OPTIMASI PID *CONTROLLER* PADA MESIN PELAPIS KERTAS MENGGUNAKAN *PARTICLE SWARM OPTIMIZATION* (PSO)

SKRIPSI

Oleh:
TAUFIK ARDIANSYAH PUTRA
NIM. 200605110071

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2024

OPTIMASI PID *CONTROLLER* PADA MESIN PELAPIS KERTAS MENGGUNAKAN *PARTICLE SWARM OPTIMIZATION* (PSO)

SKRIPSI

Diajukan kepada:
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh:
TAUFIK ARDIANSYAH PUTRA
NIM. 200605110071

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2024

HALAMAN PERSETUJUAN

OPTIMASI PID *CONTROLLER* PADA MESIN PELAPIS KERTAS MENGGUNAKAN *PARTICLE SWARM OPTIMIZATION* (PSO)

SKRIPSI

Oleh :
TAUFIK ARDIANSYAH PUTRA
NIM. 200605110071

Telah Diperiksa dan Disetujui untuk Diuji:
Tanggal: 2 Desember 2024

Pembimbing I,	Pembimbing II,
Ajib Hanani, M.T	Dr. Ir. Fachrul Kurniawan, M.MT, IPU
NIP. 19840731 202321 1 013	NIP. 19771020 200912 1 001

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Ir. Fachrul Kurniawan, M.MT, IPU
NIP. 19771020 200912 1 001

HALAMAN PENGESAHAN

OPTIMASI PID *CONTROLLER* PADA MESIN PELAPIS KERTAS MENGGUNAKAN *PARTICLE SWARM OPTIMIZATION* (PSO)

SKRIPSI

Oleh :
TAUFIK ARDIANSYAH PUTRA
NIM. 200605110071

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 5 Desember 2024

Susunan Dewan Penguji

Ketua Penguji	: A'la Syauqi, M.Kom NIP. 19771201 200801 1 007	( )
Anggota Penguji I	: Shoffin Nahwa Utama, M.T NIP. 19860703 202012 1 003	( )
Anggota Penguji II	: Ajib Hanani, M.T NIP. 19840731 202321 1 013	( )
Anggota Penguji III	: Dr. Ir. Fachrul Kurniawan, M.MT, IPU NIP. 19771020 200912 1 001	( )

Mengetahui dan Mengesahkan,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Ir. Fachrul Kurniawan, M.MT, IPU
NIP. 19771020 200912 1 001

PERNYATAAN KEASLIAN TULISAN

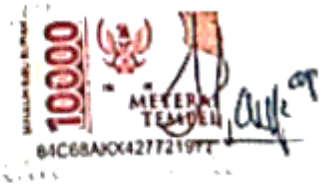
Saya yang bertanda tangan dibawah ini:

Nama	: Taufik Ardiansyah Putra
NIM	: 200605110071
Fakultas / Prodi	: Sains dan Teknologi / Teknik Informatika
Judul Skripsi	: Optimasi PID *Controller* pada Mesin Pelapis Kertas Menggunakan *Particle Swarm Optimization* (PSO)

Menyatakan dengan sebenarnya bahwa Skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini merupakan hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 18 Desember 2024
Yang membuat pernyataan,



Taufik Ardiansyah Putra
NIM. 200605110071

HALAMAN MOTTO

“Fast No Izzi - Beserta kesulitan ada kemudahan, lewati kesulitan, itu hanyalah sejarah yang hanya perlu ditertawakan”

(Taufik Ardiansyah Putra)

HALAMAN PERSEMBAHAN

Alhamdulillahi rabbil 'alamin, dengan penuh rasa syukur saya persembahkan skripsi ini untuk diri saya sendiri, Taufik Ardiansyah Putra, sebagai bentuk penghargaan atas usaha dan dedikasi dalam menyelesaikan proses penyusunan skripsi ini.

Ucapan terima kasih yang mendalam saya sampaikan kepada keluarga saya, Agus Nadi, Mahsunah, Betha Febriliana Putri, Nely Aulia Qudzi, dan Irfan Noer Ramadhan, yang senantiasa menjadi sumber motivasi dan dukungan tanpa henti selama proses ini.

Saya juga ingin mengucapkan terima kasih kepada Diah Ayu Rahma yang selalu membantu, menemani, serta menjadi motivasi dan tujuan utama saya dalam menyelesaikan proyek penelitian ini. Tak lupa, saya sampaikan rasa terima kasih kepada keluarga besar Dasaria, terutama Samsut Tabris, Ilham Saputra, Indra Pradesa, Aris, Awanda Setya, dan Satrio Adi Prakoso, yang dengan setia memberikan bantuan dalam proses pengerjaan skripsi ini. Penghargaan yang tulus juga saya berikan kepada rekan-rekan TI angkatan 2020, terutama Galan Ramadhan, atas kerja sama, dukungan, dan semangat yang telah diberikan.

Terima kasih atas segala bentuk dukungan, pengorbanan, doa, motivasi, dan kasih sayang yang telah menjadi penyemangat terbesar dalam perjalanan ini.

KATA PENGANTAR

Alhamdulillah, segala puji dan syukur ke hadirat Allah SWT atas limpahan rahmat dan karunia-Nya yang telah memampukan saya untuk menyelesaikan skripsi ini dengan judul "Optimasi PID *Controller* pada Mesin Pelapis Kertas Menggunakan *Particle Swarm Optimization* (PSO)." Shalawat serta salam senantiasa tercurah kepada junjungan kita, Nabi Muhammad SAW, semoga kita semua memperoleh syafaat beliau di hari akhir kelak. Aamiin. Proses penyusunan skripsi ini merupakan perjalanan yang penuh dengan tantangan, dan keberhasilannya tidak lepas dari peran serta banyak pihak yang telah memberikan dukungan dan bantuan tanpa batas. Dengan penuh rasa hormat dan syukur, penulis menyampaikan doa dan ucapan terima kasih yang sebesar-besarnya kepada:

1. Prof. Dr. H. M. Zainuddin, M.A., selaku rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Prof. Dr. Hj. Sri Harini, M.Si., selaku dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
3. Dr. Ir. Fachrul Kurniawan, M.MT, IPU, selaku Ketua Program Studi Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang sekaligus sebagai Dosen Pembimbing II saya.
4. Ajib Hanani, M.T., selaku Dosen Pembimbing I saya, yang dengan sabar memberikan arahan dan nasihat selama proses penulisan skripsi ini.
5. A’la Syauqi, M.Kom, S.Si, M.Kom dan Shoffin Nahwa Utama, M.T., selaku Dosen Penguji I dan II, yang telah menguji serta memberikan masukan selama proses penulisan skripsi ini.

6. Para dosen, admin dan mahasiswa Program Studi Teknik Informatika yang telah memberi dukungan selama pengerjaan skripsi ini.
7. Keluarga saya Agus Nadi, Mahsunah, Betha Febriliana Putri, Nely Aulia Qudzi, dan Irfan Noer Ramadhan yang memberikan motivasi dan doa yang menjadi sumber semangat saya.
8. Galan Ramadhan, Awanda Setya, Abel Firmansyah, Razan Ikbaar, Haris Ibram, Falah Abdurrohman, Solikin, Salma Nabila, Aqshal Radanta, Mukhlis Abdurrohman, Ahmad Azhar, dan Onny Putra selaku teman, sahabat, dan rekan yang memberikan dukungan, bantuan dan semangat untuk saya.
9. Diah Ayu Rahma yang selalu memberikan motivasi, semangat, dan kekuatan untuk membantu dan menemani dari awal hingga akhir.

Terakhir, penulis mengakui bahwa masih banyak kekurangan pada skripsi ini. Semoga karya tulis ini dapat memberikan manfaat bagi banyak pihak. Dengan segala kerendahan hati, penulis mengucapkan terima kasih kepada semua yang telah memberikan dukungan, bimbingan, dan doa hingga penelitian terselesaikan. Semoga Allah SWT senantiasa membalas kebaikan yang telah diberikan.

Wassalamualaikum Warahmatullahi Wabarakatuh.

Malang, 18 Desember 2024

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Putra, Taufik Ardiansyah. 2024. **Optimasi PID *Controller* pada Mesin Pelapis Kertas Menggunakan Particle Swarm Optimization (PSO).** Skripsi. Program Studi Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim, Malang. Pembimbing: (I) Ajib Hanani, M.T (II) Dr. Ir. Fachrul Kurniawan, M.MT, IPU.

Kata kunci: Automasi Industri, IoT, Optimasi, PID, PSO

Industri manufaktur terus berkembang pesat dengan adopsi teknologi canggih, termasuk automasi berbasis algoritma *Proportional Integral Derivative* (PID) dan *Internet of Things* (IoT). Salah satu tantangan utama dalam proses industri adalah meningkatkan efisiensi dan konsistensi, seperti pada tahapan pelapisan kertas fotografi yang sering menghadapi kendala operasional. Penelitian ini mengimplementasikan algoritma PID untuk mengontrol proses pelapisan secara otomatis, dengan tuning parameter menggunakan metode *Particle Swarm Optimization* (PSO). Berdasarkan evaluasi, skenario optimal ditemukan pada konfigurasi dengan parameter PID K_p = 90.74, K_i= 0.16, dan K_d = 0.49, menghasilkan nilai Integral of *Time-weighted Absolute Error* (ITAE) sebesar 1,56. Parameter ini memberikan kinerja terbaik dengan *rise tim*e 0,25 detik, *settling time* 1,89 detik, *overshoot* 0,02%, dan steady-state error 0,00002%. Selain itu, integrasi dengan platform IoT berbasis *Blynk* memungkinkan pemantauan parameter produksi secara *real-time*, meningkatkan fleksibilitas dan respon cepat dalam pengelolaan produksi. Penerapan teknologi ini tidak hanya memberikan manfaat teknis tetapi juga relevan secara spiritual, mencerminkan nilai-nilai kebaikan dan kemudahan dalam Islam. Penelitian ini memberikan kontribusi signifikan terhadap efisiensi dan kualitas operasional perusahaan manufaktur.

ABSTRACT

Putra, Taufik Ardiansyah. 2024. **Optimasi PID *Controller* pada Mesin Pelapis Kertas Menggunakan Particle Swarm Optimization (PSO).** Thesis. Department of Informatics Engineering, Faculty of Science and Technology, Maulana Malik Ibrahim State Islamic University, Malang. Advisors: (I) Ajib Hanani, M.T (II) Dr. Ir. Fachrul Kurniawan, M.MT, IPU.

The manufacturing industry continues to grow rapidly with the adoption of advanced technologies, including automation based on Proportional Integral Derivative (PID) algorithms and the Internet of Things (IoT). One of the main challenges in industrial processes is improving efficiency and consistency, as seen in the coating stage of photographic paper, which often encounters operational issues. This study implements a PID algorithm to control the coating process automatically, with parameter tuning using the Particle Swarm Optimization (PSO) method. Based on the evaluation, the optimal scenario was found in the configuration with PID parameters of K_p = 90.74, K_i = 0.16, and K_d = 0.49, resulting in an Integral of Time-weighted Absolute Error (ITAE) value of 1.56. These parameters provide the best performance with a rise time of 0.25 seconds, a settling time of 1.89 seconds, an overshoot of 0.02%, and a steady-state error of 0.00002%. Furthermore, integration with the Blynk-based IoT platform enables real-time monitoring of production parameters, enhancing flexibility and responsiveness in production management. The application of this technology not only offers technical benefits but also reflects spiritual relevance, embodying values of goodness and ease in Islam. This study makes a significant contribution to the efficiency and operational quality of manufacturing companies.

Keywords: Industrial Automation, IoT, Optimization, PID, PSO

مستخلص البحث

بوترا، توفيق أرديانشاه. 2024. تحسين وحدة تحكم PID في آلة طلاء الورق باستخدام تحسين سرب الجسيمات. بحث العلمي. قسم هندسة المعلوماتية، كلية العلوم والتكنولوجيا، جامعة مولانا مالك إبراهيم الإسلامية الحكومية، مالانج. المشرف : (الأول) عجيب حناني الماجستير (الثاني) الدكتور المهندس فخرول كورنياوان الماجستير

مع اعتماد التكنولوجيا المتقدمة والصناعة التحويلية لا تزال تنمو بسرعة، بما في ذلك أتمتة على أساس التكامل النسبي التفاضل وخوارزمية إنترنت الأشياء .ومن التحديات الرئيسية في العملية الصناعية هو تحسين الكفاءة والاتساق، كما في كثير من الأحيان تواجه عملية السيطرة على مستوى تصفية ورق الصور الفوتوغرافية. يطبق هذا البحث خوارزمية المشتق التكاملي التناسبي للتحكم في عملية الطلاء تلقائيا عن طريق ضبط المعلمات باستخدام طريقة (PSO)، وفقا للتقييم ، أفضل خطة تم العثور عليها في تكوين PID المعلمات Kp = 90.74, Ki= 0.16 و Kd = 0.49، مما أدى إلى 1,56 الوقت المرجحة خطأ مطلق لا يتجزأ (ITAE). هذه المعلمة لديها الأداء الأمثل وقت الارتفاع هو 0,25 ثانية ، وقت الاستقرار هو 1,89 ثانية ، تخطى 0,02 في المائة وثبات الخطأ هو 0,00002 في المائة. وبالإضافة إلى ذلك ، فإن التكامل مع منصة الإنترنت من الأمور على أساس بلينك يمكن رصد المعلمات الإنتاج في الوقت الحقيقي، وتحسين المرونة والقدرة على الاستجابة السريعة في إدارة الإنتاج . هذه التكنولوجيا ليس فقط الفوائد التقنية ، ولكن له أيضا أهمية روحية، تصوير قيم الخير والراحة في الإسلام. هذا البحث قد قدمت مساهمة كبيرة في كفاءة وجودة التشغيل من شركات التصنيع .

الكلمات الرئيسية :الأتمتة الصناعية، إنترنت الأشياء، التحسين، PID، PSO

BAB I

PENDAHULUAN

1.1 Latar Belakang

Industri manufaktur berkembang pesat berkat kemajuan teknologi, mengubah proses produksi dari manual menjadi sistem canggih yang otomatis, terhubung, dan efisien (Vivin Zulfa A., et al., 2020). Namun, tantangan seperti biaya operasional tinggi dan persaingan global memaksa perusahaan untuk segera mengadopsi teknologi mutakhir, seperti otomasi, yang mengintegrasikan sistem mekanik dan elektronik untuk meningkatkan akurasi dan efisiensi (Anaam, I. K., et al., 2022). Penundaan dalam penerapan ini berisiko membuat metode konvensional usang dan mengurangi daya saing perusahaan di pasar yang semakin kompetitif.

Dalam konteks ini, teknologi *Internet of Things* (IoT) menjadi salah satu inovasi penting yang memungkinkan sistem terhubung dengan lebih baik, memberikan kontrol yang optimal, serta meningkatkan efisiensi proses produksi. Namun implementasi teknologi seperti IoT menghadirkan tantangan spesifik pada setiap proses industri. Salah satu contohnya pada pengelolaan kertas fotografi.

Salah satu proses penting untuk meningkatkan kualitas dan ketahanan produk yang seringkali menghadapi kendala operasional adalah tahap pelapisan kertas. Mesin pelapis kertas fotografi yang digunakan terdiri dari wadah cairan pelapis, selang, dan penggulung kertas. Pengaturan secara manual pada keran mesin menghasilkan keluaran yang tidak konsisten, memakan waktu, dan memperlambat produksi (Mustafa & Hashim, 2020). Untuk mengatasi masalah ini, teknologi automasi berbasis *Proportional Integral Derivative* (PID) dapat digunakan.

Jika penelitian ini tidak dilakukan, perusahaan manufaktur yang masih menggunakan metode manual berisiko menghadapi peningkatan biaya operasional, penurunan kualitas produk, dan operasional yang kompleks, sehingga menyulitkan mereka bersaing di pasar global. Sebaliknya, implementasi teknologi modern dapat meningkatkan efisiensi dan kualitas produksi, sekaligus memberikan dampak positif secara ekonomi dan sosial. Penelitian ini berpotensi mendukung transformasi digital di industri manufaktur, membantu perusahaan tidak hanya bertahan tetapi juga berkembang di tengah persaingan global yang dinamis.

Sistem PID memanfaatkan sensor kecepatan aliran cairan, sensor putaran roda penggulung, dan aktuator motor servo pada keran untuk mengontrol proses pelapisan secara otomatis dan *real-time*. Algoritma ini dipilih karena kesederhanaannya serta kemampuannya dalam menghasilkan *steady-state error* yang rendah dan parameter kontrol yang optimal (Yumurtaci, M., & Verim, Ö., 2020). Dengan implementasi ini, standar keluaran yang konsisten dapat dicapai, sehingga meningkatkan efisiensi sekaligus kualitas produksi.

Untuk memastikan PID bekerja secara optimal, *tuning* parameter menjadi langkah penting. Salah satu metode tuning modern yang efektif adalah *Particle Swarm Optimization* (PSO). Metode ini terkenal karena kemampuannya yang stabil pada ruang pencarian kompleks. PSO telah terbukti berhasil diaplikasikan pada berbagai sistem kontrol, seperti pengendalian kendaraan (Kusuma, et al., 2017) dan sistem kontrol *tracking photovoltaic* (Ali et al., 2021).

Selain kontrol otomatis, pemanfaatan teknologi IoT seperti platform *Blynk* memberikan keunggulan tambahan dalam hal pemantauan dan pengendalian

proses. Dengan aplikasi berbasis *Blynk*, operator dapat memantau parameter produksi secara *real-time* dari perangkat *mobile*. Hal ini meningkatkan efisiensi pengawasan, memungkinkan respons cepat terhadap perubahan, serta memberikan fleksibilitas yang lebih tinggi dalam pengelolaan produksi (Lobur, M., et al., 2019).

