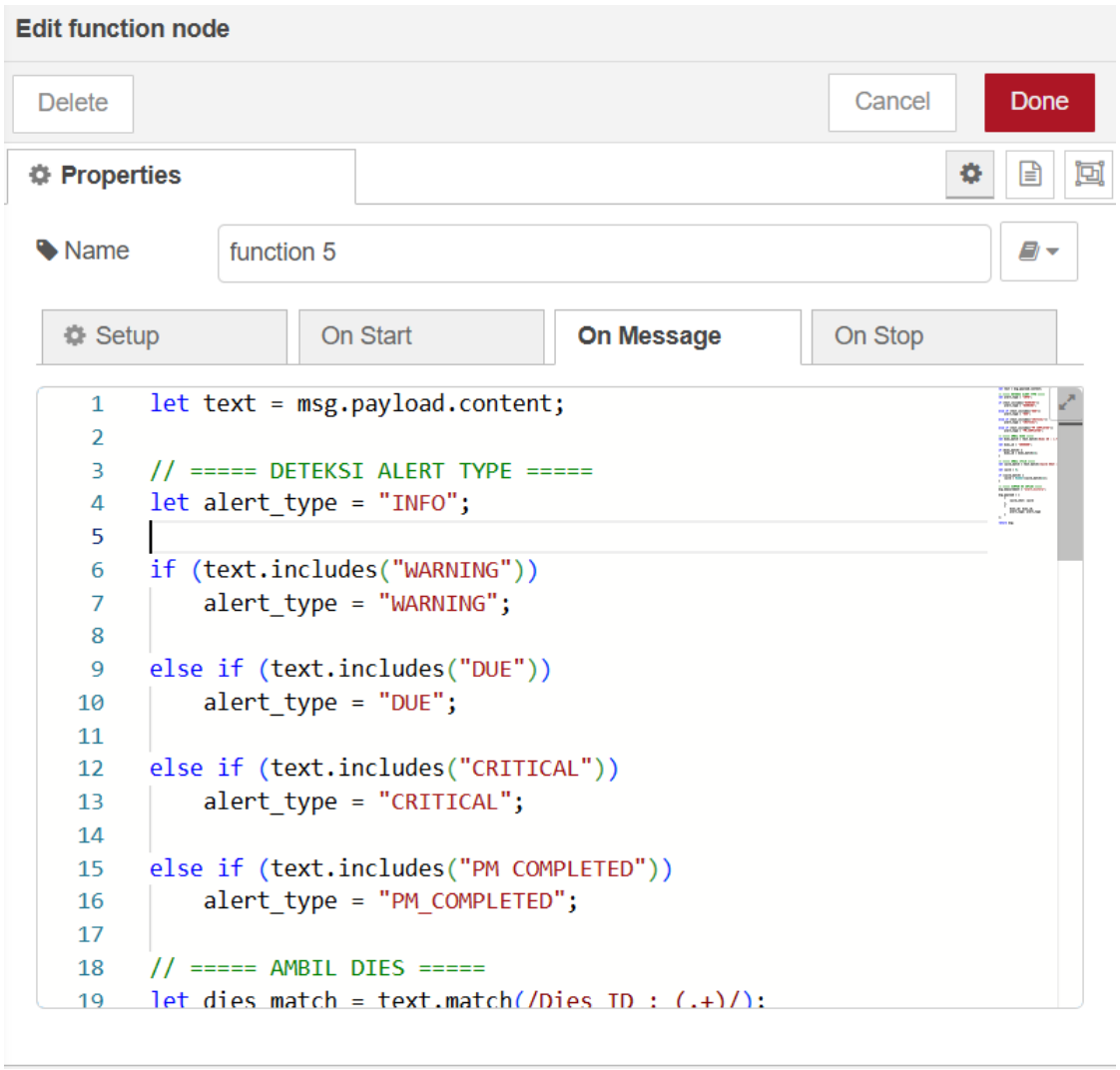
B-Properties Node-RED (Lanjutan)



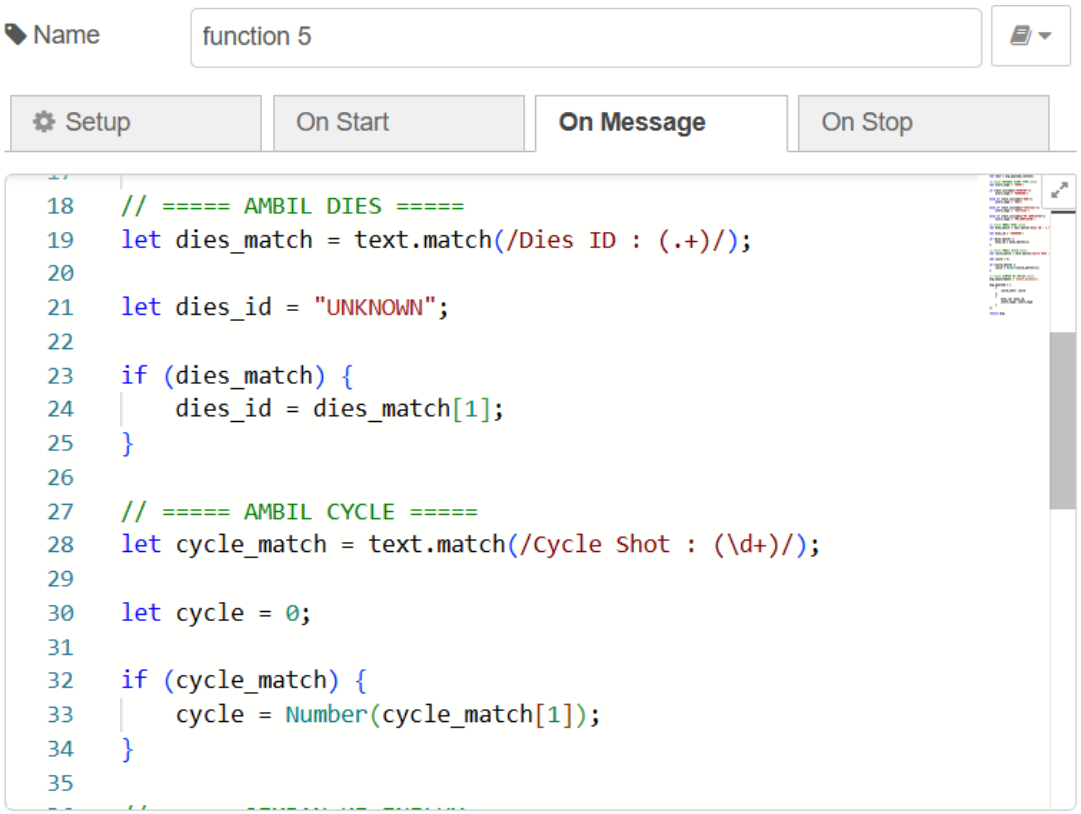
B-Properties Node-RED (Lanjutan)