Penerapan teknologi automasi berbasis IoT tidak hanya memberikan manfaat teknis, tetapi juga memiliki nilai spiritual yang relevan dalam Islam. Seperti yang dijelaskan dalam hadits yang diriwayatkan oleh Imam Muslim dari Abu Hurairah, membantu memudahkan kesulitan orang lain akan mendatangkan kemudahan dari Allah di dunia maupun akhirat:

مَنْ نَفَّسَ عَنْ مُؤْمِنٍ كُرْبَةً مِنْ كُرَبِ الدُّنْيَا نَفَّسَ اللَّهُ عَنْهُ كُرْبَةً مِنْ كُرَبِ يَوْمِ الْقِيَامَةِ وَمَنْ يَسَّرَ عَلَى مُعْسِرٍ يَسَّرَ اللَّهُ عَلَيْهِ فِي

"Siapa yang menyelesaikan kesulitan seorang mukmin dari berbagai kesulitan dunia, Allah akan memudahkan urusan-urusan kesulitannya di akhirat. Siapa yang mempermudah orang yang dalam kesulitan, Allah akan mempermudah urusan dunia dan akhirat baginya" (HR. Muslim).

Penerapan teknologi ini dalam konteks industri tidak hanya mempermudah urusan manusia, tetapi juga mencerminkan nilai-nilai kemurahan hati dan kebaikan dalam Islam. Dengan solusi yang inovatif dan bernilai spiritual, penelitian ini bertujuan untuk menghadirkan sistem yang tidak hanya efisien dan berkualitas tetapi juga membawa manfaat sosial yang lebih luas, membantu perusahaan manufaktur untuk tetap kompetitif di era modern.

1.2 Pernyataan Masalah

Berdasarkan latar belakang dan kesimpulan yang telah disusun, rumusan masalah dalam penelitian ini dapat dirumuskan sebagai berikut:

1. Bagaimana jumlah keluaran cairan pelapis pada mesin pelapis kertas dapat dioptimalkan agar konsisten dan sesuai standar produksi, dengan tetap menjaga efisiensi dan kualitas produk?
2. Bagaimana penerapan otomasi berbasis algoritma PID yang dioptimasi menggunakan PSO dapat meningkatkan performa sistem kontrol, seperti efisiensi, akurasi, dan kestabilan parameter kontrol?

1.3 Tujuan Penelitian

Penelitian ini bertujuan untuk mengoptimalkan sistem kontrol aliran cairan pelapis pada mesin pelapis kertas menggunakan algoritma PID dengan *tuning* PSO, serta mempertimbangkan efisiensi dan konsistensi pengendalian aliran cairan dalam proses produksi.

1.4 Batasan Masalah

Adapun batasan masalah dalam penelitian ini dapat dijabarkan dengan poin-poin di bawah ini.

1. Penelitian ini hanya membahas optimasi sistem kontrol aliran cairan pada mesin pelapis kertas menggunakan PID dengan *tuning* PSO. Penelitian ini tidak membahas komposisi cairan yang digunakan dalam proses pelapisan atau jenis bahan baku lainnya yang terkait dengan proses produksi.
2. Penelitian ini tidak mencakup modifikasi atau analisis terhadap rangkaian mesin atau komponen fisik lain pada mesin pelapis kertas.

Sistem dan rangkaian yang digunakan sudah disediakan oleh pihak perusahaan dan tidak dapat diubah dalam konteks penelitian ini.

3. Penelitian ini hanya mengevaluasi kinerja sistem kontrol dalam mengatur aliran cairan sesuai dengan parameter yang telah ditentukan, seperti konsistensi aliran dan kestabilan *setpoint*. Tidak membahas faktor lain, seperti pengaruh sistem lain dalam mesin atau aspek kualitas kertas.

1.5 Manfaat Penelitian

Penelitian ini memiliki beberapa manfaat yang dapat diidentifikasi, yakni sebagai berikut:

1. Perusahaan mendapatkan efisiensi terhadap biaya dari segi operasional, terutama jika perusahaan membutuhkan lebih banyak mesin pelapis kertass, meliputi efisiensi cairan, human error, dan kualitas yang diberikan.
2. Mempermudah tugas pekerja dalam menangani operasional mesin pelapis kertas yang sebelumnya masih membutuhkan berbagai tahapan yang tidak diperlukan, dan akhirnya meningkatkan kemungkinan pekerja dalam mengejar target perusahaan dengan lebih baik.
3. Penelitian ini juga dapat memberikan beberapa petunjuk kepada peneliti lain apabila menangani permasalahan atau desain sistem yang kurang lebih sama meskipun dengan mesin yang berbeda.

BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Pembahasan mengenai metode *tuning* menggunakan *Particle Swarm Optimization* (PSO) pada pengontrol PID telah dilakukan di berbagai jurnal penelitian, salah satunya adalah jurnal yang berjudul "Desain Optimasi PID *Controller* Pada Heating Furnace Temperature Menggunakan Metode *Particle Swarm Optimization* (PSO)" oleh (Muhammad Agil Haikal, Dandy Tulus Herlambang, Machrus Ali and Muhlasin, 2021). Penelitian tersebut telah memberikan salah satu gambaran penting mengenai perbandingan dari sistem kontrol tungku pemanas menggunakan beberapa skenario pengujian. Pengujian dilakukan menggunakan pengontrol PID dengan auto *tuning* matlab 2013a, optimasi *tuning* dengan PSO, *tuning* dengan kontrol loop tunggal konvensional, maupun tanpa menggunakan metode PID pada *setpoint* yang tetap maupun dinamis. Pada penelitian tersebut, pengontrol *PID* ditempatkan di dalam pengatur utama yang bertanggung jawab atas pengaturan suhu secara keseluruhan pada tungku pemanas. Hal ini dilakukan agar dapat mencapai suhu yang diinginkan dan mempertahankannya dengan akurat. Peningkatan maupun penurunan suhu dilakukan menggunakan panas dari pembakaran pada bahan bakar. Kuantitas bahan bakar ditentukan oleh bukaan katup sumber bahan bakar dan pengatur setelahnya berdasarkan nilai yang dihasilkan oleh *controller* sebelumnya. Berdasarkan hasil simulasi pengujian yang telah dilakukan menggunakan *setpoint* tetap maupun berubah, *tuning* menggunakan PSO mendapatkan hasil terbaik dengan *overshot*

sebesar 0,0721, *undershot* sebesar 0,0081, dan *settling time* pada 30,4283 detik. Meskipun pada metode *auto tuning* matlab 2013a menghasilkan nilai *overshot* yang sedikit lebih kecil daripada metode PSO, akan tetapi nilai dari *settling time* dan *undershot* metode tersebut masih cukup besar. Dengan hasil tersebut, PSO dapat menjadi salah satu pilihan terbaik dalam menentukan *tuning* pada metode PID, apabila pemanfaatannya dilakukan pada desain sistem yang kurang lebih sama.

Terkait *setpoint* dinamis menggunakan metode *tuning* yang sama juga pernah dibahas pada jurnal berjudul "Optimization of PID *Controller* Parameters on Flow Rate Control System Using Multiple Effect Evaporator *Particle Swarm Optimization*" oleh (Bambang Dwi. A., et al., 2015). PSO mampu memberikan perbaikan *tuning* sistem PID. *Tuning* dengan metode tersebut dapat memperbaiki respons sistem kendali pada sistem kendali *Multiple Effect Evaporator* (MEE) dengan *rise time* sebesar 0,01 detik, *overshoot* sebesar 3,35%, *settling time* sebesar 6,10 detik dan mampu memberikan respons *undershoot* sebesar 28,11%.

Metode *tuning* menggunakan PSO juga pernah dimanfaatkan oleh (Norhaida Mustafa, and Fazida Hanim, H., 2020) di dalam penelitiannya yang berjudul "Design of a Predictive PID *Controller* Using *Particle Swarm Optimization*". Mereka menyampaikan bahwa metode *tuning* menggunakan PSO adalah salah satu metode optimasi pengontrol PID modern yang banyak digunakan pada kebutuhan industri. Ada berbagai alasan mereka memilih PSO sebagai metode *tuning* pada penelitian tersebut, salah satu alasan utamanya adalah memiliki hasil kinerja optimal yang telah dibuktikan pada berbagai penelitian, diantaranya adalah pemanfaatan PID yang disetel PSO untuk mengontrol posisi kamera pada

Unmanned Aerial Vehicle (UAV), mengontrol *twin rotor* pada *Multi-Input Multi-Output* (MIMO), dll. Pernyataan ini juga sesuai dengan hasil dari simulasi yang mereka lakukan di dalam penelitian ini. Parameter PSO yang dikaji pada penelitian ini adalah bobot inersia dan jumlah iterasi. Simulasi MATLAB digunakan untuk mengevaluasi performa motor dengan PID-PSO *Controller*, PID *gain* 1, dan tanpa PID. Hasil simulasi menunjukkan bahwa kombinasi bobot inersia 0.9 dan 100 iterasi dengan 54 partikel menghasilkan performa terbaik untuk kontroler PID-PSO. Perbandingan performa motor dengan kontroler PID-PSO, PID *gain* 1, dan tanpa PID menunjukkan bahwa PID-PSO menghasilkan *overshoot* yang lebih rendah, *riset time* yang minimal, dan *settle time* yang lebih singkat. Keseluruhan penelitian tersebut cukup menjelaskan bagaimana metode PSO bekerja pada pengontrol PID, termasuk juga performanya yang lebih baik daripada metode lain. Hanya saja, objek yang dipakai dalam penelitian ini hanya berupa pengendalian dc motor. Masih belum menggunakan beberapa parameter sekaligus dengan *output* aktuator seperti *motor servo*.

Studi yang sejenis telah dilakukan oleh Moch Febriawan Harianto dan Yuliyanto Agung Prabowo (2021) dalam jurnal berjudul "Sistem Kontrol Pemanas Air Kamar Mandi Menggunakan PID *Controller*". Mereka mengemukakan bahwa pengendalian pemanas air kamar mandi dengan suhu *setpoint* 35°C dapat diatur dengan baik menggunakan PID *Controller*. Pengaturan PID menggunakan metode *tuning trial-and-error* dengan nilai Kp = 10, Ki = 5, dan Kd = 2, yang menghasilkan respons *transien* dengan rise time 13 detik, settling time 16 detik, dan *steady-state error* 0,2%. Prototipe sistem pemanas air kamar mandi ini terdiri dari beberapa

komponen utama, termasuk sensor *DS18b20*, sensor *level* air, dan sensor *DHT11* sebagai *input*, serta *heater*, *solenoid door lock*, lampu DC, kipas, pompa air, dan LCD sebagai *output*. Sistem ini terbagi menjadi dua bagian, yaitu pintu otomatis dan pengisian tangki otomatis. Pada pintu otomatis, tombol tekan mengaktifkan *solenoid door lock* untuk membuka pintu, menyalakan lampu dan kipas, serta menampilkan suhu dan kelembaban ruangan di LCD. Proses penstabilan suhu pada tangki kamar mandi melibatkan tiga tangki, yaitu panas, dingin, dan utama. *Heater* memanaskan air dalam tangki panas hingga 45°C sebelum proses PID dimulai dengan *setpoint* antara 33–35°C. Pengisian tangki utama berlangsung secara terus-menerus sampai mencapai *setpoint*, dan pompa air secara otomatis mengurangi volume air saat tangki utama penuh.

Penelitian berjudul "Design of an Autonomous Smart Shower with Sensors and Actuators" yang diteliti oleh (Tareq Khan, 2018) memiliki desain sistem yang serupa. Jurnal tersebut membahas mengenai sistem penentuan suhu air yang optimal berdasarkan besaran suhu dan kecepatan air yang diinginkan oleh pengguna. Sensor *water flow* dan suhu digunakan di sini dengan hasil *output* digunakan untuk mengatur bukaan *motor servo.* Katup bola dengan *motor servo* diletakkan pada kedua media penghubung antara air panas maupun dingin dan diatur sedemikian rupa untuk memberikan kuantitas air yang sesuai dengan suhu yang diinginkan pengguna. Algoritma yang digunakan untuk mengatasi kalibrasi pada parameter suhu dan kecepatan air tersebut disesuaikan sedemikian rupa agar dapat menjadi nilai dalam menentukan lebar sudut bukaan dari tiap *motor servo*, yang mana tiap *servo motor* tersebut menjadi nilai dalam penentuan lebar bukaan

tiap katup bola. Sistem yang sudah diatur tersebut, diuji dalam dua tahap. Pada tahap pertama, rotasi *motor servo* diamati dan diverifikasi untuk suhu yang berbeda mulai dari 0 hingga 102˚F dengan laju aliran mulai dari 0 hingga 266 mL / detik. Berdasarkan pengujian tersebut, didapatkan hasil bahwa, frekuensi nilai keduanya dapat dikontrol secara eksternal dengan memutar knop potensiometer. Dan pada tahap kedua, pengujian pertama dilakukan kembali dengan mengubah nilai target secara *real-time*. Dengan *setpoint* awal suhu air adalah 90˚F dan kecepatan air sebesar 13 mL/sec, komposisi air dingin yang dihasilkan adalah 77˚F dan air panas di 104˚F. Semua nilai tersebut juga diuji dengan mengubah *setpoint* suhu air dan kecepatan air. Dan hasilnya sistem mampu menyesuaikan rasio untuk mendapatkan *setpoint* sesuai dengan yang diharapkan pada suhu optimal yang diinginkan oleh pengguna. Meskipun, ketika beberapa percobaan ketika melakukan kalibrasi kecepatan aliran dan suhu air melalui sensor ditemukan bahwa setelah roda sensor berhenti, dibutuhkan jumlah debit aliran tertentu agar dapat memulai roda berputar lagi karena inersia. Dengan cara yang sama, ketika roda berjalan dengan kecepatan tinggi, kecepatan roda tidak segera berkurang karena inersia roda, bahkan jika laju aliran berkurang. Fenomena ini dapat menyebabkan penundaan dalam menghitung laju aliran yang akurat oleh sistem yang diusulkan. Kekurangan tersebut menjadi kendala yang masih belum teratasi pada keseluruhan penelitian ini.

Perihal pemanfaatan *Infrared* (IR) sensor sebagai alat penghitung kecepatan putaran suatu objek, telah diteliti oleh (Moch Umar Hidayat dan Usman Abdul Hakim, 2021) pada jurnal berjudul “Purwarupa Alat Untuk Memantau Kecepatan Putaran Pada Mesin Berputar Berbasis IoT”. Pada penelitian tersebut,

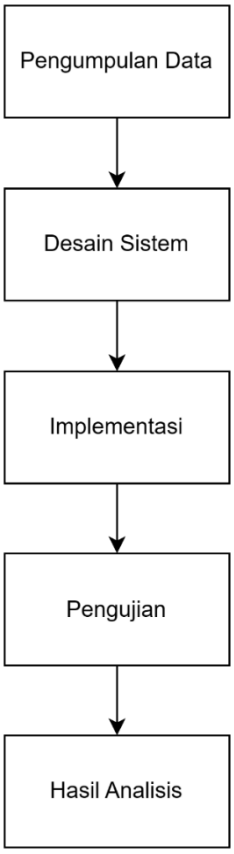
sensor *IR Obstacle* digunakan untuk memantau atau memonitoring mesin berputar, seperti turbin, rotor, kompresor, sentrifugal, dll. Sensor *IR Obstacle* dipakai untuk nilai masukkan, Arduino Uno dan NodeMCU ESP8266 sebagai prosesornya, dan hasil nya berupa basis data yang berisi nilai rpm kecepatan putaran generator yang tertampil pada LCD, aplikasi *Blynk*, dan *mobile apps* melalui *ThingSpeak*. Saat kondisi *overspeed* notifikasi akan terkirim via email. Pengujian dilakukan pada miniatur generator menggunakan sebuah dinamo dc yang sudah dibuat dimmer dengan 3 sesi, yakni *off speed*, *on speed*, dan *over speed*. Dengan kecepatan konstan, respons yang didapatkan sudah sesuai pada tiap media keluaran yang diinginkan. Dari posisi mati, hingga melebihi batas kecepatan yang telah ditentukan. Mulai dari keluaran dari nilai kecepatannya di LCD, *Blynk*, hingga *ThingSpeak*.

BAB III

DESAIN DAN IMPLEMENTASI

3.1 Desain Penelitian

Desain penelitian adalah kerangka kerja atau rencana yang digunakan untuk mengatur, menjalankan, dan mengarahkan sebuah penelitian yang mencangkup metode, prosedur, dan strategi yang digunakan peneliti untuk menjawab pertanyaan penelitian atau menguji hipotesis. Adapun jenis penelitian yang digunakan pada penelitian ini adalah penelitian kuantitatif, yakni analisis data yang digunakan dapat diukur. Alur penelitian dapat dilihat pada Gambar 3.1 di bawah ini.



Gambar 3.1 Alur Penelitian

Penelitian dimulai dengan tahap pengumpulan data, yakni bagaimana rangkaian pengumpulan data yang telah dirancang dapat mengumpulkan informasi sesuai dengan kebutuhan sistem yang akan didesain. Tahap ini juga mencakup langkah-langkah untuk memastikan data diperoleh secara akurat melalui proses yang tepat yang kemudian divisualisasikan dalam bentuk tabel.

Setelah data terkumpul, data tersebut dianalisis terlebih dahulu sebelum digunakan pada desain sistem. Analisis ini menggunakan persamaan linier sederhana untuk menentukan hubungan antara kecepatan air dan waktu dalam penetapan *setpoint*. Hasil analisis ini disajikan dalam bentuk tabel *setpoint* yang telah dianalisis, serta menghasilkan persamaan yang akan digunakan oleh sistem untuk menentukan *setpoint* secara *realtime* berdasarkan parameter aktual.

Setelah data dikumpulkan dan telah didapatkan persamaan untuk menentukan *setpoint* sistem didesain sedemikian rupa untuk menangani permasalahan yang ada. Dimulai dengan pembuatan rangkaian yang melibatkan sensor, aktuator, dan unit kontrol. Desain ini bertujuan mengoptimalkan keluaran cairan pelapis dengan mempertimbangkan kebutuhan spesifik seperti akurasi, sensitivitas, dan daya tahan komponen. Kalibrasi sensor dilakukan untuk memastikan data akurat, dan rangkaian diuji melalui simulasi serta pengujian nyata. Hasilnya adalah rangkaian yang efisien dan terintegrasi, mendukung pengumpulan data dan pengujian sistem dengan baik.