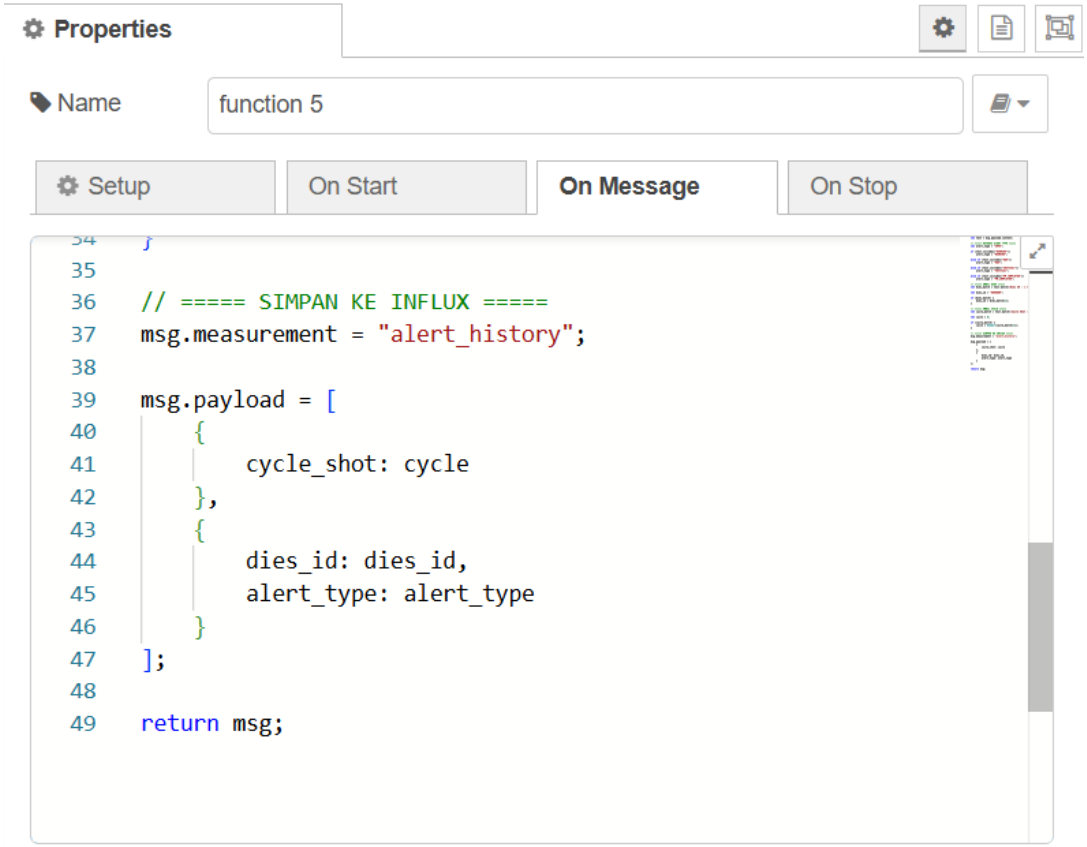
B-Properties Node-RED (Lanjutan)



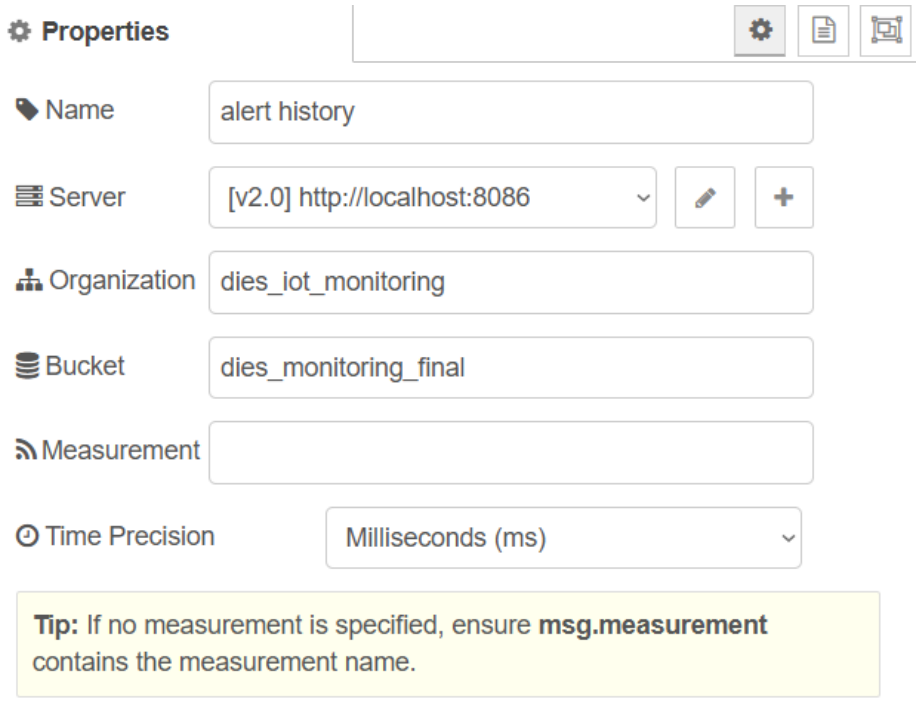
B-Properties Node-RED (Lanjutan)

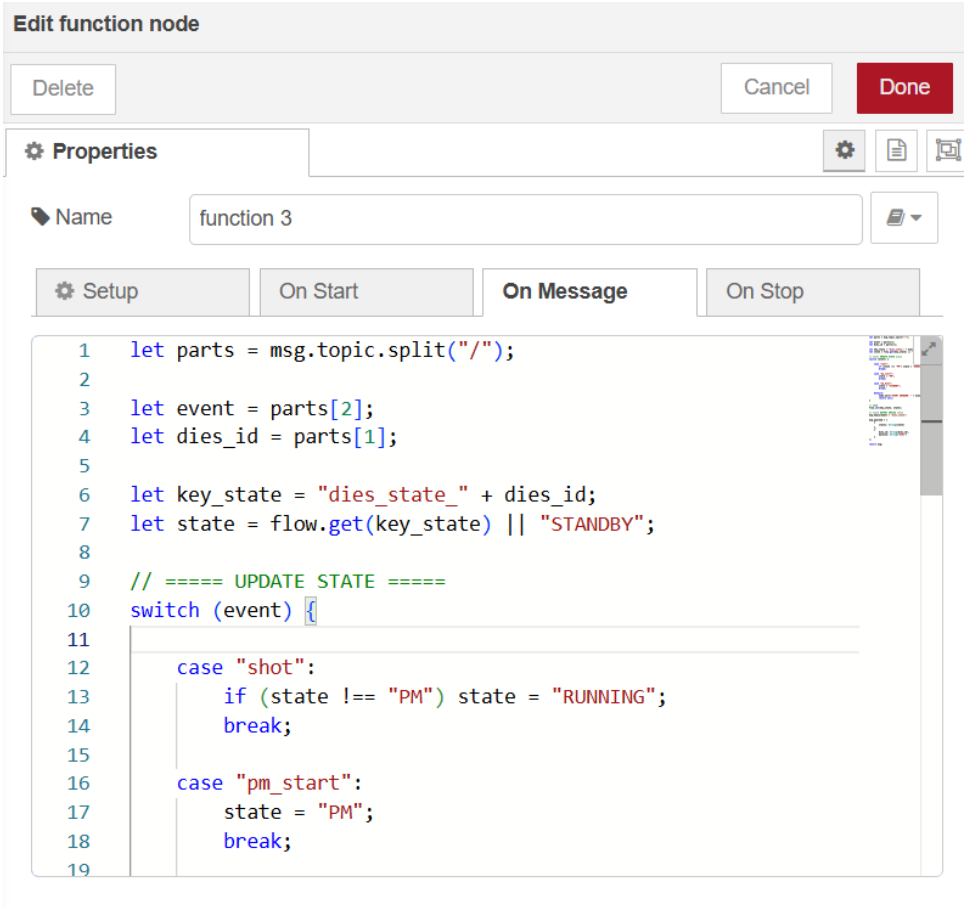


B-Properties Node-RED (Lanjutan)

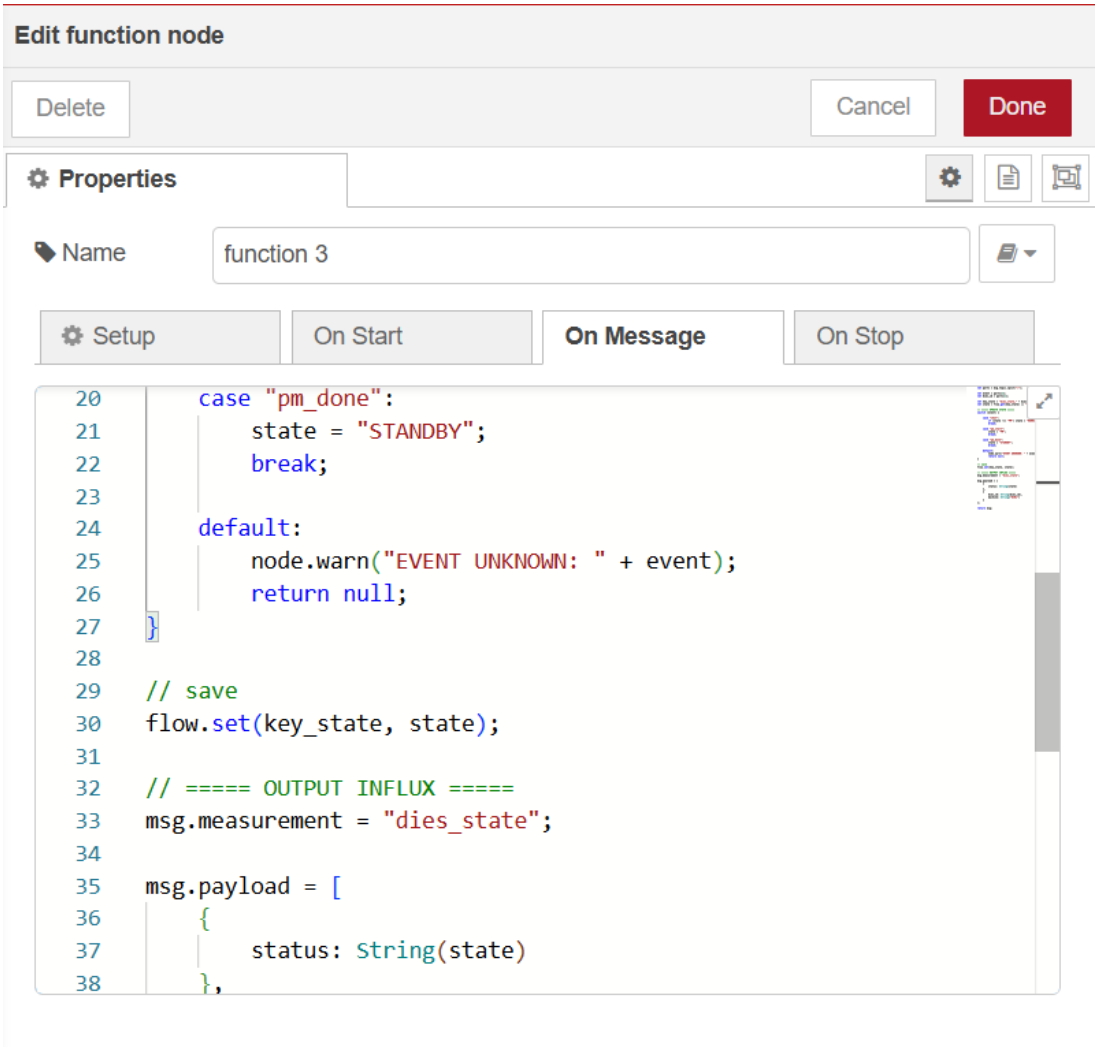


B-Properties Node-RED (Lanjutan)