Tahap berikutnya mengimplementasikan *tuning* PSO untuk menyempurnakan parameter PID. PSO diimplementasikan melalui sejumlah proses pada program pengaplikasian yang menggambarkan proses tuning dari inisialisasi

hingga solusi optimal. Parameter seperti jumlah partikel dan iterasi disesuaikan untuk memastikan stabilitas dan responsivitas sistem. Kinerja PID dievaluasi terhadap *setpoint* yang diperoleh sebelumnya, menghasilkan sistem kontrol yang optimal. Grafik dan tabel hasil evaluasi menunjukkan peningkatan efisiensi, memastikan sistem mampu menghadapi berbagai kondisi *setpoint* dengan desain yang stabil.

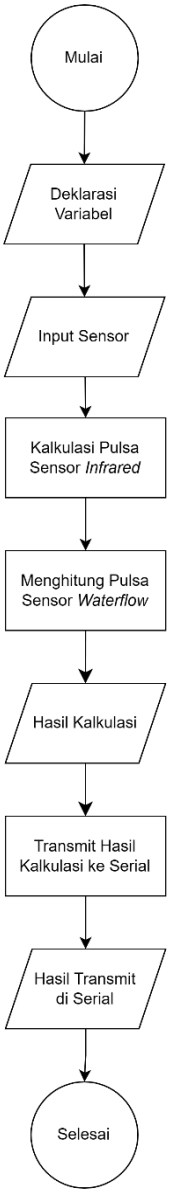
Pada tahap akhir, sistem PID yang telah dioptimasi menggunakan PSO akan diuji pada desain sistem. Pengujian dilakukan melalui simulasi menggunakan prototipe dan evaluasi langsung oleh pihak perusahaan dalam kondisi nyata pada mesin pelapis kertas. Tujuan pengujian ini adalah memastikan sistem yang dibuat mampu menyelesaikan permasalahan yang ada secara efektif.

Hasil dari pengujian ini akan dirangkum dalam analisis akhir untuk mengevaluasi kinerja sistem secara ringkas. Analisis tersebut mencakup kesesuaian sistem dengan kebutuhan operasional serta efektivitas solusi dalam menangani permasalahan yang diidentifikasi sebelumnya. Dengan pendekatan ini, kesimpulan akhir dapat memberikan gambaran menyeluruh mengenai keberhasilan implementasi desain sistem.

3.2 Pengumpulan Data

Tahap pengumpulan data bertujuan untuk menentukan korelasi antara kecepatan air dengan waktu pada kecepatan roda tetap yang digunakan untuk mendapatkan nilai *setpoint*. Kecepatan air diukur dalam mililiter per detik (mL/s), dan kecepatan putaran roda dalam rotasi per menit (RPM). Kedua data ini didapatkan di tiap detik (s) dan digunakan untuk mengetahui pengaruh dari

kecepatan aliran cairan yang digunakan sebagai *setpoint*. Alur proses dari pengumpulan data oleh *atmega2560* dapat dilihat pada Gambar 3.2.



Gambar 3.2 Alur proses rangkaian pengumpulan data

Proses diawali dengan deklarasi variable yang berisi nilai yang dibutuhkan untuk memproses pulsa dari tiap sensor, maupun persamaan untuk melakukan kalkulasi pada *flow rate* maupun RPM. Setelah itu, *Arduino* menghitung pulsa dari

sensor *waterflow* maupun *infrared* yang masuk. Hasil dari kalkulasi dikirimkan ke *Serial* untuk diproses oleh *ESP8266* yang kemudian dikirimkan ke *Blynk.* Proses ini akan berulang hingga roda tidak bergerak dalam jangka waktu tertentu atau *Arduino* dimatikan, diberhentikan, ataupun *ESP8266* terjadi eror seperti tidak terhubung lagi melalui *WiFi* atau *Blynk.*

3.2.1 Hasil Pengumpulan Data

Tiap sensor yang telah dipasang digunakan untuk mengotomasikan proses pengumpulan data. Sensor *infrared* mengukur kecepatan putaran roda, sedangkan sensor *water flow* mengukur kecepatan air. Data tersebut diproses oleh *Arduino* dan dikirim ke *Blynk* melalui modul *ESP8266,* kemudian dikumpulkan tiap iterasi (saat rol kertas selesai). Pengumpulan data dilakukan dengan kecepatan roda tetap di 50 RPM.

3.2.2 Analisis Data

Setelah data terkumpul, analisis terhadap korelasi antara data perlu dilakukan terlebih dahulu. Proses analisis dilakukan melalui pembuatan persamaan linier untuk menghitung nilai kemiringan (m) dan intersep (b). Persamaan linear secara umum dapat ditulis pada persamaan 3.1.

$$y = mx + b \tag{3.1}$$

Peneliti mengidentifikasi dua variabel yang berkaitan, yakni kecepatan air (y) dan kecepatan putaran roda (x). Dengan mengumpulkan data empiris dari pengukuran, peneliti menghitung nilai kemiringan (m) yang menunjukkan

bagaimana perubahan pada satu variabel (x) mempengaruhi variabel lainnya (y). Kemiringan ini dapat dihitung dengan menggunakan persamaan 3.2 di bawah ini.

$$m = \frac{\Delta y}{\Delta x} \tag{3.2}$$

Setelah nilai kemiringan diperoleh, peneliti melanjutkan dengan menghitung nilai intersep (b) pada persamaan linier. Intersep ini mewakili nilai y ketika x sama dengan nol, memberikan gambaran tentang kecepatan aliran dasar pada kondisi tanpa pengaruh dari waktu dengan kecepatan tetap. Nilai intersep dapat ditentukan dengan menggunakan salah satu pasangan data yang ada dalam *dataset*. Dengan kedua nilai ini, peneliti dapat membangun persamaan linier yang menjelaskan hubungan antara kecepatan air dan waktu.

Seluruh data dikumpulkan dan diproses secara berulang hingga menyelesaikan jumlah rol yang ditentukan dalam proses pengumpulan data secara empiris. Data tersebut dikumpulkan dan disajikan dalam bentuk tabel yang dapat dilihat pada Tabel 3.1.

Tabel 3.1 Tabel skenario pengumpulan data

Menit	**Putaran roda (RPM)**	**Putaran roda aktual (RPM)**	**Kecepatan aliran aktual (mL/s)**
1	50,0	…	…
…	50,0	…	…
n	50,0	…	…

3.3 Desain Sistem

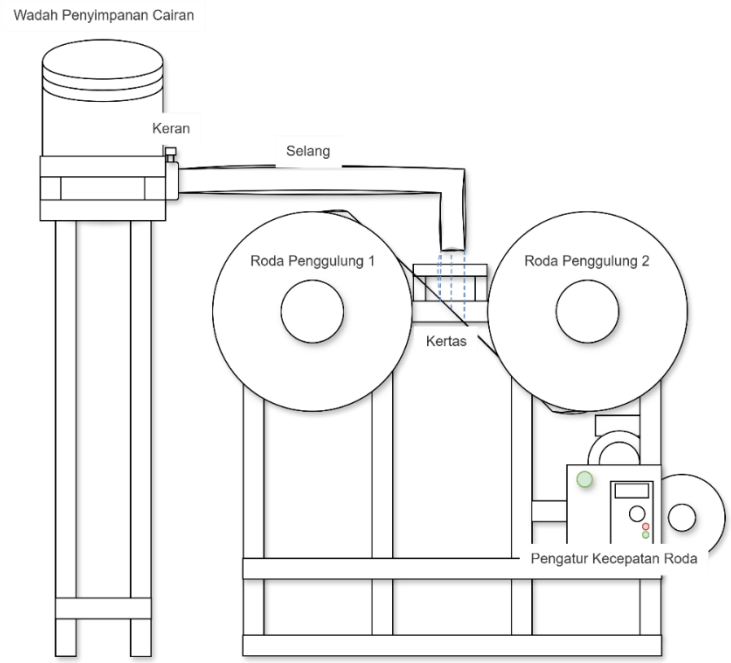
Desain sistem dibuat untuk menentukan rangkaian seperti apa yang akan digunakan pada tahap pengumpulan data maupun pengujian sistem serta penjelasan terkait PID dan *tuning* PSO seperti apa yang akan diimplementasikan.

Rangkaian awal yang saat ini digunakan oleh pihak perusahaan akan dijabarkan secara mendetail terlebih dahulu agar bisa menentukan langkah selanjutnya, yakni penentuan kebutuhan dari desain rangkaian. Setelah kebutuhan dijabarkan, komponen perlu dikalibrasi agar dapat dievaluasi apakah sesuai dengan kebutuhan atau tidak. Jika tidak sesuai, kebutuhan dari sistem perlu ditinjau ulang dan dilakukan proses kalibrasi dan evaluasi ulang. Setelah itu, desain akhir dari rangkaian akan ditentukan berdasarkan kebutuhan yang telah dievaluasi.

Setelah tahapan rangkaian fisik dari desain sistem telah dibuat, langkah berikutnya adalah memahami lebih lanjut model dari otak sistem yang akan diproses oleh komponen proses. Metode yang digunakan dalam desain sistem ini melibatkan penggabungan antara PID dan metode *tuning* PSO.

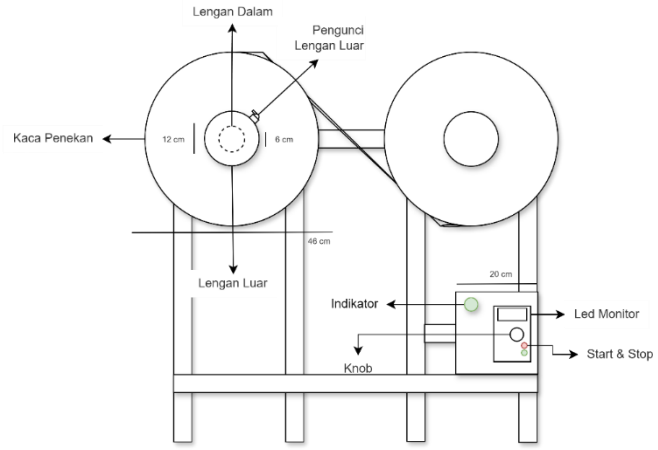
3.3.1 Desain Awal Mesin Pelapis Kertas

Desain pengaliran cairan terdiri dari 2 bagian utama, yakni roda penggulung kertas dan pengaliran cairan dari wadah ke kertas. Mesin roda penggulung kertas berfungsi untuk menggerakkan dua roda agar rol kertas dapat digulung dari kertas mentah ke kertas yang telah dilapisi dengan kecepatan roda yang diatur manual. Hanya saja, konfigurasi kecepatan tidak bisa diintegrasikan dengan sistem dari luar. Hal ini menyebabkan sistem dari luar harus mendapatkan nilai dari kecepatan rodanya sendiri. Desain awal dari mesin ini dapat dilihat pada Gambar 3.3.



Gambar 3.3 Desain awal mesin pelapis kertas

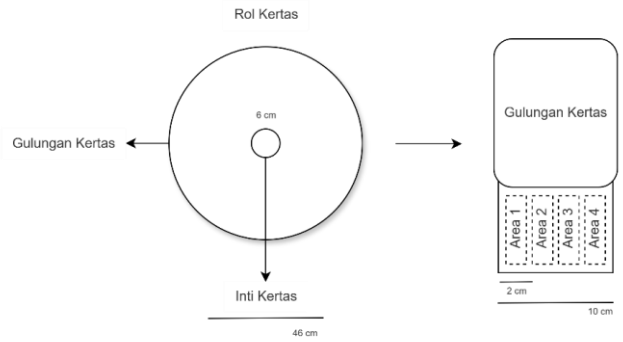
Penggulung kertas memiliki kaca penekan pada kedua sisi rol kertas yang menahan gulungan kertas agar tidak keluar dari batasnya, serta lengan bagian dalam dan luar. Lengan bagian dalam berdiameter 6 cm berfungsi untuk menahan rol kertas, sedangkan lengan bagian luar, dengan diameter 12 cm dan pengunci kuat, menjaga kaca penekan tetap pas dengan lebar rol yang memiliki diameter total 46 cm. Sementara itu, mesin pengatur kecepatan berbentuk kubus dengan panjang sisi sekitar 20 cm dilengkapi dengan knob yang digunakan untuk mengatur kecepatan putaran roda, tombol *start* dan *stop* untuk menjalankan atau menghentikan putaran roda, dan disertai indikator status penggulung kertas. Pengatur kecepatan tersebut dilengkapi dengan monitor *LED* kecil berukuran sekitar 4 inci untuk menampilkan kecepatan mesin saat ini. Gambar 3.4 memberikan pemahaman terkait detail mesin penggulung kertas.



Gambar 3.4 Detail roda dan pengatur kecepatannya

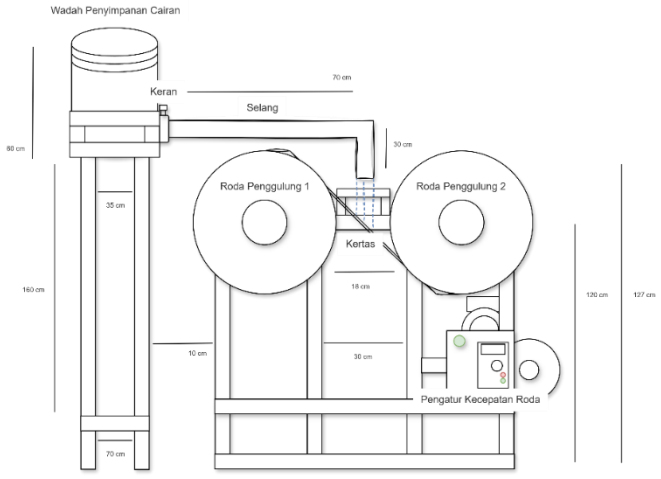
Selain membahas detail dari mesin pelapis kertas, pemahaman mendetail terhadap rol kertas perlu diperjelas. Setiap rol terdiri dari dua bagian utama, yaitu gulungan kertas dan inti gulungan. Gulungan kertas adalah bagian kertas yang digulung dan dilapisi, sedangkan inti gulungan merupakan lubang pada bagian tengah gulungan yang digunakan untuk memasukkan lengan penggulung kertas dan terbuat dari karton tebal.

Setiap rol kertas memiliki diameter total yang mirip dengan kaca penekan pada mesin pelapis kertas, yakni sekitar 46 cm, dengan 20 cm sebagai jari-jari dari gulungan kertas dan 3 cm sebagai jari-jari dari inti gulungan. Kertas pada gulungannya sendiri memiliki lebar total 10 cm yang dibagi menjadi 4 area pelapisan. Area tersebut dialiri oleh cairan pelapis kertas melalui 4 selang dan keran berbeda. Masing-masing areanya dibagi menjadi kurang lebih 2 cm. Gambar 3.5 di bawah ini menjadi gambaran penuh terkait detail dari rol kertas beserta kertas yang digulung.



Gambar 3.5 Detail rol kertas

Untuk mengatasi pengaliran cairan dari wadah ke kertas, selang akan dihubungkan dari wadah penyimpanan cairan berdiameter 30 cm ke antara dua roda penggulung. Jumlah cairan yang keluar diatur melalui motor *servo* berdasarkan kecepatan air dan putaran roda penggulung. Diantara kedua roda yang sedang berputar, terdapat jarak sebesar 18 cm yang dapat digunakan untuk mengalirkan cairan khusus ke lapisan kertas. Kertas tersebut bergerak dari roda penggulung 1 ke roda penggulung 2. Dari yang belum dilapisi cairan ke rol kertas yang sudah dilapisi cairan. Detail dari pembahasan pengaliran cairan dari wadah ke kertas dapat dilihat melalui Gambar 3.6 berikut.



Gambar 3.6 Detail pengaliran cairan

3.3.2 Menentukan Kebutuhan

Menentukan kebutuhan dari rangkaian akhir yang akan dibuat dilakukan terlebih dahulu untuk menghindari fluktuasi biaya karena salah dalam menentukan komponen yang diperlukan. Selain itu, rangkaian akhir yang akan dibuat akan tergambar lebih baik pada tahap penyelesaian akhir.

Berdasarkan desain rangkaian awal dan permasalahan yang dihadapi dalam penelitian ini, diketahui bahwa terdapat beberapa komponen penting yang diperlukan, yaitu komponen untuk memproses keseluruhan kebutuhan, mengukur kecepatan air di dalam selang dan kecepatan putaran dari roda, serta komponen yang dapat mengontrol keran air agar menyesuaikan secara *realtime* dan dikirim ke *mobile app*. Berdasarkan hal itu, rangkaian ini tak terlepas dari *hardware* maupun *software* yang berhubungan dengan perangkat elektronik maupun IoT.

3.3.2.1 Sensor *Water Flow*

Sensor *water flow* adalah perangkat yang mampu mendeteksi pergerakan gas atau cairan. Ada berbagai jenis sensor *flow*, baik yang berbasis elektronik dengan menggunakan teknologi ultrasonik, maupun yang bersifat mekanis (Kalsoom, T., 2020).

Pada sistem ini, *water flow* digunakan untuk mengukur kecepatan air yang mengalir melalui selang. Sensor ini bekerja dengan prinsip *hall effect*, di mana adanya aliran air akan menyebabkan perubahan medan magnet di dalam sensor yang kemudian diubah menjadi sinyal listrik. Keunggulan dari *water flow* ini adalah

kemampuannya untuk mendeteksi aliran air secara presisi dan respons cepat dengan konsumsi energi yang rendah (Kalsoom, T., 2020).

3.3.2.2 Sensor *Infrared*

Sensor *Infrared* (IR) merupakan sensor yang digunakan untuk mendeteksi atau mengukur cahaya inframerah yang tidak terlihat oleh mata manusia. Sensor ini bekerja dengan mengukur intensitas cahaya inframerah yang dipancarkan atau dipantulkan oleh objek di sekitarnya. Sensor ini biasanya digunakan dalam berbagai aplikasi seperti deteksi gerakan, pengukuran suhu tubuh, dan kontrol jarak jauh.