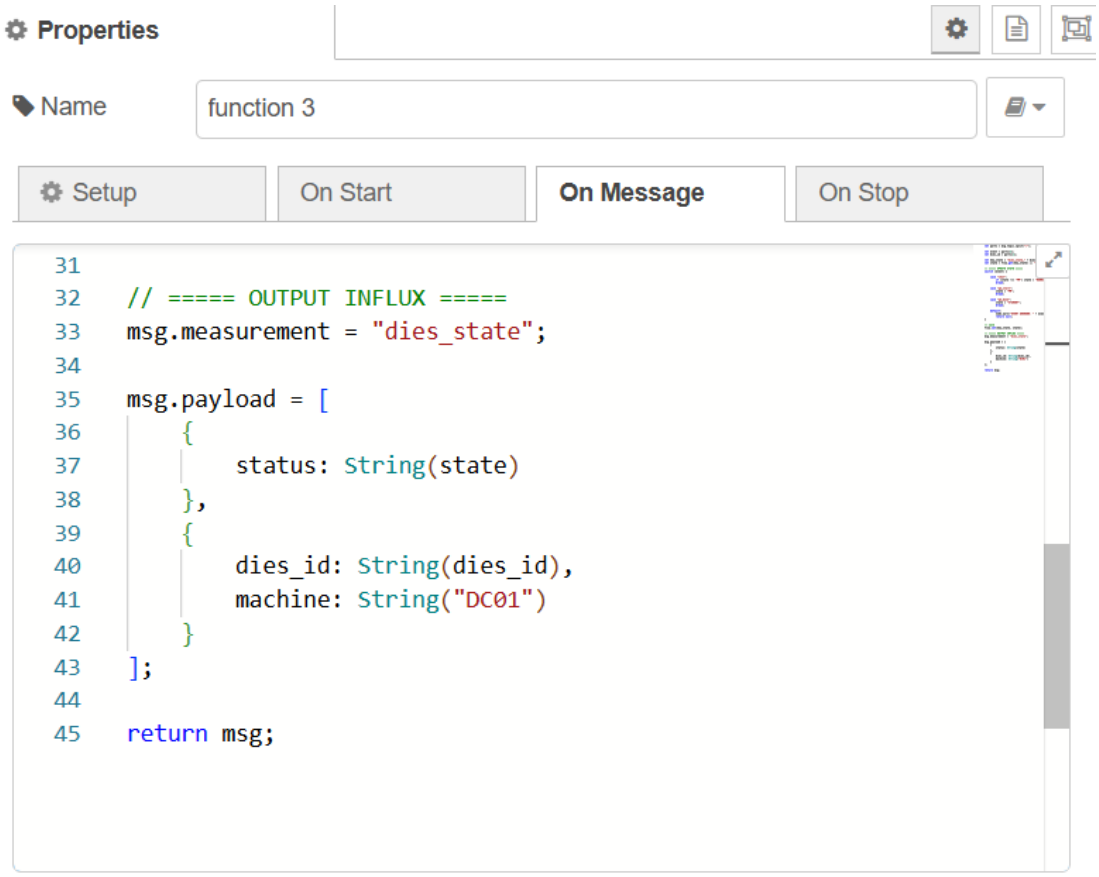




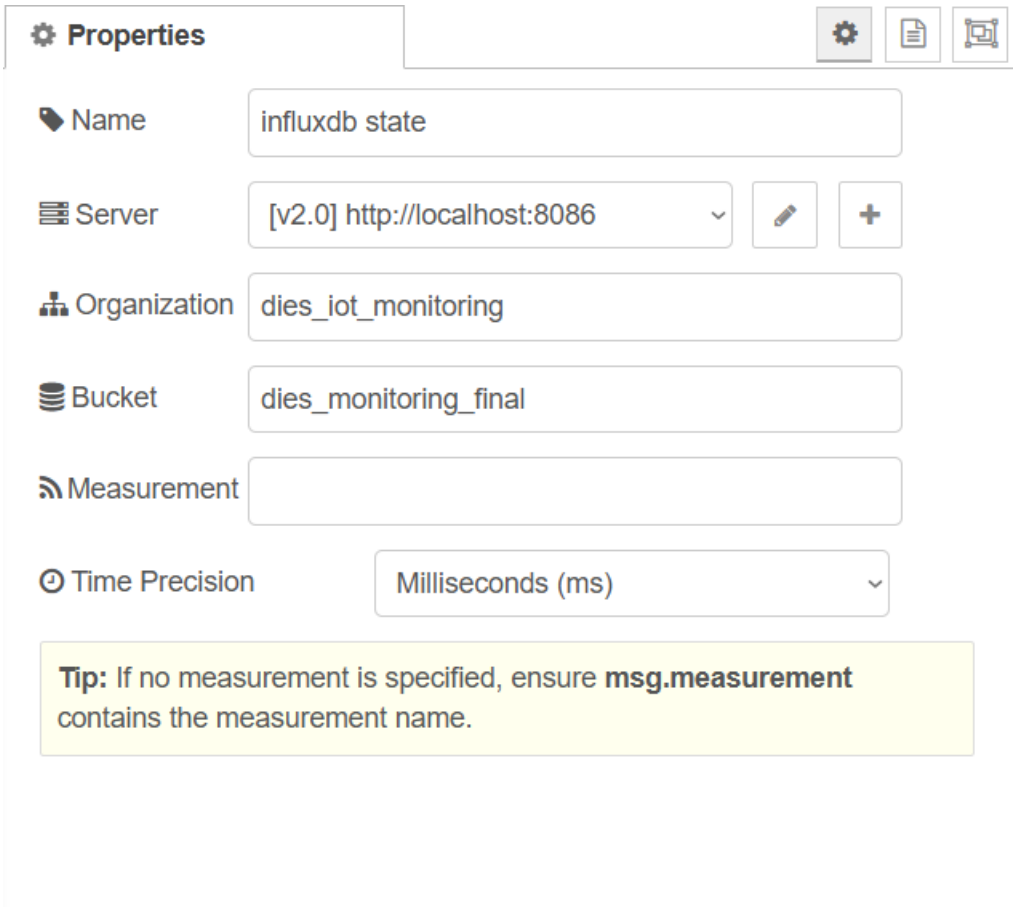
B-Properties Node-RED (Lanjutan)



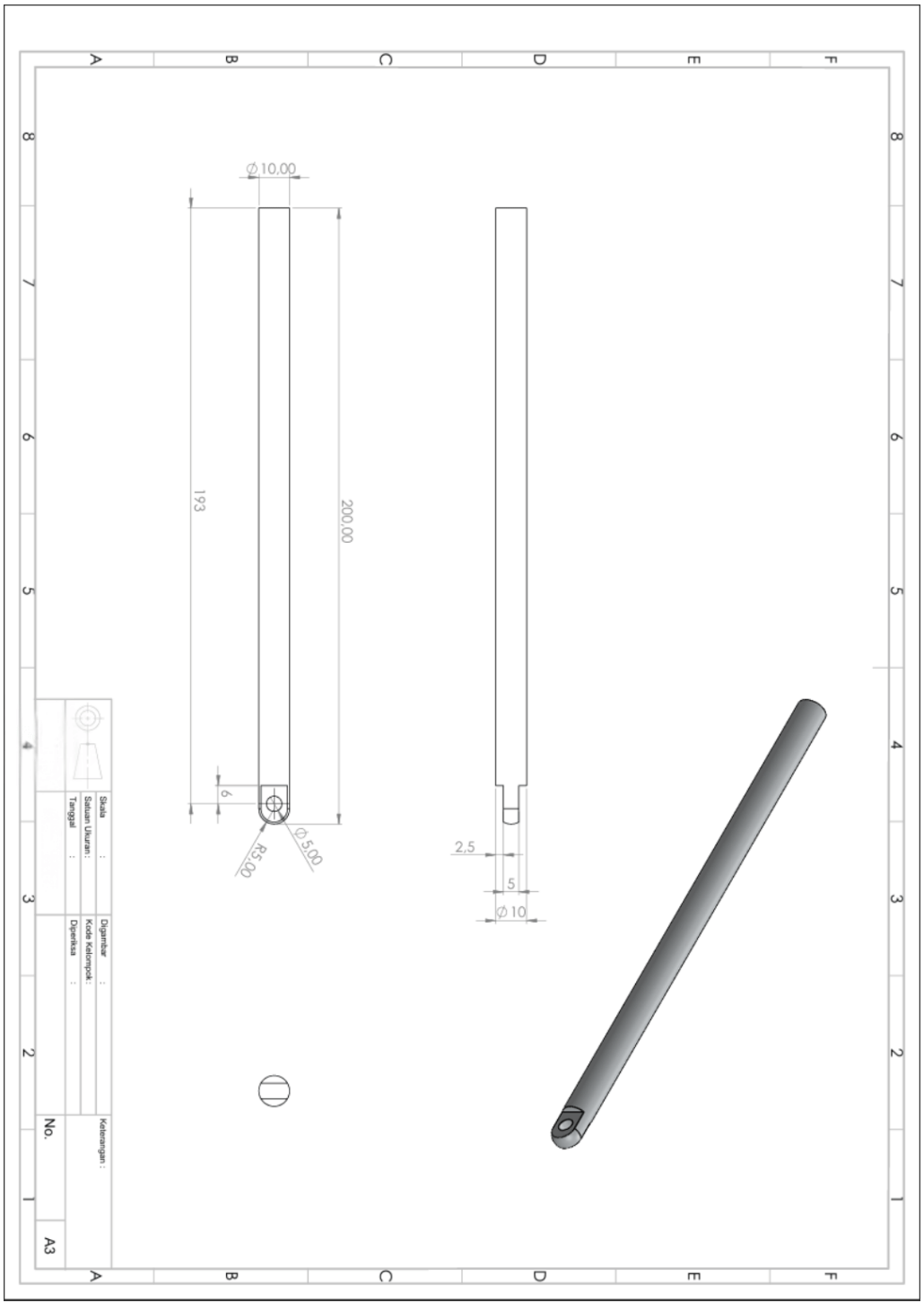
B-Properties Node-RED (Lanjutan)