Dalam konteks sistem ini, *IR Obstacle Sensor* dapat digunakan untuk mendeteksi gerakan atau objek yang masuk ke dalam area kerja sistem (Moch Umar, H., & Usman Abdul, H., 2021). Data yang diperoleh dari sensor ini dapat diproses untuk mengetahui kecepatan putaran roda penggulung melalui rumus tertentu pada tahap kalibrasi nantinya. Sensor ini nantinya memanfaatkan penanda berwarna putih agar sensor dapat mendeteksi putaran dari roda penggulung. Penentuan penanda ini akan dibahas pada poin-poin berikutnya.

3.3.2.3 *Blynk*

Blynk adalah platform pengembangan aplikasi berbasis IoT yang memungkinkan pengguna untuk membuat, mengontrol, dan memantau perangkat IoT melalui *mobile apps* secara *real time* dengan mudah dan biaya yang relatif rendah (Hasan, D., & Ismaeel, A., 2020). Dengan *Blynk*, pengguna dapat membuat tampilan yang dapat disesuaikan untuk perangkat IoT mereka sendiri tanpa perlu memiliki pengetahuan pemrograman yang mendalam. Platform ini menawarkan

berbagai *widget* yang dapat disesuaikan, seperti tombol, gulir, grafik, dan tampilan data lainnya, yang dapat dengan mudah diintegrasikan ke dalam aplikasi *Blynk* yang dibuat oleh pengguna. Selain itu, *Blynk* juga menyediakan layanan *cloud* yang memungkinkan pengguna untuk menyimpan dan mengakses data perangkat mereka dari mana saja, serta mengirimkan pemberitahuan atau peringatan berbasis kondisi.

3.3.2.4 *Arduino Mega 2560*

Arduino Mega 2560 merupakan salah satu jenis mikrokontroler yang memiliki kemampuan yang luas dan fleksibel dalam mengendalikan berbagai perangkat elektronik. Mikrokontroler ini memiliki banyak pin *input* dan *output* digital serta analog, sehingga cocok digunakan untuk proyek yang membutuhkan banyak interaksi dengan lingkungan luar. Terdiri dari 54 pin digital *I/O*, 16 *input* analog, 4 *UART*, koneksi USB, header ICSP, tombol reset dan ruang sketsa yang lebih besar. Selain itu, *Arduino Mega 2560* juga dilengkapi dengan memori yang besar, sehingga dapat menangani proyek-proyek yang memerlukan penyimpanan data yang besar atau kompleks (Siswanto, M., et al,. 2019).

Dalam sistem ini, *Arduino Mega 2560* berperan sebagai otak dari sistem, yang mengatur semua proses yang terjadi. *Arduino* ini sudah termasuk dengan modul *ESP2866* yang digunakan untuk terhubung dengan jaringan internet. Selain itu, *Arduino Mega 2560* juga membutuhkan akses ke semua sensor dan aktuator yang digunakan dalam sistem, seperti sensor kecepatan air, sensor putaran roda penggulung, dan aktuator *motor servo*. Dengan demikian, *Arduino Mega 2560* menjadi komponen kunci dalam menjalankan sistem ini dengan efisien dan akurat.

3.3.2.5 Modul *ESP8266*

Modul ESP8266 (*Espressif Systems Protocol*) merupakan modul Wi-Fi yang serbaguna dan hemat daya yang memungkinkan mikrokontroler seperti *Arduino Mega 2560* untuk terhubung ke jaringan Wi-Fi dan berkomunikasi secara nirkabel (Tahir M., et. al., 2022). Modul ini berfungsi sebagai klien atau akses titik *Wi-Fi,* sehingga dapat digunakan untuk mengirim dan menerima data dari internet bahkan secara *real time* dengan baik serta latensi yang rendah sesuai dengan kebutuhan desain sistem transmisi data (Moon M. A. M., 2023). Dalam konteks sistem ini, modul *ESP8266* sudah menjadi satu komponen dengan *Arduino* Mega yang telah disebutkan. Tujuan dari komponen ini adalah menghubungkan sistem dengan *Dashboard Blynk* serta mengirimkan data sensor secara *real time*.

3.3.2.6 Aktuator Motor *Servo*

Motor *servo* adalah satu jenis aktuator yang digunakan untuk menggerakkan atau mengontrol suatu mekanisme dengan presisi. Motor *servo* bekerja dengan menggunakan sinyal kontrol yang dikirimkan ke dalamnya untuk mengatur posisi sudut rotor. Motor *servo* juga biasanya dilengkapi dengan fitur-fitur tambahan seperti *feedback* posisi, sehingga posisi akhir mekanisme dapat dikontrol dengan sangat tepat (Kusumadiarti, R. S., et. al. (2021).

Dalam konteks sistem ini, aktuator *motor servo* digunakan untuk mengontrol bukaan atau penutupan katup pada keran. Data yang diterima dari *Arduino Mega 2560*, yakni *setpoint* kecepatan air, digunakan untuk mengatur sudut putaran *motor servo* sehingga dapat mengendalikan kecepatan air sesuai dengan kebutuhan sistem. Dengan adanya *motor servo,* sistem dapat mengatur kecepatan

air secara otomatis dan presisi, sehingga meningkatkan efisiensi dan konsistensi proses produksi.

3.3.2.7 *Arduino IDE*

Arduino IDE adalah lingkungan pengembangan yang memudahkan pengguna untuk memprogram dan mengunggah kode ke *Arduino*. Dilengkapi dengan editor kode yang dilengkapi dengan fitur *syntax highlighting* dan *auto-completion*, IDE ini juga menyediakan berbagai pustaka dan contoh kode sebagai referensi. Dengan dukungan untuk bahasa pemrograman *Arduino* yang berbasis *C/C++*, *Integrated Development Environment* (IDE) ini cocok untuk digunakan oleh pemula maupun pengembang berpengalaman (Daoud, A. M. I. N. E., (2021). Kemudahan penggunaan dan dukungan untuk berbagai jenis *Arduino*, mulai dari *Uno* hingga *Mega*, menjadikan *IDE* ini populer di kalangan pengembang. Selain gratis, kehadiran *Arduino IDE* memungkinkan eksplorasi kreativitas dan inovasi dalam proyek elektronika, dari yang sederhana hingga kompleks.

3.3.3 Kalibrasi dan Evaluasi Komponen

Tiap komponen yang telah ditentukan pada poin sebelumnya dikalibrasi dan dievaluasi terlebih dahulu untuk menentukan apakah komponen yang telah disiapkan sesuai dengan kebutuhan secara aktual atau tidak. Tahap ini juga dapat menjadi acuan untuk penentuan spesifikasi yang dibutuhkan pada tiap komponen.

Komponen *Arduino* dan *ESP8266* tidak perlu dikalibrasi maupun dievaluasi karena telah disesuaikan dengan kebutuhan perusahaan. Komponen yang perlu diproses pada tahap ini adalah sensor dan aktuator, yakni sensor *water flow*,

infrared, dan aktuator motor *servo*. Proses kalibrasi maupun evaluasi ketiga komponen tersebut dapat dijabarkan pada poin-poin berikutnya.

3.3.3.1 Sensor *Waterflow*

Dalam menentukan spesifikasi dari sensor *water flow*, mesin pelapis kertas saat ini perlu dilakukan kalibrasi sebanyak 3 kali percobaan dengan bukaan keran yang berbeda. Masing-masing bukaan kerannya adalah paling kecil, paling besar dan diantara keduanya. Kalibrasi tersebut dilakukan dengan menentukan berapa volume air yang diinginkan terlebih dahulu. Setelah itu, keran dibuka untuk mengalirkan cairan pelapis sampai memenuhi kriteria dari volume yang telah ditentukan. Dalam proses tersebut, waktu diukur dari awal keran dibuka hingga memenuhi target dari volume cairannya. Setelah 3 kali percobaan kalibrasi dilakukan, hitung masing-masing nilai dengan persamaan laju aliran air berikut.

$$\text{kecepatan air} = V \,/\, t \tag{3.3}$$

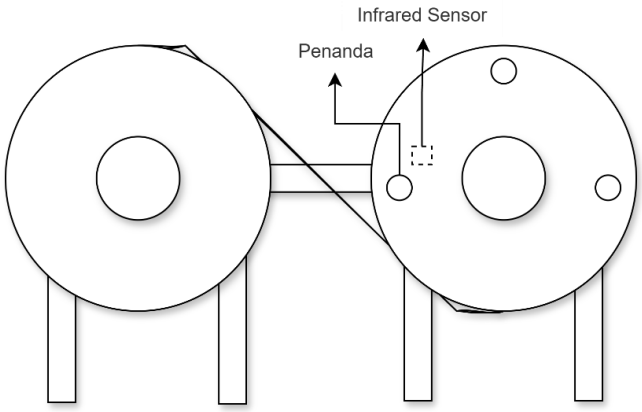
Hasil dari tiap percobaan akan dianalisis untuk menentukan sensor *water flow* mana yang sesuai. Kesesuaian dari sensor tersebut dinilai berdasarkan rentang nilai yang dapat mencangkup nilai terendah hingga tertinggi dari hasil kalibrasi.

Evaluasi komponen pada sensor ini dapat dilakukan dengan cara memasang sensor pada selang dan menghubungkannya dengan *Arduino*. *Arduino* digunakan untuk mendapatkan data dari sensor untuk mengetahui volume air yang didapatkan oleh sensor. Simulasi evaluasi sendiri mirip dengan saat melakukan kalibrasi, yakni melakukan percobaan sebanyak 3 kali dengan bukaan keran air

yang berbeda. Ketiga data tersebut akan dibandingkan dengan hasil yang didapatkan melalui sensor dengan nilai toleransi tertentu.

3.3.3.2 Sensor *Infrared*

Sensor perlu diubah dari *feedback* yang didapatkan terhadap hasil sensor dengan RPM dalam melakukan kalibrasi. Agar memiliki respons yang sesuai terhadap perubahan kecepatan roda, 1 penanda dipasang pada kaca penekan dari luar roda. Gambar 3.7 menunjukkan desain dari cara kerja sensor *infrared* pada roda.



Gambar 3.7 Cara kerja infrared pada roda

Cara kerja *infrared sensor* yang menjadi salah satu *obstacle sensor* adalah dengan mengetahui selisih waktu antara *pulse* atau sensor mendeteksi penanda sebelumnya dengan yang sekarang. Selisih tersebut dihitung dengan persamaan dasar RPM di persamaan 3.4 berikut ini.

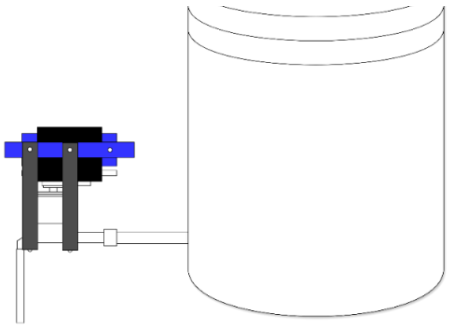
$$RPM = (PR1) \ * \ 60 \ / \ PPR * GR \tag{3.4}$$

Di mana RPM adalah putaran per menit, PR1 adalah jumlah pulsa yang diterima dalam satu detik, PPR adalah jumlah pulsa per putaran, dan GR adalah rasio roda gigi. Dengan persamaan tersebut, sensor *infrared* dapat memberikan nilai yang sesuai dengan keadaan kecepatan roda saat ini.

Untuk evaluasinya didasarkan pada selisih nilai dari ketentuan perusahaan, yakni 50 RPM. Dari tiap percobaan di tiap kecepatan, sensor akan dihitung dan dianalisis hasil dari selisihnya. Kemudian ditentukan toleransi akhirnya.

3.3.3.3 Aktuator Motor *Servo*

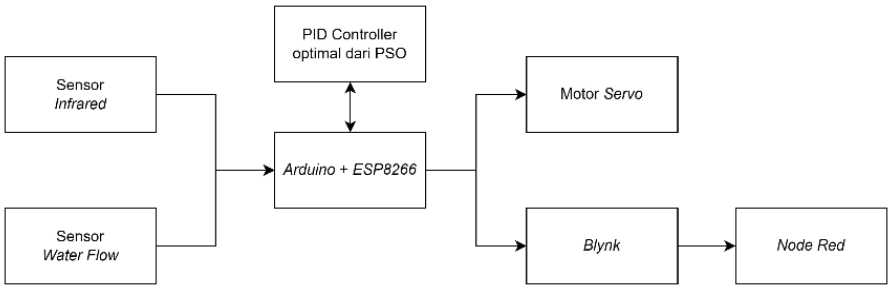
Pemilihan *servo* yang sesuai didasari oleh penelitian / percobaan dari referensi jurnal atau proyek. Pada komponen ini, evaluasi didasari langsung melalui percobaan secara langsung dengan memasangkan komponen dengan keran air. Baik dari segi kekuatan *servo* maupun kemudahan pemasangan. Pemasangannya sendiri dilakukan dengan memodifikasi keran dengan *servo* yang ada.



Gambar 3.8 Desain pemasangan *Servo* pada keran

3.3.4 Desain Akhir Rangkaian

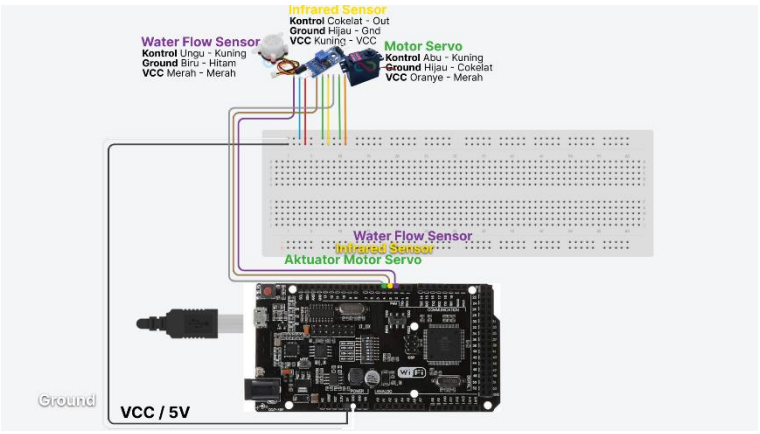
Desain akhir dari rangkaian ini terdiri beberapa bagian, yakni komponen *input*, proses dan *output*. Komponen-komponen tersebut memiliki tugas dan fungsinya masing-masing. Desain dari rangkaian ini dapat dilihat pada Gambar 3.9.



Gambar 3.9 Desain akhir rangkaian pada mesin pelapis kertas

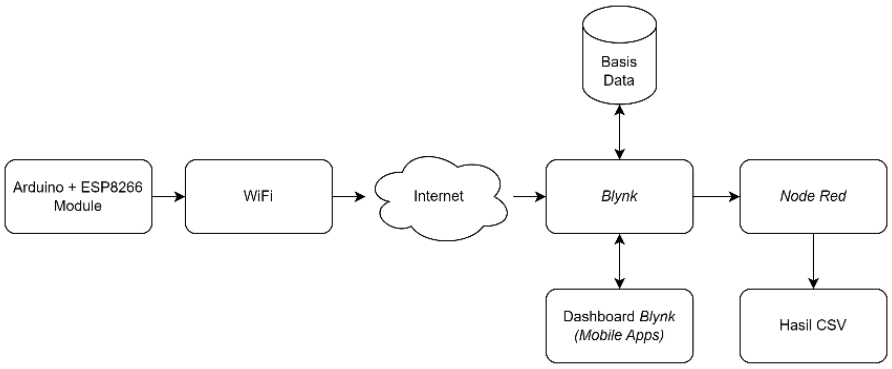
Komponen *input* terdiri dari sensor *infrared* dan *water flow*. Masing-masing memiliki fungsi untuk mendapatkan nilai dari kecepatan putaran roda dari mesin roda penggulung kertas dan kecepatan air yang mengalir dari wadah penyimpanan cairan melalui selang secara *realtime*.

Komponen proses terdiri dari *Arduino* yang digunakan untuk memproses data kedua sensor yang masuk. Data tersebut akan diproses menggunakan algoritma PID dengan *tuning* PSO. Selain itu, pada bagian komponen ini terdapat modul *esp8266* yang digunakan untuk menghubungkan *Arduino* dengan jaringan internet. Adapun desain rangkaian yang akan dipakai untuk menghubungkan komponen proses ini dengan komponen lainnya dapat dilihat pada Gambar 3.10 di bawah ini.



Gambar 3.10 Desain rangkaian *Arduino*

Transmisi dari komponen tersebut dilakukan ketika semua nilai dari parameter telah didapatkan pada tiap iterasi proses yang telah dilakukan, yakni setelah aktuator dijalankan. Data dikirimkan oleh *Arduino* ke *Blynk* melalui jaringan internet yang didapatkan dari modul *ESP2866.* Setelah itu *Node-Red* mengambil data sensor secara *realtime* melalui API dari *Blynk untuk menghasilkan* nilai CSV yang sesuai. Desain sistem dari transmisi data tersebut dapat dilihat melalui Gambar 3.11 berikut.



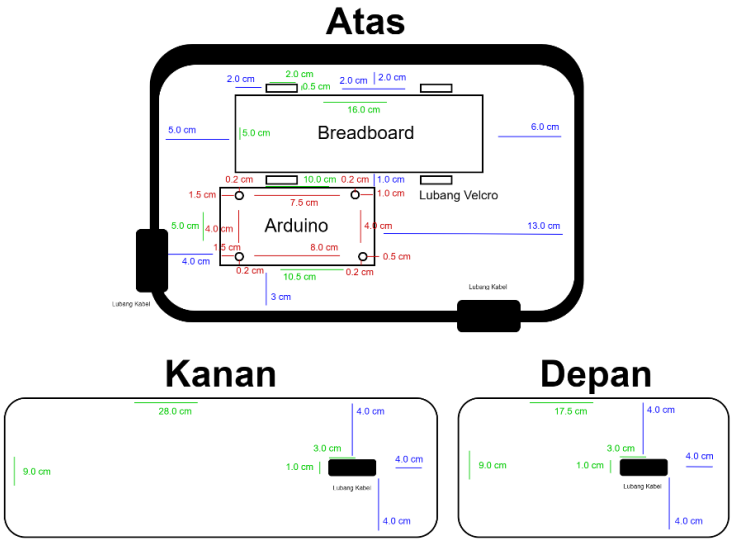
Gambar 3.11 Desain transmisi data

Arduino akan memproses data yang masuk, kemudian mengirimkannya ke *Blynk* melalui *WiFi* yang terhubung dengan jaringan internet secara *realtime*. *Blynk* akan menyimpan data tersebut ke basis data, kemudian menampilkan data-data tersebut apabila ada permintaan melalui *mobile apps.*

Komponen *output* terdiri dari motor *servo* yang digunakan untuk mengatur lebar bukaan keran dan *Blynk* yang digunakan untuk menampilkan data tiap iterasi dari sistem melalui *Blynk* berupa *mobile apps*.