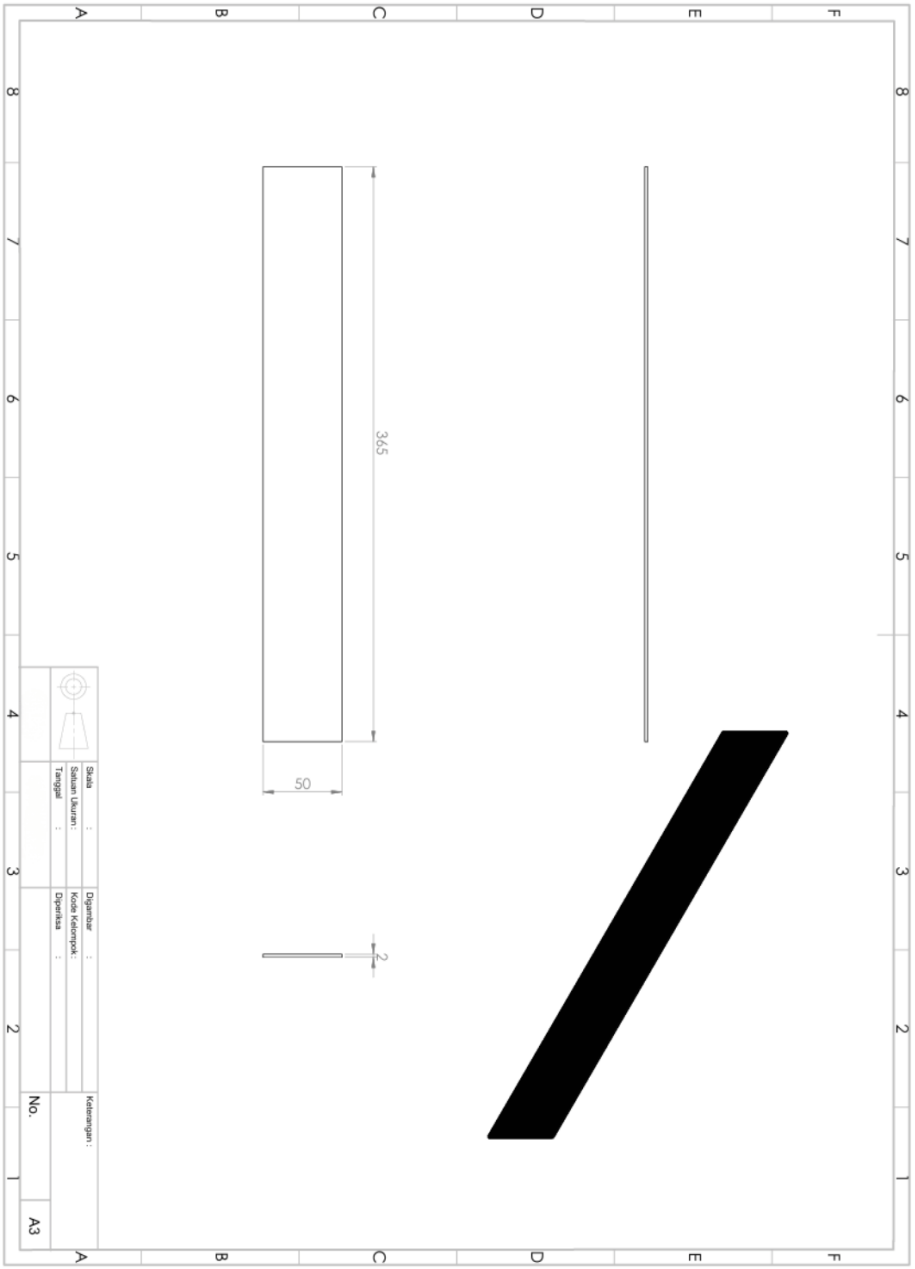
B-Properties Node-RED (Lanjutan)