Dalam pembagiannya, komponen-komponen ini terdiri dari komponen internal yang disimpan di *project box,* dan komponen eksternal yang diletakkan

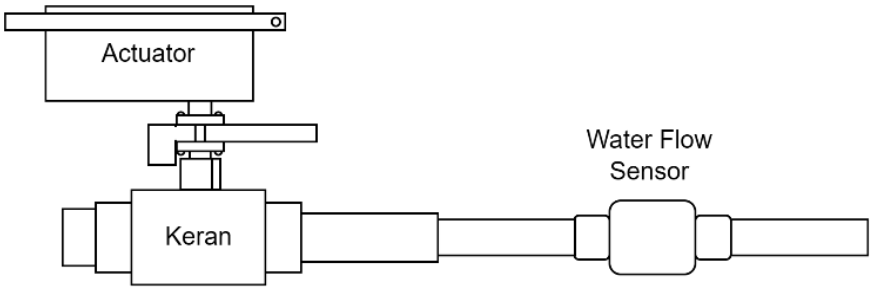
sesuai fungsinya. *Project box* adalah wadah berbentuk balok yang berfungsi sebagai tempat untuk menyimpan komponen utama seperti *Arduino, breadboard, power supply, dan ESP8266.* Aktuator dan sensor akan terhubung ke *Arduino* yang tersimpan di dalam *project box* sebagai komponen eksternal, menjaga agar rangkaian lebih rapi, terlindungi, dan terorganisir dengan baik. Adapun desain rangkaian dari *project box* ini dapat dilihat dilihat pada Gambar 3.12 berikut ini.



Gambar 3.12 Desain *project box*

Komponen eksternal perlu diletakkan sesuai dengan fungsinya masing-masing. Aktuator dipasang di atas keran, dengan roda penggeraknya sejajar dengan poros putaran keran. Saat aktuator berputar, ia mengendalikan bukaan keran melalui gagang keran berdasarkan sudut rotasi yang diberikan. Setelah keran terbuka, cairan mengalir keluar melalui lubang keluaran menuju pipa yang terhubung dengan sensor *water flow*. Sensor ini mengukur kecepatan aliran cairan sebelum dialirkan melalui selang menuju titik keluaran, di mana cairan akan

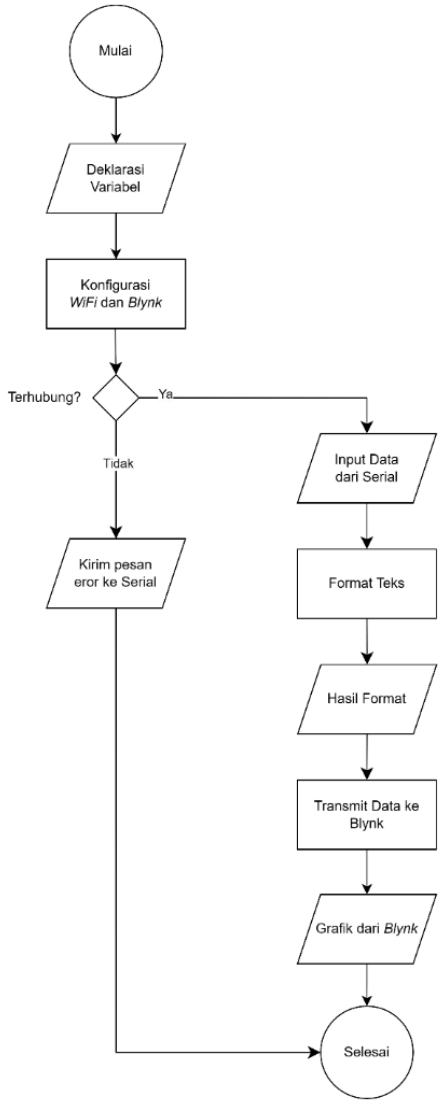
dialirkan ke kertas yang sedang digulung. Gambar 3.13 menggambarkan desain dari penjelasan ini, termasuk interaksi antara aktuator dan sensor *water flow*.



Gambar 3.13 Implementasi aktuator dan sensor *water flow*

Selain aktuator dan sensor *water flow*, *infrared sensor* juga menjadi salah satu komponen eksternal. Sensor ini diletakkan pada tiang di sebelah roda kertas yang sudah terlapisi. Dengan penanda berupa berupa tonjolan putih yang ditempelkan pada kaca penekan rol kertas.

ESP82666 bertugas untuk mengirimkan data dari *Atmega2560* ke *ESP8266* yang didapatkan melalui *Serial*. Langkah pertamanya adalah mengecek apakah sudah terhubung dengan *WiFi* maupun *Blynk*. Jika terhubung, maka data yang didapatkan melalui *Serial* dari *Atmega2560* dicek untuk mengetahui data mana saja yang perlu diterima dan dikirimkan ke *Blynk*. Proses ini akan berulang hingga *Arduino* dimatikan, diberhentikan, ataupun tidak terhubung lagi melalui *WiFi* atau *Blynk*. Untuk lebih jelasnya *flowchart* ditampilkan pada Gambar 3.14 di bawah ini.



Gambar 3.14 Alur proses transmisi data oleh *ESP8266*

3.3.5 Penerapan PID *Controller*

PID *Controller* adalah metode kontrol yang digunakan dalam sistem kontrol otomatis. Metode ini beroperasi dengan menggabungkan tiga komponen utama, yakni *Proportional* (Kp), *Integral* (Ki), dan *Derivative* (Kd). Masing-masing komponen ini memainkan peran penting dalam menghasilkan sinyal kontrol yang bertujuan untuk mengarahkan *output* sistem agar mencapai dan mempertahankan nilai target atau *setpoint* dengan presisi yang tinggi. Ketika

bekerja bersama, ketiga komponen ini memungkinkan PID untuk mengelola respons sistem secara efektif, mengurangi eror, dan meningkatkan stabilitas.

3.3.5.1 Komponen PID

Komponen *Proportional* (Kp) bertugas memberikan respons yang sebanding dengan besarnya eror, yaitu selisih antara *setpoint* dengan nilai aktual dari *output*. Dengan kata lain, semakin besar eror yang terjadi, semakin besar pula sinyal kontrol yang dihasilkan oleh komponen ini. Namun, jika hanya menggunakan komponen ini, sistem cenderung tidak dapat mencapai *setpoint* secara akurat dan sering kali menghasilkan *steady-state error*, yaitu perbedaan konstan antara *output* dan *setpoint* meskipun sistem telah mencapai stabilitas.

Komponen ini berperan dalam menghasilkan sinyal kontrol yang berbanding lurus dengan eror yang terdeteksi. Rumus dasar untuk komponen *Proportional* ini dapat dilihat pada persamaan 3.5 di bawah ini.

$$u(t) = k_p e(t) \tag{3.5}$$

Di mana:
- $u(t)$ adalah sinyal kontrol pada waktu t.
- k_p adalah konstanta proporsional.
- $e(t)$ adalah eror, yaitu perbedaan antara *setpoint* dan proses variabel.

Fungsi utama dari komponen *Proportional* adalah untuk mempercepat respons sistem terhadap perubahan *setpoint*, sehingga sistem dapat dengan cepat mendekati nilai yang diinginkan. Namun, tanpa kontribusi dari komponen lain, penggunaan komponen *Proportional* saja dapat menyebabkan *steady-state error*, yang berarti sistem tidak dapat mencapai *setpoint* dengan tepat. Oleh karena itu,

diperlukan penyesuaian lebih lanjut melalui komponen *Integral* dan *Derivative* untuk mengurangi atau menghilangkan eror yang terjadi.

Komponen *Integral* (I) memiliki peran penting dalam menghilangkan *steady-state error*, yaitu kesalahan yang bertahan dalam jangka panjang meskipun sistem telah mencapai kestabilan. Komponen ini bekerja dengan cara mengakumulasi eror seiring berjalannya waktu. Jika eror tetap ada, nilai integral akan terus bertambah, yang menyebabkan sinyal kontrol meningkat sampai eror tersebut benar-benar nol. Namun, komponen *Integral* harus diatur dengan hati-hati, karena jika tidak, dapat menyebabkan sistem mengalami osilasi, yaitu kondisi di mana sistem berfluktuasi atau berayun secara periodik di sekitar *setpoint* atau nilai keseimbangan. Rumus dasar untuk komponen integral dapat dilihat di persamaan 3.6 di bawah ini.

$$u(t) = k_i \int e(t)\, dt \tag{3.6}$$

Di mana:
- k_i adalah konstanta integral.
- Integrasi dari *error* $e(t)$ dilakukan dari waktu 0 hingga waktu t.

Fungsi utama dari komponen *Integral* adalah untuk memastikan bahwa eror yang terakumulasi dari waktu ke waktu akan diperbaiki, sehingga sistem dapat mencapai *setpoint* dengan tepat tanpa meninggalkan kesalahan yang bertahan lama. Namun, jika konstanta integral k_i terlalu besar, sistem dapat menjadi tidak stabil, sehingga mengakibatkan osilasi atau ayunan yang tidak diinginkan. Oleh karena itu, penyesuaian parameter ini sangat penting untuk menjaga keseimbangan antara respons cepat dan stabilitas sistem.

Komponen *Derivative* (D) bertugas untuk mengantisipasi perubahan yang akan terjadi pada sistem dengan memperhatikan laju perubahan eror dari waktu ke waktu. Dengan mendeteksi perubahan yang cepat pada eror, komponen ini dapat memberikan respons yang lebih cepat, yang pada akhirnya membantu mengurangi *overshoot* dan meningkatkan stabilitas sistem secara keseluruhan. *Overshoot* terjadi ketika *output* sistem melebihi *setpoint* sebelum stabil pada nilai yang diinginkan. Rumus dasar yang digunakan untuk komponen derivatif ini dapat dilihat pada persamaan 3.7 berikut.

$$u(t) = k_d \frac{de(t)}{dt} \tag{3.7}$$

Di mana:

- k_d adalah konstanta derivatif.
- $\frac{de(t)}{dt}$ adalah turunan dari *error* terhadap waktu.

Fungsi utama dari komponen ini adalah untuk memperkirakan eror yang mungkin terjadi di masa mendatang berdasarkan laju perubahan eror saat ini. Komponen ini membantu mengurangi *overshoot* dan mempercepat respons sistem dalam mencapai *setpoint*. Namun, perlu diingat bahwa jika konstanta derivatif k_d terlalu besar, sistem bisa menjadi terlalu sensitif terhadap perubahan kecil, yang justru dapat menyebabkan ketidakstabilan sehingga harus diatur dengan hati-hati untuk mencapai keseimbangan antara kecepatan respons dan stabilitas sistem.

3.3.5.2 Penggabungan Komponen PID

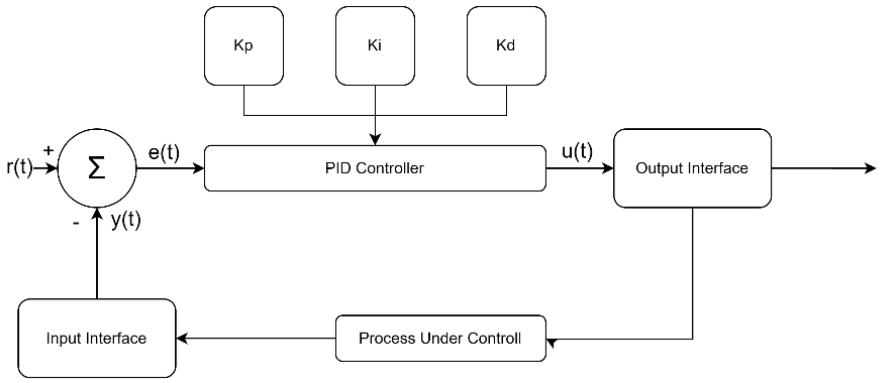
Ketiga komponen PID sebelumnya digabungkan untuk membentuk komponen penuh pengontrol PID dengan rumus umum yang dapat dijabarkan pada persamaan 3.8 di bawah ini.

$$u(t) = k_p e(t) + k_i \int e(t)\, dt + k_d \frac{de(t)}{dt} \quad (3.8)$$

Pada pengontrol PID, masing-masing komponen memiliki peran yang saling melengkapi untuk mencapai performa sistem yang optimal. Nilai k_p, k_i, dan k_d harus disetel dengan hati-hati agar sistem dapat mencapai *setpoint* dengan cepat, tanpa *overshoot* berlebihan, dan tanpa osilasi yang tidak perlu.

3.3.5.3 Implementasi PID *Controller* pada Sistem

Agar mendapatkan kualitas yang baik, lebar bukaan keran disesuaikan setiap waktu menggunakan PID berdasarkan *setpoint* kecepatan aliran cairan yang dipengaruhi oleh waktu. Data dari parameter yang masuk akan diproses oleh pengontrol PID dengan nilai *Kp, Ki,* dan *Kd*. Nilai dari PID ditentukan melalui metode *tuning* PSO, sehingga tidak perlu menentukan nilainya pada bagian ini. Hasil dari pengontrol PID akan diproses ulang dan dibuat sedemikian rupa agar menjadi nilai input baru untuk PID selanjutnya dengan harapan memiliki hasil nilai yang mendekati *setpoint* yang ditentukan. Proses ini dapat dilihat pada Gambar 3.15.



Gambar 3.15 Penerapan PID

Proses kontrol dalam penerapan PID melibatkan perbandingan antara *reference voltage* atau *setpoint*, $r(t)$, dan *real output voltage* atau *process variable*, $y(t)$, yang menghasilkan *error voltage* yang disebut sebagai $e(t)$. *Error voltage* ini kemudian dimasukkan ke dalam PID yang menghasilkan variabel kontrol $u(t)$ sebagai *output*. Tujuan dari PID adalah meminimalkan eror antara $y(t)$ dan $r(t)$, dengan menyesuaikan input kontrol berdasarkan *feedback*. $u(t)$ ditentukan dengan mempertimbangkan nilai *Proportional*, *Integral*, dan *Derivatif* dari $e(t)$, serta koefisien Kp, Ki, dan Kd. Koefisien-koefisien ini saling terkait dan memainkan peran penting dalam *tuning controller* untuk kinerja optimal. Dalam bentuk ideal, $u(t)$ dari pengontrol PID adalah jumlah dari ketiganya (Wang, L., 2020)

Turunan pertama mewakili perubahan kecepatan dari eror sedangkan turunan kedua mewakili percepatan dari eror. Dengan membatasi percepatan dari eror agar tidak semakin besar, turunan kedua ditambahkan untuk mendapatkan respons sistem yang lebih cepat, *overshoot* yang lebih rendah, dan peningkatan stabilitas sistem (Xiaodong, Z., et al., 2009). Hal ini dapat memberikan kontrol yang lebih baik pada tiap parameter, tetapi meningkatkan parameter ke turunan yang

lebih tinggi bisa menyebabkan masalah *rise time*, peningkatan kompleksitas dalam penyetelan parameter, dan peningkatan gangguan atau *noise* (Mary, A. H., 2011).

Berdasarkan sistem yang akan dibuat, nilai aktual kecepatan air dan roda berputar saat ini menjadi *real output voltage* melalui tahap *input interface*. *Reference voltage* didapatkan dari nilai *setpoint* melalui tahap pengumpulan data, sedangkan *error voltage* merupakan selisih dari nilai *setpoint* dengan nilai aktual kecepatan air dan putaran roda. Hasilnya berupa nilai *output interface*, besaran bukaan aktuator dan nilai kontrol untuk tahap *process under control*.

3.4 Optimasi dengan Metode *Tuning* PSO

Particle Swarm Optimization (PSO) adalah salah satu metode optimasi stokastik yang terinspirasi oleh gerakan dan kerjasama dari sekawanan burung. Konsep utama di balik algoritma PSO serupa dengan bagaimana burung-burung mencari makanan dalam area terbatas (Li, X., et al., 2013). Setiap partikel dalam PSO mewakili solusi potensial dan memiliki posisi serta kecepatan yang diperbarui setiap iterasi berdasarkan dua nilai utama.

Personal Best (pBest) merupakan nilai terbaik yang telah dicapai oleh partikel tertentu selama pergerakannya dalam ruang pencarian. Setiap partikel menyimpan informasi tentang posisi terbaik yang pernah dicapainya dan menggunakan informasi ini untuk mengarahkan pergerakannya ke posisi yang lebih baik di masa depan.

Global Best (gbest) merupakan nilai terbaik yang telah ditemukan oleh seluruh populasi partikel dalam ruang pencarian. Informasi ini disebarkan ke semua

partikel, sehingga setiap partikel dapat menyesuaikan pergerakannya berdasarkan pengalaman terbaik dari seluruh kelompok.

Keunggulan penerapan *tuning* PSO pada PID terletak pada kemampuannya untuk menemukan solusi yang optimal dalam ruang pencarian yang kompleks. PSO mampu menangani variasi dan interaksi antar tiap parameter PID dengan cara yang efisien, sehingga dapat menghasilkan konfigurasi kontrol yang lebih baik. Dengan demikian, penerapan *tuning* PSO pada PID menjadi salah satu pendekatan yang efektif dalam meningkatkan kualitas kontrol sistem pengaliran cairan secara otomatis dan adaptif (Mustafa, N., & Hashim, F. H., 2020).

3.4.1 Persamaan Dasar PSO

Dalam metode PSO, terdapat sistem perhitungan yang digunakan untuk menentukan perubahan kecepatan dan posisi setiap partikel dalam setiap iterasi. Pertama, perlu diperbarui kecepatan partikel menggunakan persamaan 3.9 berikut.

$$v_{i,m}(t+1) = w \,.\, v_{i,m}(t) + c_1 \,.\, rand() \,.\, (pbest_{i,m} - x_{i,m}(t)) + c_2 \,.\, rand() \,.\, (gbest_m - x_{i,m}(t)) \quad (3.9)$$

Dalam persamaan 3.9 tersebut, $v_{i,m}(t+1)$ adalah kecepatan baru dari partikel i pada dimensi m dan iterasi $t+1$. w adalah bobot inersia yang mengatur seberapa besar pengaruh kecepatan sebelumnya terhadap kecepatan baru. c_1 dan c_2 adalah konstanta positif yang menentukan pengaruh pbest dan gbest terhadap perubahan kecepatan. $rand()$ adalah fungsi acak antara 0 dan 1. $pbest_{i,m}$ adalah posisi terbaik yang pernah dicapai oleh partikel i pada dimensi m. Ini adalah titik yang memberikan nilai terbaik (*fitness*) yang dicapai oleh partikel tersebut selama

iterasi pencarian. $x_{i,m}(t)$ adalah posisi partikel i pada dimensi m pada waktu t. Ini menunjukkan lokasi saat ini dari partikel dalam ruang pencarian. $gbest_m$ adalah posisi terbaik global yang ditemukan oleh seluruh partikel dalam populasi pada dimensi m. Ini merupakan solusi terbaik yang ditemukan oleh seluruh *swarm* hingga waktu t. $x_{i,m}(t))$ digunakan lagi untuk menghitung perbedaan antara posisi terbaik individu *pbest* dan posisi saat ini dari partikel, serta antara gbest dan posisi saat ini. Selanjutnya, posisi partikel diperbarui menggunakan persamaan 3.10.

$$x_{i,m}(t+1) = x_{i,m}(t) + v_{i,m}(t+1) \quad (3.10)$$

Dalam persamaan 3.10, $x_{i,m}(t+1)$ adalah posisi baru dari partikel i pada pada dimensi m dan iterasi $t+1$. Dan $v_{i,m}(t+1)$ adalah kecepatan baru yang telah dihitung sebelumnya.

3.4.2 Persiapan Data dan Parameter

Langkah pertama adalah melakukan persiapan data dan parameter yang diperlukan untuk simulasi. Data input utama yang diperlukan adalah *hasil_setpoint* yang mengandung informasi "Putaran Roda" (RPM) dan "*Setpoint* Aliran Air" (mL/s). Sistem akan melakukan pemeriksaan apakah *hasil_setpoint* memiliki kolom yang sesuai. Jika kolom yang dibutuhkan ada, maka data RPM akan disimpan dalam variabel *rpm_data* dan nilai setpoint aliran air disimpan dalam *flow_setpoints*. Selain itu, penentuan parameter PID seperti Kp, Ki, dan Kd dilakukan dengan menginisialisasi nilai acak dalam rentang tertentu sebagai batas awal. Kemudian, parameter optimasi PSO juga disiapkan, seperti ukuran *swarm*, jumlah iterasi maksimum, serta koefisien inersia dan kognitif yang akan mengatur

pergerakan partikel dalam optimasi. Langkah-langkah proses ini dapat dijabarkan pada poin-poin berikut.

a. Periksa apakah *hasil_setpoint* memiliki kolom Menit dan *Setpoint* (mL/s).
b. Jika kolom ada, simpan data putaran roda dalam *rpm_data* dan *setpoint* aliran air dalam *flow_setpoints*.
c. Tentukan batas parameter PID dengan nilai acak dalam rentang tertentu.
d. Tentukan parameter PSO seperti ukuran *swarm*, jumlah iterasi maksimum, koefisien inersia, dan koefisien kognitif.

3.4.3 Simulasi PID dengan fungsi *Fitness* PSO

Dalam sistem ini, fungsi *fitness* dirancang untuk mengevaluasi kinerja parameter PID (Kp, Ki, Kd) dalam mengendalikan kecepatan aliran menuju *setpoint* yang diinginkan. Evaluasi ini dilakukan dengan mensimulasikan respons sistem terhadap *setpoint* yang telah didapatkan dari tahap pengumpulan data, dan menghitung total eror menggunakan pendekatan ITAE, di mana semakin kecil nilai eror total, semakin baik kontrolnya. Proses dimulai dengan menentukan *setpoint* dan kondisi awal untuk kecepatan air. Pada setiap iterasi simulasi, fungsi PID akan menghitung nilai eror (selisih antara *setpoint* dan *output* saat ini), memperbarui integral untuk koreksi akumulasi eror, dan menghitung derivatif untuk koreksi perubahan eror. Nilai PID yang dihasilkan digunakan untuk menyesuaikan *output* kecepatan air. Setelah iterasi selesai, selisih akhir antara *setpoint* dan output disimpan sebagai bagian dari total eror.

Fungsi *fitness* akan memanggil simulasi ini untuk setiap data *setpoint* yang diberikan, menghasilkan total eror yang mengukur efektivitas kontrol parameter

PID tersebut. Total eror ini kemudian digunakan sebagai nilai *fitness*, yang nantinya akan diminimalkan oleh PSO untuk menemukan parameter PID optimal. Secara sederhana ini telah dibahas pada Gambar 3.19.

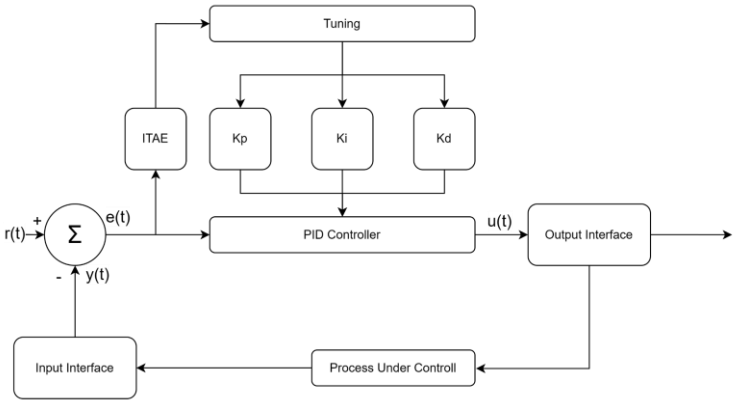
Setelah data dan parameter siap, tahap ini melakukan simulasi PID untuk setiap *setpoint* pada data RPM. Fungsi *simulate_pid()* didefinisikan untuk menerima nilai Kp, Ki, dan Kd, yang kemudian digunakan untuk menghitung respons PID. Pada setiap titik data di *rpm_data*, simulasi PID dilakukan sebanyak 200 iterasi dengan interval waktu (dt) tertentu. Setiap iterasi menghitung eror antara *setpoint* dan variabel proses (*process_var*). Integral eror diakumulasikan, tetapi dibatasi untuk menghindari efek *windup*. Derivatif eror dihitung berdasarkan perubahan eror sebelumnya, dan *output* PID digunakan untuk memperbarui *process_var*. Hasil akhirnya adalah total eror yang dihitung untuk setiap titik data. Langkah-langkah proses ini dapat dijabarkan pada poin-poin berikut.

a. Definisikan fungsi *simulate_pid()* yang menerima parameter Kp, Ki, dan Kd.
b. Untuk setiap titik data pada *rpm_data*, lakukan simulasi PID sebanyak 200 iterasi dengan interval waktu dt.
c. Pada setiap iterasi, lakukan langkah berikut lalu hitung total eror.
    - Hitung eror antara *setpoint* dan *process_var*.
    - Akumulasikan eror dalam integral dan batasi nilainya untuk menghindari *windup*.
    - Hitung derivatif berdasarkan perubahan eror sebelumnya.
    - Gunakan PID untuk menghasilkan *output* dan perbarui *process_var*.

3.4.4 Proses *Tuning* PSO

Tahapan pertama dalam proses *tuning* PSO adalah inisialisasi jumlah partikel atau *swarm size*, kecepatan awal (*velocity*) atau inersia, nilai kognitif, sosial dan batas pencarian. Pada umumnya, jumlah populasi partikel berkisar 20 hingga 50, tetapi pada praktiknya semakin besar maka semakin baik (Piotrowski, A., Napiorkowski, J., & Piotrowska, A., 2020). Selanjutnya, setiap partikel diinisialisasi dengan tiga dimensi yang mewakili nilai-nilai *Kp, Ki,* dan *Kd*. Nilai-nilai ini juga diinisialisasi secara acak berdasarkan rentang yang ditetapkan untuk masing-masing parameter. Masing-masing rentang nilai pada *Kp, Ki,* dan *Kd* adalah 0 hingga 10, 0 hingga 5, dan 0 hingga 5. Ketiga rentang nilai tersebut menjadi didapatkan berdasarkan penelitian terdahulu yang memiliki sistem yang sama dengan kebutuhan dari desain sistem yang dirancang pada penelitian ini, yakni memiliki prioritas terhadap respons yang tinggi.

Selama iterasi, kecepatan dan posisi pada tiap partikel diperbarui menggunakan persamaan 3.9 dan 3.10 yang telah dibahas sebelumnya. Hal ini berdasarkan nilai *pbest* dan gbest serta bobot inersia yang diatur dinamis. Melalui serangkaian iterasi, partikel secara bertahap mendekati solusi yang optimal, di mana parameter PID mencapai nilai yang meminimalisir eror sistem dan meningkatkan kinerja kontrol sistem. Secara umum dapat digambarkan pada Gambar 3.16 berikut.



Gambar 3.16 Proses tuning PSO pada PID Controller

Kemudian, setiap partikel dievaluasi berdasarkan kinerjanya dalam sistem PID dengan menggunakan fungsi objektif *Integral of Time-Weighted Absolute Error* (ITAE). Fungsi objektif ini mengukur sejauh mana konfigurasi parameter PID yang diusulkan meminimalkan kesalahan sistem. Persamaan 3.11 adalah persamaan umum pada perhitungan ITAE, dimana kesalahan antara *setpoint* dan *output* sistem pada waktu t.

$$ITAE = \int_0^\infty t \cdot | \, e(t) \, | \, dt \tag{3.11}$$

Selama setiap iterasi, kecepatan dan posisi setiap partikel diperbarui dengan menggunakan persamaan yang melibatkan nilai pbest dan gbest, serta bobot inersia. Bobot inersia adalah parameter yang mengontrol pengaruh dari kecepatan sebelumnya dalam pembaruan kecepatan saat ini. Kecepatan partikel dihitung dengan mempertimbangkan pbest dan gbest, serta jarak antara posisi saat ini dengan pbest dan gbest. Bobot inersia diatur secara dinamis untuk menyeimbangkan antara eksplorasi global atau penjelajahan ruang pencarian dan eksploitasi lokal atau penyempurnaan solusi yang ada. Setelah kecepatan baru dihitung, posisi setiap partikel diperbarui. Posisi baru ini merepresentasikan parameter PID yang telah disesuaikan, dan akan dievaluasi kembali dalam iterasi berikutnya.

Proses ini diulang melalui serangkaian iterasi. Selama iterasi tersebut, partikel terus-menerus memperbarui pbest dan gbest berdasarkan informasi terbaru, dengan bobot inersia yang disesuaikan untuk meningkatkan keseimbangan antara pencarian global dan lokal. Seiring berjalannya waktu, partikel semakin mendekati

solusi optimal, yaitu parameter PID yang meminimalkan kesalahan sistem dan meningkatkan kinerja kontrol sistem. Setelah itu, PSO akan menghasilkan parameter PID yang optimal. Parameter ini kemudian diterapkan pada PID untuk sistem kontrol yang lebih baik dengan kesalahan sistem yang minimal.

Sebelum parameter PID yang telah dioptimalkan digunakan pada sistem kontrol, PSO akan mengulangi keseluruhan proses hingga mendapatkan parameter PSO yang sesuai. Proses ini dilakukan dengan cara meningkatkan, menurunkan, dan menyempitkan batasan atas maupun bawah dari tiap parameter berdasarkan ukuran pencarian sebelumnya.

Optimasi PSO dilakukan untuk menemukan parameter PID yang optimal. Pada langkah ini, PSO diinisialisasi dengan parameter seperti batas bawah (lb), batas atas (ub), ukuran *swarm*, iterasi maksimum, serta koefisien inersia, kognitif, dan sosial. Setiap partikel dalam *swarm* diatur untuk mengikuti aturan Kp > Ki > Kd. Pada setiap iterasi, posisi setiap partikel diperbarui berdasarkan kecepatan baru yang dihitung dari komponen inersia, kognitif, dan sosial. *Fitness score* dihitung untuk mengevaluasi kualitas parameter pada setiap posisi partikel. Jika *fitness score* lebih baik daripada personal best atau global best, maka nilai tersebut akan diperbarui. Iterasi dihentikan ketika tidak ada perubahan signifikan atau sudah mencapai iterasi maksimum, dan parameter PID optimal beserta *fitness score* terbaik akan ditampilkan. Langkah-langkah proses ini dapat dijabarkan pada poin-poin berikut.

a. Inisialisasi PSO dengan parameter seperti batas bawah (lb), batas atas (ub), ukuran swarm, iterasi maksimum, dan koefisien.

b. Pastikan aturan Kp > Ki > Kd untuk setiap partikel dalam *swarm*.
c. Perbarui posisi setiap partikel berdasarkan kecepatan baru yang dihitung dari komponen inersia, kognitif, dan sosial.
d. Hitung *fitness score* untuk setiap posisi partikel.
e. Jika *fitness score* lebih baik dari pbest atau gbest, maka perbarui nilai tersebut.
f. Hentikan iterasi ketika tidak ada perubahan signifikan atau sudah mencapai iterasi maksimum.
g. Tampilkan parameter PID optimal beserta *fitness score* terbaik.

3.4.5 Simulasi Respons PID

Setelah parameter PID yang optimal ditemukan, langkah berikutnya adalah melakukan simulasi respons sistem menggunakan parameter PID tersebut. Simulasi dimulai dengan menginisialisasi variabel seperti *process_var*, integral, dan *prev_error*. Pada setiap waktu simulasi, eror antara *setpoint* dan *process_var* dihitung. Nilai integral dan derivatif diperbarui, kemudian output dihitung menggunakan persamaan PID dengan Kp, Ki, dan Kd yang optimal. *Output* ini digunakan untuk memperbarui *process_var*, yang menunjukkan laju aliran air. Setiap hasil *process_var* dicatat dalam *flow_history* untuk memantau respons PID terhadap perubahan *setpoint*. Langkah-langkah proses ini dapat dijabarkan pada poin-poin berikut.

a. Inisialisasi variabel seperti *process_var*, integral, dan *prev_error*.
b. Pada setiap waktu simulasi, hitung eror antara *setpoint* dan *process_var*.

c. Perbarui nilai integral dan derivatif, lalu hitung *output* menggunakan PID dengan Kp, Ki, dan Kd optimal.
d. Gunakan *output* untuk memperbarui *process_var*, yang menunjukkan laju aliran air.
e. Simpan nilai *process_var* dalam *flow_history* untuk mencatat respons PID terhadap perubahan *setpoint*.

3.4.6 Evaluasi Kinerja PID *Controller*

Evaluasi kinerja PID *Controller* dilakukan untuk memastikan bahwa sistem kontrol bekerja sesuai dengan kebutuhan yang diinginkan. Beberapa parameter utama yang digunakan dalam evaluasi kinerja PID dapat dijelaskan pada paragraf berikutnya.

Rise Time adalah waktu yang dibutuhkan oleh sistem untuk pertama kali mencapai nilai tertentu dari *setpoint*, biasanya dihitung dari 10% hingga 90% dari nilai *setpoint* (Dorf, R. C., & Bishop, R. H., 2011). *Rise Time* memberikan gambaran seberapa cepat sistem merespons perubahan *setpoint*. Semakin pendek waktu naik, semakin cepat sistem merespons, tetapi hal ini juga dapat meningkatkan risiko *overshoot* jika tidak diimbangi dengan pengaturan parameter PID yang tepat.

Overshoot adalah nilai maksimum yang dicapai oleh output sistem di atas *setpoint* sebelum stabil. Ini merupakan indikasi seberapa jauh sistem melampaui *setpoint* sebelum akhirnya berkurang dan stabil pada *setpoint*. *Overshoot* yang besar menunjukkan bahwa sistem mungkin terlalu agresif dalam merespons perubahan, yang dapat mengakibatkan ketidakstabilan atau osilasi. Mengurangi *overshoot* sering kali menjadi salah satu tujuan utama dalam mengoptimalkan parameter PID,

khususnya komponen *Derivative* yang berperan dalam mengantisipasi dan mengurangi *overshoot*.

Settling Time adalah waktu yang dibutuhkan oleh sistem untuk mencapai dan tetap berada pada batas toleransi di sekitar *setpoint*, biasanya 2% atau 5% dari *setpoint*, tanpa mengalami osilasi signifikan *setpoint* (Dorf, R. C., & Bishop, R. H., 2011). Waktu penetapan yang lebih cepat menunjukkan bahwa sistem lebih efisien dalam mencapai stabilitas setelah mengalami perubahan *setpoint*. Namun, dapat menyebabkan *overshoot* yang tinggi atau osilasi yang berkepanjangan jika terlalu cepat atau terjadi ketidakseimbangan antara kecepatan respons dan stabilitas.

Steady-State error adalah perbedaan antara *setpoint* dan nilai *output* setelah sistem telah mencapai kondisi stabil. Idealnya, *steady-state error* harus mendekati nol, yang menunjukkan bahwa sistem mampu mencapai *setpoint* dengan presisi tinggi. Komponen *Integral* dari PID berfungsi untuk mengeliminasi kesalahan *steady-state*, namun pengaturan yang tidak tepat dapat menyebabkan sistem menjadi lambat atau bahkan tidak stabil.

Evaluasi kinerja ini sangat penting karena membantu dalam mengidentifikasi apakah parameter PID yang digunakan sudah sesuai atau perlu disesuaikan. Jika kinerja sistem tidak memuaskan, misalnya, jika *rise time* terlalu lambat, *overshoot* terlalu tinggi, *settling time* terlalu lama, atau ada *steady-state error* yang signifikan, maka penyesuaian pada konstanta *Proportional* (k_p), *Integral* (k_i), dan *Derivative* (k_d) perlu dilakukan.