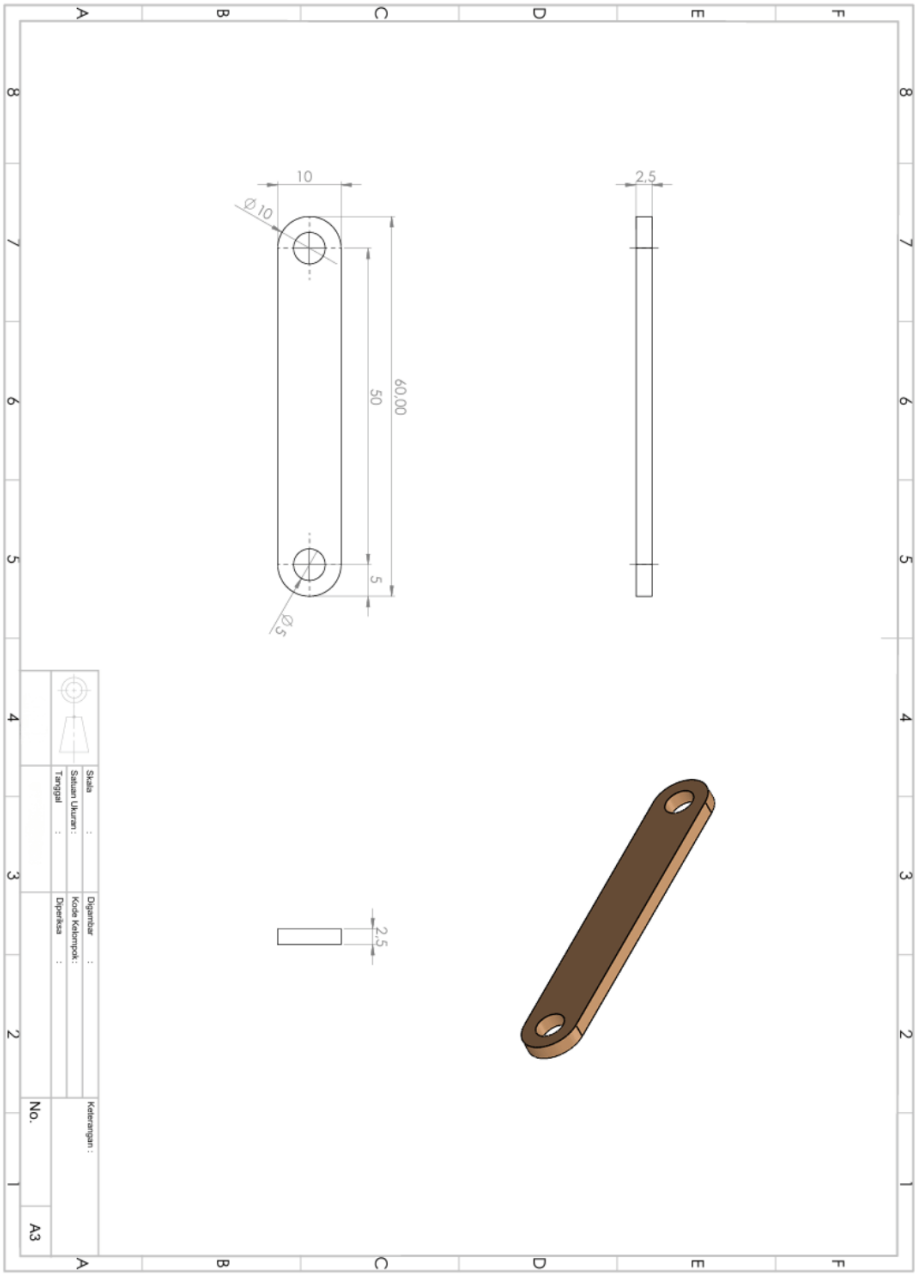
C-Gambar Teknik *Prototype*



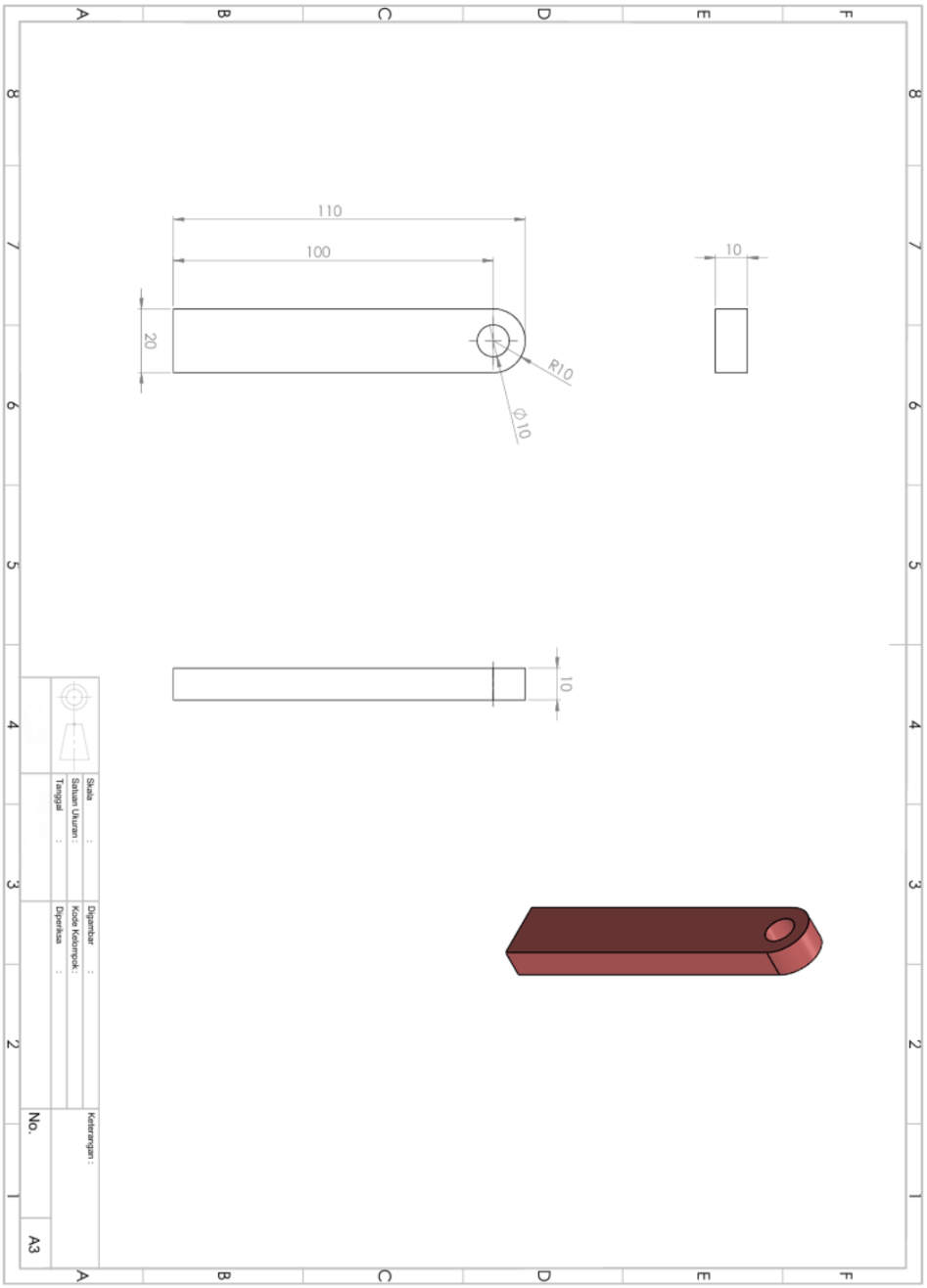
C-Gambar Teknik *Prototype* (Lanjutan)