Evaluasi ini bertujuan untuk mengukur performa PID dalam memenuhi kebutuhan respons dan stabilitas aktuator dengan cara menilai berbagai parameter,

termasuk nilai masing-masing parameter PID yang telah dioptimalkan, jumlah populasi partikel, iterasi, waktu komputasi, serta nilai-nilai seperti *rise time*, *settling time*, *overshoot*, *steady state error*, dan ITAE.

Evaluasi kinerja dilakukan untuk mengukur seberapa baik respons kontrol PID yang telah dihasilkan. Beberapa metrik yang digunakan dalam evaluasi antara lain adalah *overshoot*, *settling time*, *rise time*, dan *steady-state error*. *Overshoot* mengukur persentase kelebihan respons awal terhadap *setpoint*. *Settling time* mengukur waktu yang dibutuhkan untuk mencapai kondisi stabil dalam toleransi tertentu. *Rise time* mengukur waktu yang diperlukan untuk mencapai 90% dari *setpoint* pertama kali, sedangkan *steady-state error* adalah eror antara nilai *steady-state* dan *setpoint*. Hasil evaluasi ini kemudian ditampilkan dalam bentuk tabel. Langkah-langkah proses ini dapat dijabarkan pada poin-poin berikut.

a. Hitung beberapa metrik evaluasi kinerja PID, yakni *overshoot*, *settling time*, *rise time*, dan *steady-state error*.
b. Tampilkan hasil evaluasi dalam bentuk tabel.

3.4.7 Visualisasi Hasil Simulasi

Langkah terakhir adalah visualisasi hasil simulasi untuk mempermudah analisis. Visualisasi dilakukan dengan menggunakan *matplotlib* untuk membuat grafik waktu dengan *flow_history* dan *setpoint*. Grafik ini menunjukkan respons aliran air terhadap perubahan *setpoint*, serta waktu *settling time* dan *rise time* yang ditandai pada grafik. Label sumbu-x dan sumbu-y juga ditambahkan untuk kejelasan interpretasi. Langkah-langkah proses ini dapat dijabarkan pada poin-poin berikut.

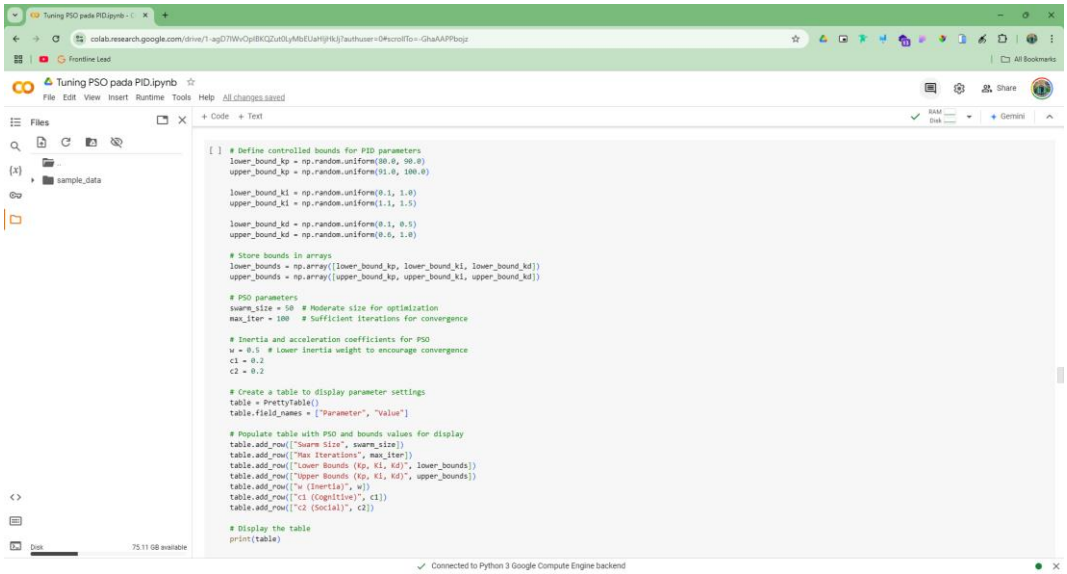
a. Buat plot waktu dengan *flow_history* dan *setpoint* menggunakan *matplotlib*.

b. Tandai waktu *settling time* dan *rise time* pada grafik.

c. Tambahkan label pada sumbu-x dan sumbu-y untuk kejelasan.

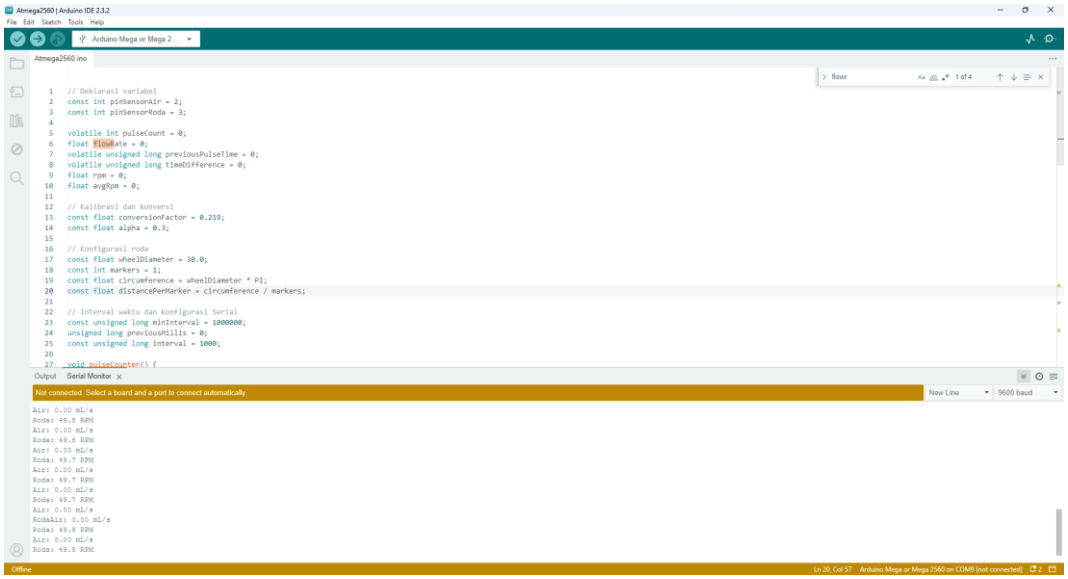
3.5 Implementasi Sistem

Setelah desain sistem dibuat, mulai dari desain fisik hingga otak dari proses sistem kontrol, desain sistem tersebut diimplementasikan pada rangkaian akhir untuk dilakukan pengujian melalui *source code* atau serangkaian proses di dalam program yang dibuat melalui *Arduino IDE* maupun *Google Colaboration* dan *Node-Red.* Implementasi evaluasi dari sistem melalui *Google Colaboration* tersebut dapat dilihat pada Gambar 3.17 di bawah ini.



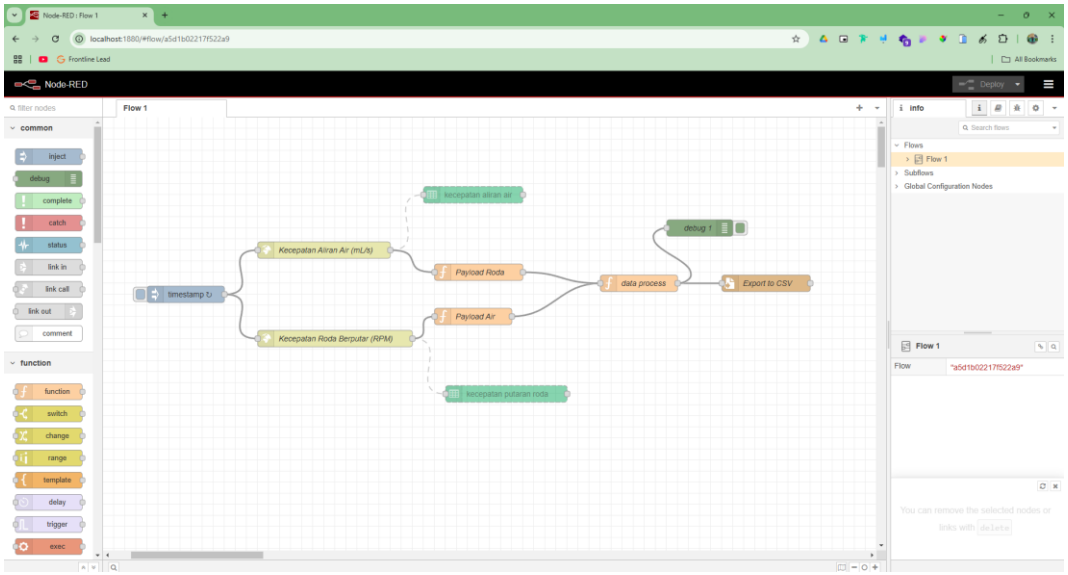
Gambar 3.17 Implementasi sistem pada *Google Colaboration*

Sedangkan, implementasi desain sistem dari hasil analisis serta pencarian parameter optimal PID terhadap rangkaian fisik melalui *Arduino* dapat dilihat pada Gambar 3.18 di bawah ini.



Gambar 3.18 Implementasi sistem pada *Arduino IDE*

Dalam menangani proses pengumpulan data maupun data dari hasil pengujian, dapat dilakukan melalui *Node-Red* dengan hasil implementasi yang dapat dilihat pada Gambar 3.19 di bawah ini.



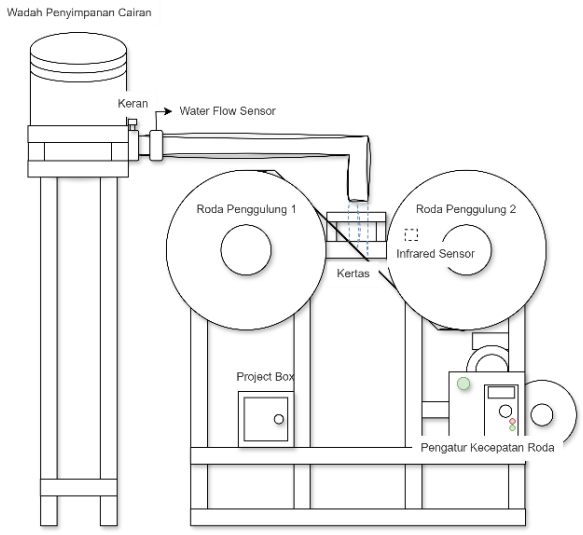
Gambar 3.19 Implementasi sistem pada *Node-Red*

Adapun hasil akhir dari desain rangkaian ini dapat implementasi menjadi 2 jenis, yakni desain akhir rangkaian pengumpulan data dan pengujian. Perbedaan

dari keduanya terletak pada bagian aktuator yang tidak ada di pengumpulan data, karena rangkaian tersebut hanya melakukan pengumpulan data dan kirim data ke *Blynk*. Penjelasan kedua rangkaian tersebut dapat dilihat pada poin berikutnya.

3.6.1 Rangkaian Pengumpulan Data

Rangkaian pengumpulan data hanya memiliki 2 fungsi utama, yakni mengumpulkan data dari sensor dan mengirimkannya ke *Dashboard Blynk* yang kemudian digunakan untuk menentukan *setpoint.* Perbedaannya hanya pada bagian kontrol keran yang masih manual, yakni tanpa dipasang *servo*. Adapun hasil akhir dari rangakaian pengumpulan data dapat dilihat pada Gambar 3.20 di bawah ini.

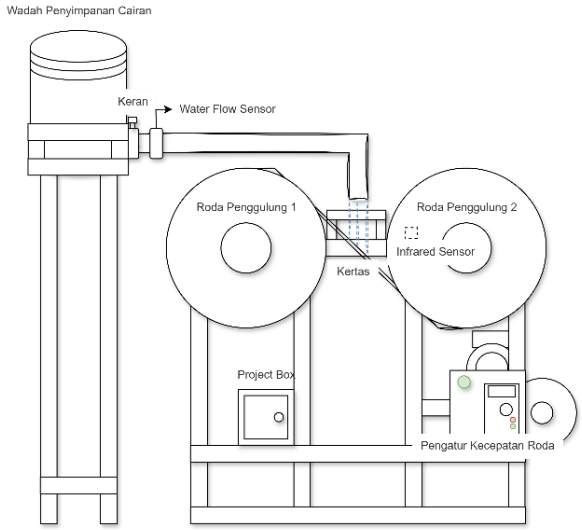


Gambar 3.20 Desain akhir rangkaian pengumpulan data

3.6.1 Rangkaian Pengujian

Rangkaian pada tahap pengujian hampir sama dengan rangkaian pengumpulan data. Yang membedakan hanya pada penentuan *setpoint* dan proses menggerakkan *servo* berdasarkan hasil dari kalkulasi PID yang diatur oleh *Arduino*.

Adapun hasil akhir dari rangakaian pengumpulan data dapat dilihat pada Gambar 3.21 di bawah ini.



Gambar 3.21 Desain akhir rangkaian pengujian

3.6.2 Kode Sumber

Desain sistem pada penelitian ini diprogram agar bisa memenuhi permintaan proses yang dibutuhkan, mulai dari mengoperasikan tahap kalibrasi dan evaluasi komponen, rangkaian pengumpulan data, analisis data, penentuan *setpoint*, rangkaian pengujian, hingga hasil akhir dari rangkaian yang dapat dilihat melalui bagian lampiran. Adapun kode sumber yang digunakan untuk PID, ITAE maupun PSO dapat dijelaskan pada kode sumber di bawah ini.

```
[ ]  def simulate_pid(Kp, Ki, Kd, time_data, flow_setpoints):
         dt = 0.1  # Interval waktu
         itae = 0  # Inisialisasi ITAE
         process_var = 0  # Nilai awal posisi keran (0 derajat)

         for i in range(len(time_data)):
             setpoint = flow_setpoints[i]
             integral = 0
             prev_error = 0

             for t in range(200):  # Simulasi selama 200 iterasi per titik data
                 time = t * dt  # Waktu simulasi saat ini
                 error = setpoint - process_var
                 integral += error * dt

                 derivative = (error - prev_error) / dt
                 output = Kp * error + Ki * integral + Kd * derivative

                 # Perbarui variabel proses
                 process_var += output * 0.01
                 process_var = np.clip(process_var, 0, 90)

                 # Hitung ITAE
                 itae += time * abs(error)  # Bobot waktu pada absolute error

                 prev_error = error

         return itae
```

Gambar 3.22 Kode Sumber PID beserta nilai *fitness* berbasis ITAE

Persamaan 3.5 yang menjelaskan rumus untuk nilai *Kp* terdapat pada baris ke-17, dimana kontribusinya dihitung dengan *Kp * error*. Nilai *error* sendiri didapatkan melalui *error = setpoint - process_var*. Persamaan 3.6 yang digunakan untuk menentukan *Ki* terdapat pada baris ke-14, yakni nilai integral diakumulasikan secara iteratif menggunakan *integral += error * dt*. Persamaan 3.7 yang menentukan nilai *Kd* terdapat pada baris ke-16, yakni *derivative = (error - prev_error) / dt*. Ketiga nilai tersebut menjadi satu bagian yang sama, yakni pada baris ke-17. Digunakan untuk menentukan hasil akhir dari ketiganya berdasarkan persamaan 3.8 dengan menggunakan *Kp * error + Ki * integral + Kd * derivative*. Hasil tersebut ditentukan nilai optimalnya menggunakan ITAE sesuai persamaan 3.11 yang terdapat pada baris ke-24, yakni *itae += time * abs(error)*.

Kemudian fungsi PID dengan ITAE di atas dipanggil melalui fungsi *fitness* dengan parameter yang akan didapatkan melalui proses PSO dengan data

berdasarkan hasil dari tahap pengumpulan dan analisis data. Fungsi tersebut dapat dilihat pada kode sumber di bawah ini.

```
[ ]  def fitness(params):
         Kp, Ki, Kd = params
         return simulate_pid(Kp, Ki, Kd, time_data, flow_setpoints)
```

Gambar 3.23 Kode sumber fungsi *fitness*

Setelah fungsi PID dan nilai *fitness* berbasis ITAE ditentukan, tahap selanjutnya adalah proses *tuning* PSO. Proses ini dapat dilihat melalui fungsi pada kode sumber di bawah ini.

```
class PSO:
    def __init__(self, fitness, lower_bounds, upper_bounds, swarm_size, max_iter, w, c1, c2):
        """
        Inisialisasi algoritma PSO (Particle Swarm Optimization).
        :param fitness: Fungsi fitness yang ingin diminimalkan.
        :param lower_bounds: Batas bawah untuk ruang pencarian.
        :param upper_bounds: Batas atas untuk ruang pencarian.
        :param swarm_size: Jumlah partikel dalam swarm.
        :param max_iter: Jumlah iterasi maksimum.
        :param w: Koefisien inersia (inertia weight).
        :param c1: Koefisien akselerasi personal (kognitif).
        :param c2: Koefisien akselerasi global (sosial).
        """
        self.fitness = fitness
        self.lower_bounds = lower_bounds
        self.upper_bounds = upper_bounds
        self.swarm_size = swarm_size
        self.max_iter = max_iter
        self.w = w
        self.c1 = c1
        self.c2 = c2

    def optimize(self):
        # Inisialisasi partikel
        # Partikel diinisialisasi secara acak dalam ruang pencarian.
        particles = np.random.uniform(self.lower_bounds, self.upper_bounds, (self.swarm_size, len(self.lower_bounds)))
        velocities = np.zeros_like(particles)  # Kecepatan awal semua partikel diatur ke nol.
        personal_best_positions = np.copy(particles)  # Posisi pbest awal adalah posisi awal partikel.
        personal_best_values = np.array([self.fitness(p) for p in particles])  # Evaluasi fitness awal untuk setiap partikel.
        global_best_position = personal_best_positions[np.argmin(personal_best_values)]  # Posisi gbest awal.
        global_best_value = np.min(personal_best_values)  # Nilai fitness gbest awal.
```

Gambar 3.24 Kode sumber fungsi PSO 1

```
# Loop utama optimasi
for iter_num in range(self.max_iter):
    for i in range(self.swarm_size):
        # Update kecepatan partikel berdasarkan rumus PSO.
        velocities[i] = (self.w * velocities[i]  # Inersia
                         + self.c1 * np.random.random() * (personal_best_positions[i] - particles[i])  # Komponen personal
                         + self.c2 * np.random.random() * (global_best_position - particles[i]))  # Komponen global

        # Update posisi partikel berdasarkan kecepatan.
        particles[i] += velocities[i]

        # Pastikan partikel tetap dalam batas ruang pencarian.
        particles[i] = np.clip(particles[i], self.lower_bounds, self.upper_bounds)

        # Evaluasi nilai fitness untuk posisi baru.
        current_value = self.fitness(particles[i])
        if current_value < personal_best_values[i]:  # Jika nilai fitness lebih baik dari pbest
            personal_best_values[i] = current_value
            personal_best_positions[i] = particles[i]  # Perbarui posisi gbest.

    # Update posisi dan nilai gbest jika ditemukan yang lebih baik.
    new_global_best_value = np.min(personal_best_values)
    if new_global_best_value < global_best_value:
        global_best_value = new_global_best_value
        global_best_position = personal_best_positions[np.argmin(personal_best_values)]

    # Cetak informasi iterasi saat ini.
    print(f'End of iteration {iter_num + 1}, Best Score: {global_best_value}')
    print(f'Personal Best Values: {personal_best_values}')
    print(f'Global Best Position: {global_best_position}')

# Kembalikan posisi gbest dan nilai fitness terbaik.
return global_best_position, global_best_value
```

Gambar 3.25 Kode sumber fungsi PSO 2

3.6.3 Tampilan Desain *Interface* dengan *Blynk*

Pada penelitian ini, *Blynk* digunakan untuk menampilkan data seperti kecepatan air, kecepatan putaran roda penggulung dan data lain yang dibutuhkan. Data yang ditampilkan pada *Blynk* didasari oleh pengiriman data dari *Arduino* yang tersambung pada jaringan internet menggunakan modul *ESP8266* pada tiap iterasi yang telah dilakukan. Kemudian, data-data tersebut akan ditampilkan bersama dalam satu grafik untuk melihat riwayat data tiap parameter yang telah masuk. Tampilan tersebut dapat dilihat pada 3.26 di bawah ini.