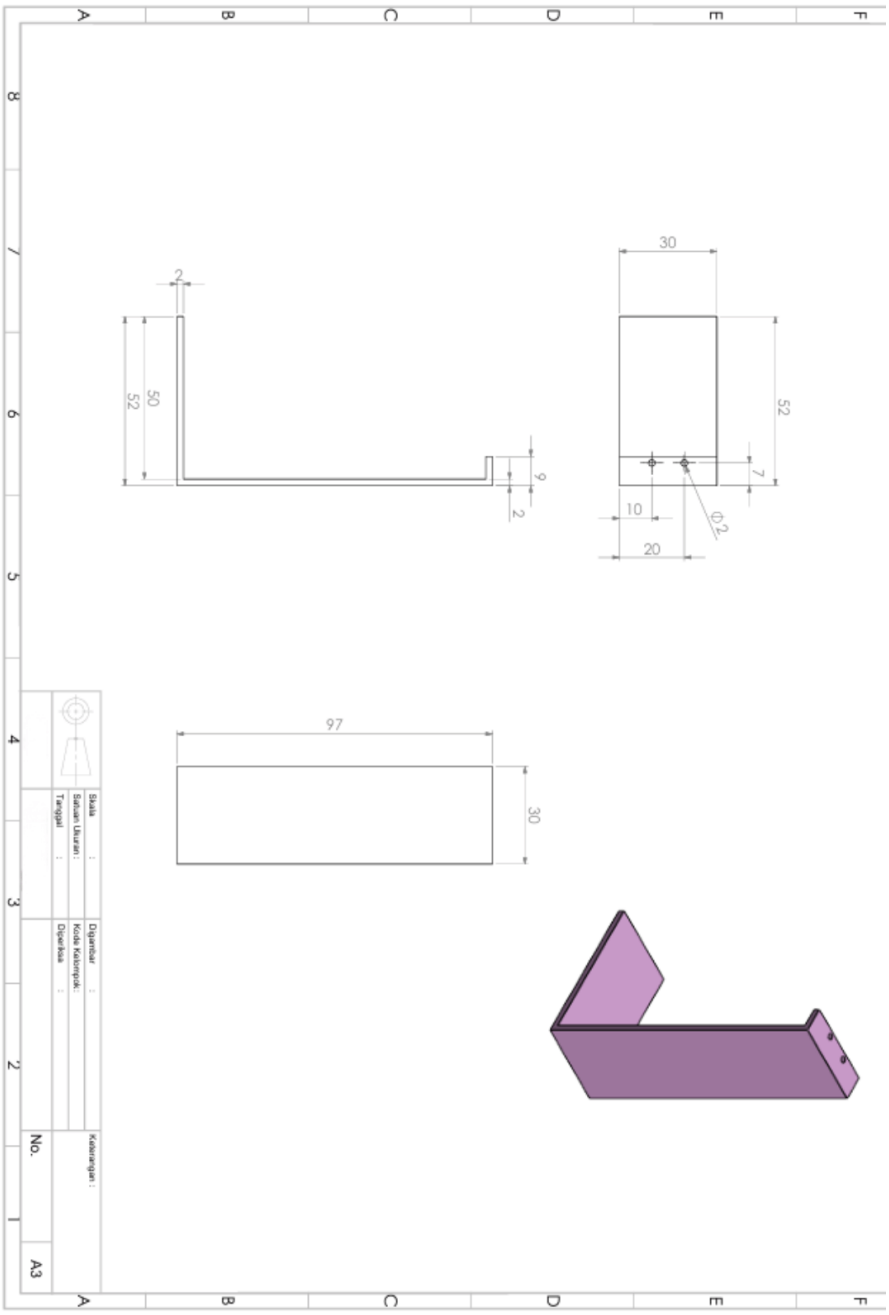
C-Gambar Teknik *Prototype* (Lanjutan)



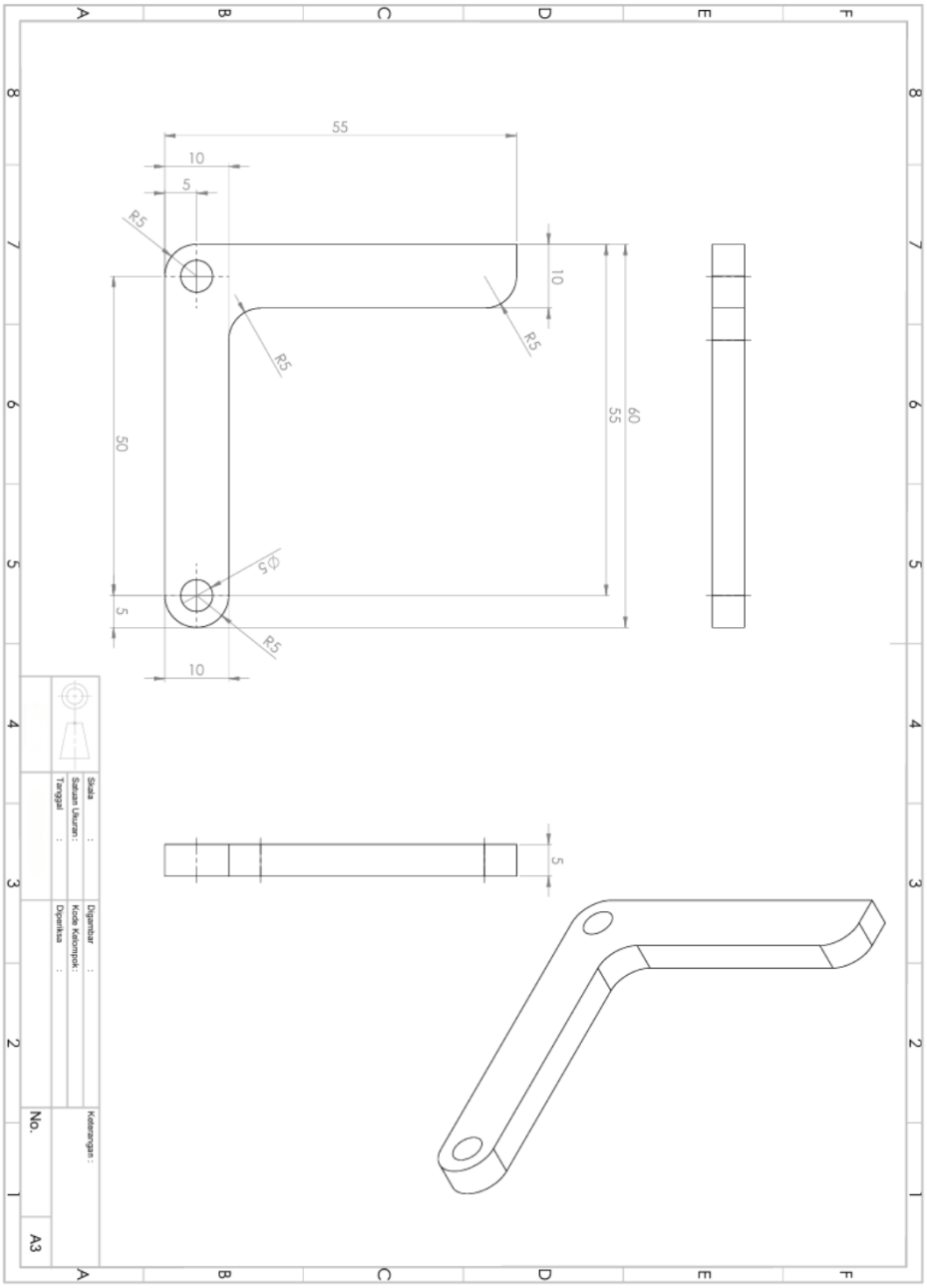
C-Gambar Teknik *Prototype* (Lanjutan)



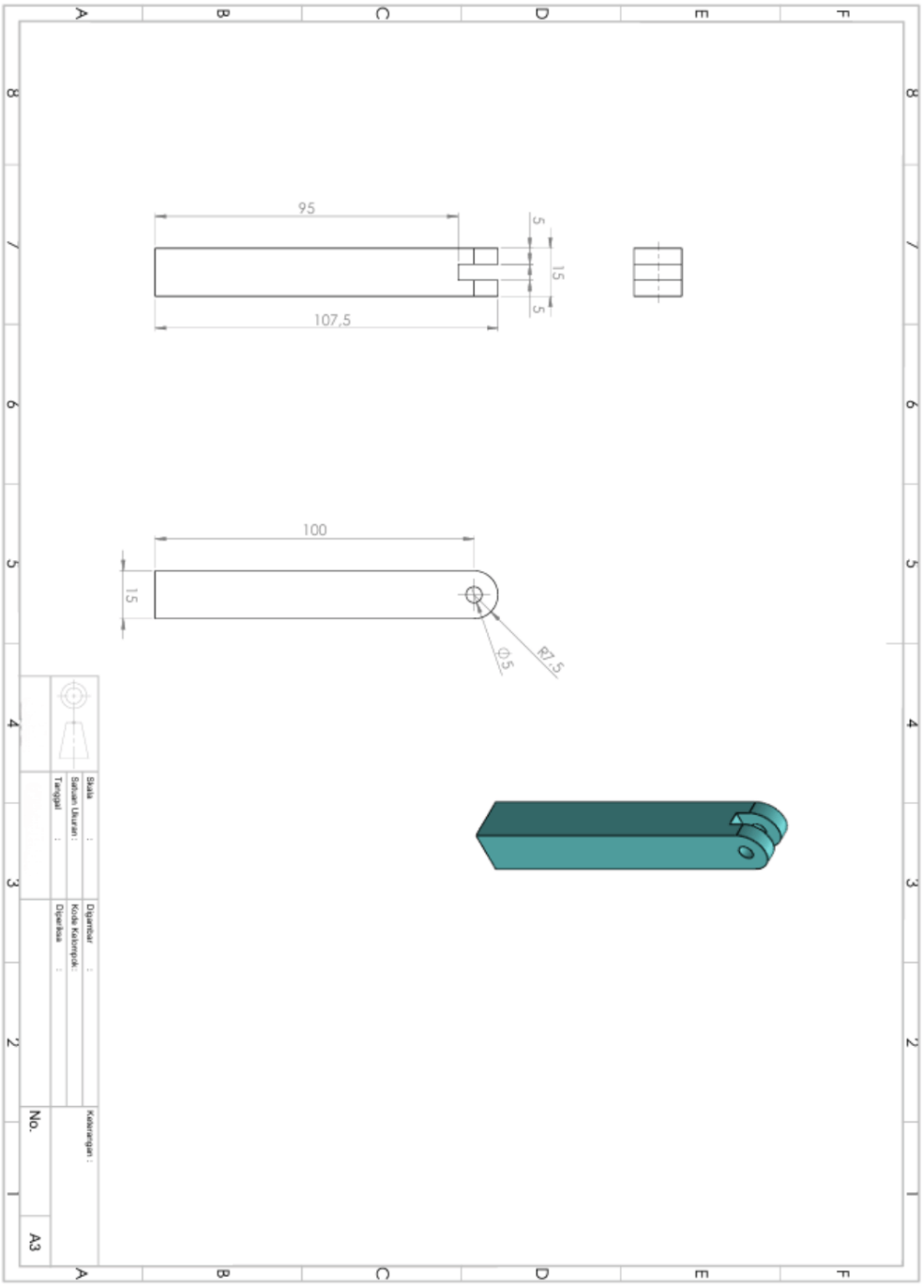
C-Gambar Teknik *Prototype* (Lanjutan)



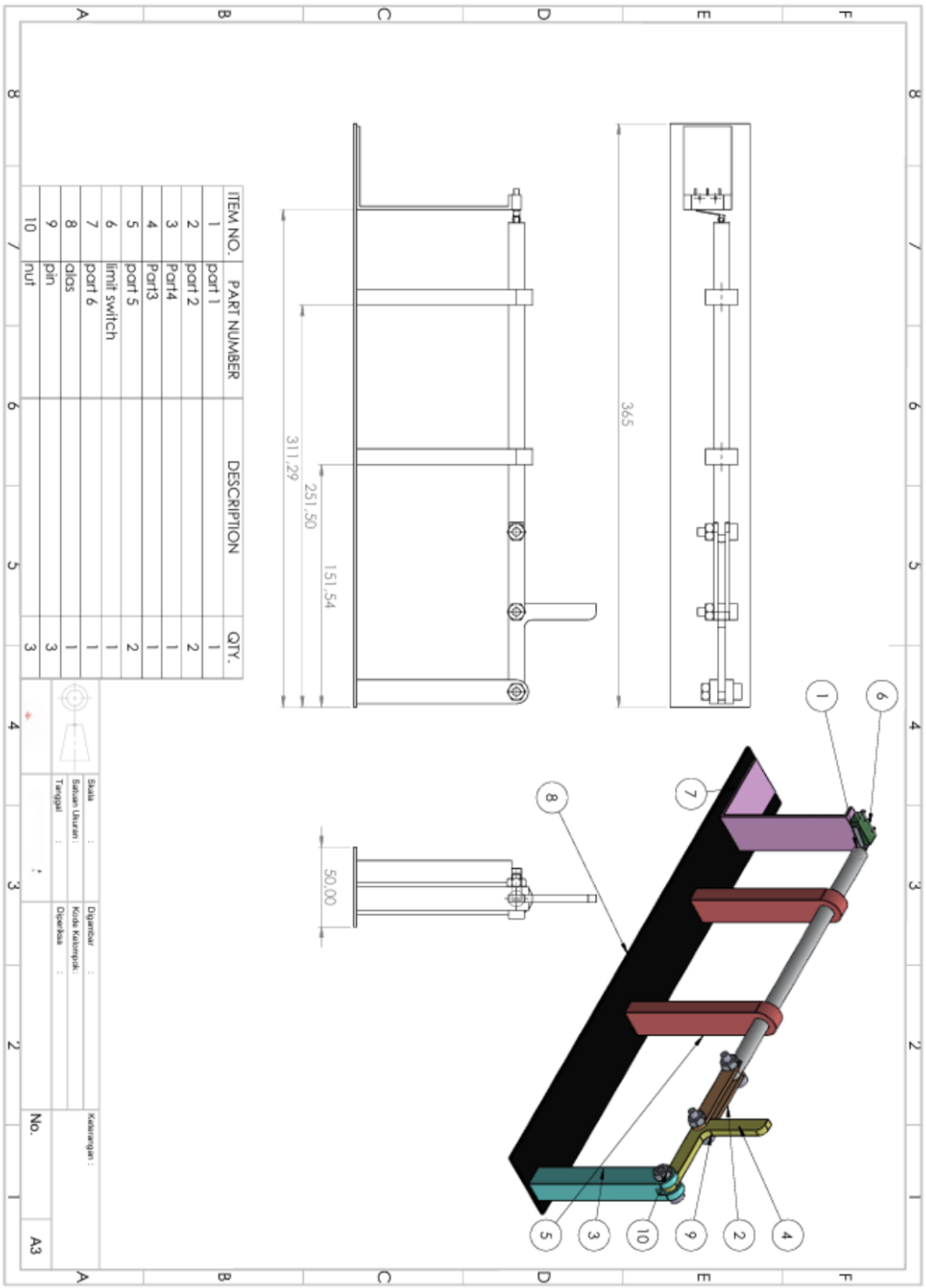
C-Gambar Teknik *Prototype* (Lanjutan)



C-Gambar Teknik *Prototype* (Lanjutan)



C-Gambar Teknik *Prototype* (Lanjutan)



D-Hasil *Prototype*