Gambar 3.26 Tampilan *Dashboard Blynk*

3.6 Skenario Pengujian

Pengujian ini dilakukan untuk menentukan kombinasi parameter yang menghasilkan performa optimal pada PID menggunakan metode *tuning* PSO berdasarkan desain sistem yang telah dibuat. Tabel 3.2 digunakan sebagai acuan bagi peneliti untuk melakukan serangkaian pengujian.

Tabel 3.2 Skenario pengujian

Skenario	Kp	Ki	Kd	Hasil Statis	Hasil Dinamis
1A	…	…	…	…	…
1B	…	…	…	…	…
2A	…	…	…	…	…

Pengujian sendiri dilakukan sebanyak 2 iterasi percobaan variabel PSO dengan tiap iterasi dilakukan 2 kali evaluasi kinerja PID. Setiap evaluasi terdapat

hasil yang menentukan kriteria apa yang kurang dari hasil optimasi berdasarkan evaluasi kinerja PID yang telah dilakukan. Iterasi pada variabel PSO disajikan dalam bentuk tabel untuk memudahkan dalam akses informasi konfigurasi. Tabel tersebut dapat dilihat pada Tabel 3.3 di bawah ini.

Tabel 3.3 Iterasi variabel PSO

Skenario	**Lower Bounds (Kp, Ki, Kd)**	**Upper Bounds (Kp, Ki, Kd)**	**Inersia**	**C1**	**C2**
1A	…	…	…	…	…
1B	…	…	…	…	…
2A	…	…	…	…	…
2B	…	…	…	…	…

Setelah iterasi variabel PSO ditentukan, *tuning* PSO dijalankan dan menghasilkan parameter PID yang optimal berdasarkan nilai *fitness* atau ITAE terkecil. Hasil tersebut ditampilan berupa tabel hasil.

Setelah itu, simulasi pengujian evaluasi kinerja PID dilakukan dengan mempertimbangkan hasil analisis data, yaitu *setpoint* dengan nilai PID yang telah dioptimalkan. Skenario dari simulasi ini dibedakan menjadi 2, yakni simulasi evaluasi dengan *setpoint* tetap, dan *setpoint* dinamis. Dari kedua simulasi tersebut, diperoleh *rise time*, *settling time*, *overshoot*, dan *steady state error*. Hasil dari evaluasi ini akan ditampilkan melalui Tabel 3.4 berikut ini.

Tabel 3.4 Evaluasi kinerja PID dengan *tuning* PSO

Skenario	**Kp**	**Ki**	**Kd**	**Rise Time (s)**	**Settling Time (s)**	**Overshoot (%)**	**Steady State Error (%)**	**ITAE**
1A	…	…	…	…	…	…	…	…
1B	…	…	…	…	…	…	…	…
2A	…	…	…	…	…	…	…	…
2B	…	…	…	…	…	…	…	…

Setelah komponen PID yang optimal ditentukan melalui serangkaian pengujian dan evaluasi, rangkaian pengujian dibuat mengikuti alur proses tertentu. Pengujian implementasi PID dalam rangkaian ini melibatkan variasi kecepatan putaran roda, disesuaikan dengan kondisi operasional yang diterapkan oleh perusahaan.

Alur proses pada tahap pengujian ini hampir sama dengan alur pengumpulan data. Perbedaan utamanya berada pada penentuan *setpoint* dan proses penggerakan *servo* berdasarkan hasil kalkulasi PID yang ditambahkan ke dalam alur proses. Kemudian diterima oleh *ESP8266* untuk dikirim ke *Blynk*.

Langkah selanjutnya adalah menguji desain sistem yang dihasilkan oleh PSO terhadap PID untuk memastikan kesesuaiannya dengan kebutuhan penelitian. Hasil tersebut dibuat menjadi rata-rata tiap menit agar mempermudah dalam melakukan visualisasi data. Data pengujian lengkapnya akan dicantumkan pada bagian lampiran. Pengujian ini dirangkum dalam Tabel 3.5.

Tabel 3.5 Pengujian desain sistem

Menit	***Setpoint***	**Putaran roda aktual (RPM)**	**Kecepatan aliran aktual (mL/s)**	**Hasil**
1	…	…	…	
…	…	…	…	
n	…	…	…	

Pada tabel ini, setiap nilai yang tercatat merupakan rata-rata yang diperoleh setiap menit. Berdasarkan *setpoint* yang telah ditentukan, hasil pengujian menunjukkan apakah kecepatan air yang diperoleh sesuai dengan batas toleransi yang ditetapkan. Kolom hasil memiliki beberapa kriteria, yakni sesuai dengan *setpoint*, berada di batas toleransi minimal atau maksimal, atau mengalami kegagalan. Pengujian dilakukan satu kali iterasi dengan pencatatan nilai per menit

hingga iterasi selesai. Jika terjadi kegagalan yang signifikan, proses *tuning* PSO dan evaluasi kinerja PID atau bahkan pengumpulan data perlu diulang.

3.7 Hasil Analisis

Hasil analisis menunjukkan pentingnya penerapan pengontrol PID yang dioptimalkan dengan metode PSO untuk menjaga agar output cairan sesuai *setpoint* dan memastikan stabilitas sistem selama proses pelapisan kertas. Simulasi dilakukan untuk mengevaluasi parameter *rise time*, *settling time*, *overshoot*, dan *steady-state error* guna menentukan kombinasi parameter PID yang optimal. Selain itu, integrasi metode PSO dirancang untuk menghasilkan sistem yang adaptif terhadap variasi operasional, sehingga dapat menjaga performa stabil saat roda berputar dan memenuhi kebutuhan operasional sistem pelapisan secara efisien.

BAB IV

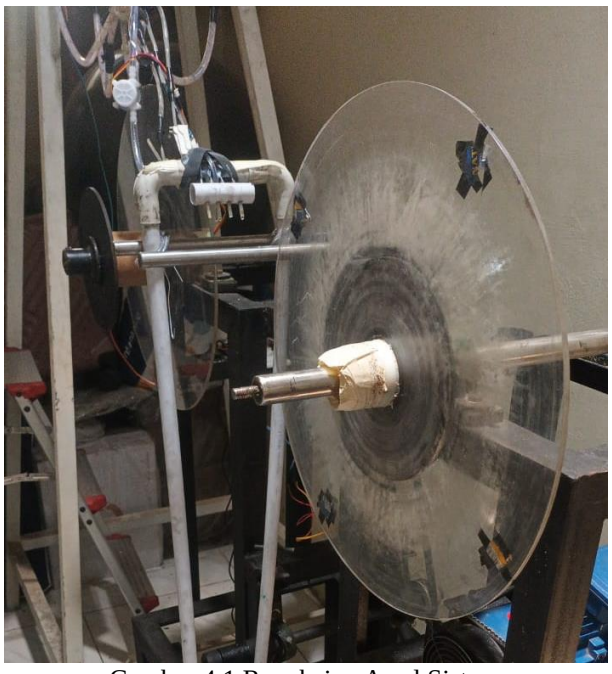
HASIL DAN PEMBAHASAN

4.1 Hasil

Bab ini menjelaskan tahap pengumpulan data, analisis data, evaluasi kinerja PID dengan tuning PSO, dan hasil pengujian pada sistem kontrol kecepatan air. Setiap hasil uji coba disajikan dalam bentuk tabel dan grafik yang mendukung hasil analisis secara kualitatif dan kuantitatif.

4.1.1 Rangkaian Pengumpulan Data

Langkah pertama sebelum melakukan pengujian adalah mempersiapkan rangkaian pengumpulan data yang dibutuhkan untuk mendapatkan informasi mengenai *setpoint* kecepatan air. Desain awal mesin yang akan dipasangkan dengan sistem kontrol yang telah dirancang dapat dilihat pada Gambar 4.1.



Gambar 4.1 Rangkaian Awal Sistem

Rangkaian pengumpulan data terdiri dari pemasangan penanda pada roda, pemasangan sensor *infrared* di belakang roda, dan pemasangan sensor *waterflow* pada selang. Penanda yang digunakan berbahan kertas tahan air berwarna putih, ditempelkan pada sisi luar bagian belakang kaca penekan roda. Untuk mempermudah pembacaan oleh sensor *infrared*, bagian bawah penanda diberi tonjolan dari kertas yang digumpalkan, sehingga memunculkan perbedaan fisik yang signifikan dibandingkan area sekitarnya. Gambar 4.2 menunjukkan hasil pemasangan penanda.

Gambar 4.2 Hasil pemasangan penanda

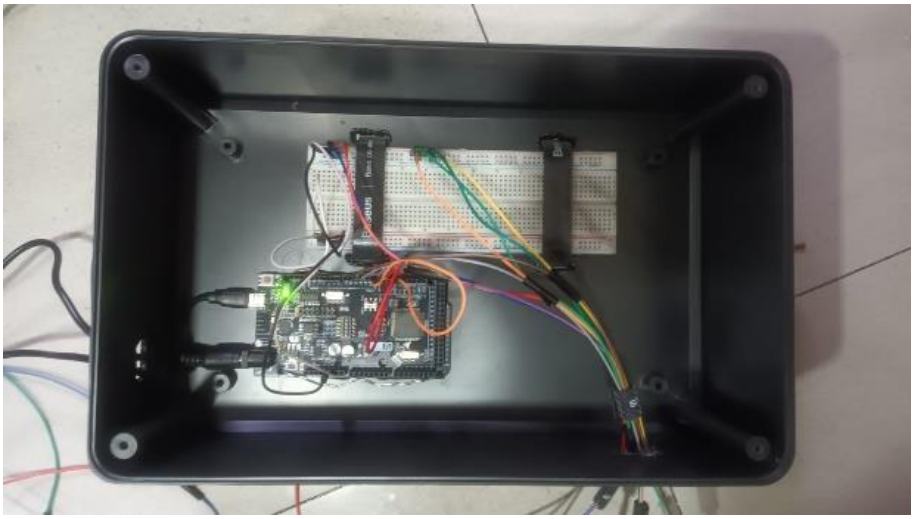
Sensor *infrared* dipasang di seberang kaca penekan, dengan posisi sejajar terhadap penanda. Sensor ini dikoneksikan ke *Arduino* melalui pin 3 untuk mengirimkan sinyal kontrol. Hasil pemasangan sensor *infrared* dapat dilihat pada Gambar 4.3.

Gambar 4.3 Hasil pemasangan sensor *infrared*

Sensor *waterflow* dipasang pada bagian selang, di antara keran dan selang yang mengarah ke area kertas yang digulung. Kedua selang ini berukuran sama, yakni 3/18 inci. Sensor ini dikontrol oleh *Arduino* pada pin 2. Hasil dari pemasangan sensor ini dapat dilihat pada Gambar 4.4.

Gambar 4.4 Hasil sensor *waterflow*

Selain pemasangan ketiga komponen tersebut, *project box* juga perlu dirangkai untuk menyimpan *Arduino* dan *Breadboard* dengan aman. *Project box* diberikan dua lubang, masing-masing untuk adaptor dan kabel sensor serta *servo*. Selain itu, *Arduino* dan *Breadboard* dipasang di bagian dalam *Project Box* yang kemudian diletakkan di dekat sumber listrik yang berada di sisi roda penggulung kiri. Adapun hasil dari pemasangan *project box* ini dapat dilihat pada Gambar 4.5.



Gambar 4.5 Hasil rangkaian *project box*

Berdasarkan hasil pemasangan seluruh rangkaian tersebut, kedua sensor perlu dilakukan kalibrasi maupun evaluasi untuk menyamakan nilai aktual dari keduanya sesuai skenario yang telah dijelaskan pada Bab 3.3.3. Proses pengujian ini diawasi oleh pihak perusahaan untuk memastikan kedua sensor memenuhi kriteria toleransi yang diinginkan.

Hasil kalibrasi menunjukkan bahwa sensor *waterflow* menghasilkan volume aktual 100 mL dalam waktu kurang lebih di 45,7 detik, dan nilai akhir pulsa air per detiknya adalah sebesar 0,219. Nilai tersebut didapatkan melalui kalkulasi persamaan berikut ini.

$$\textbf{kecepatan air} = V \ / \ t$$

$$\textbf{kecepatan air} = 100 \ / \ 45.7$$

$$\textbf{kecepatan air} = 0.219$$

Tidak hanya kalibrasi dan evaluasi sensor *waterflow*, sensor *infrared* juga perlu dikalibrasi melalui persamaan 3.4 yeng telah di bahas sebelumnya. Berdasarkan pengujian selama 1 menit menggunakan persamaan tersebut, ditentukan bahwa perlu menambahkan nilai sebesar 0.3 pada hasil akhirnya. Hal ini dikarenakan nilai aktualnya masih memiliki sebesar 0.3 RPM dari nilai target kecepatan roda, yakni sebesar 50 RPM.

Berdasarkan hasil rangkaian pengumpulan data, yang mencakup pemasangan kedua sensor utama, penanda, serta *project box*, beserta kalibrasi dan evaluasinya dapat dilihat pada Gambar 4.6.

Gambar 4.6 Hasil Rangkaian Pengumpulan Data

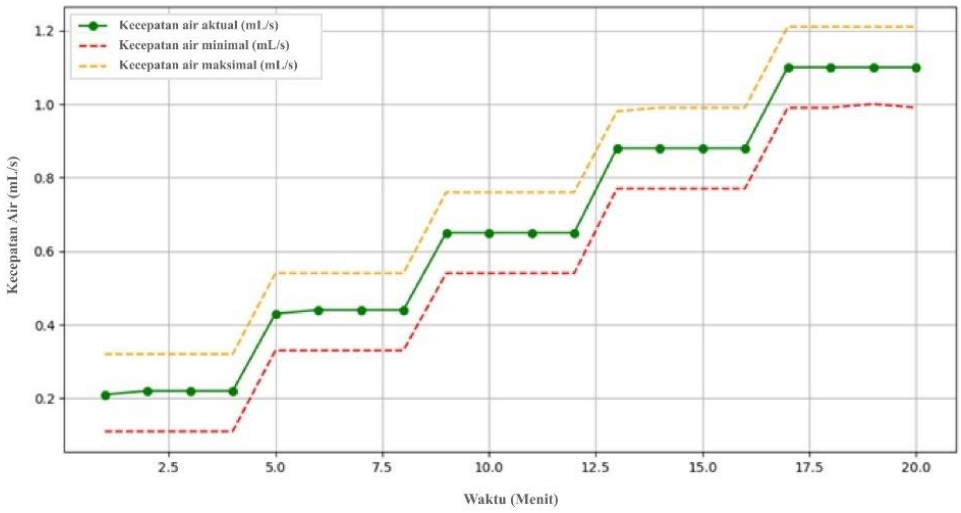
4.1.2 Pengumpulan Data

Pada tahap ini, data dikumpulkan berdasarkan rangkaian yang sebelumnya telah dibuat. Data yang diambil meliputi kecepatan putaran roda, menit ke berapa data diambil, kecepatan putaran roda aktual, dan kecepatan air aktual. Dengan rangkaian yang telah dibuat, pengumpulan data dilakukan tanpa ada masalah dan telah divisualisasiksan melalui Tabel 4.1 berikut.

Tabel 4.1 Hasil pengumpulan data

Menit	Putaran roda (RPM)	Putaran roda aktual (RPM)	Kecepatan aliran aktual (mL/s)
1	50	49.08	0.21
2	50	49.97	0.22
3	50	50.00	0.22
4	50	50.00	0.22
5	50	49.99	0.43
6	50	49.98	0.44
7	50	49.98	0.44
8	50	49.95	0.44
9	50	49.94	0.65
10	50	49.94	0.65
11	50	49.91	0.65
12	50	49.88	0.65
13	50	49.86	0.88
14	50	49.81	0.88
15	50	49.78	0.88
16	50	49.73	0.88
17	50	49.68	1.10
18	50	49.62	1.10
19	50	49.53	1.10
20	50	49.47	1.10

Berdasarkan hasil pengumpulan data ini juga didapatkan data terkait toleransi untuk menentukan berapa minimal dan maksimal cairan yang diperlukan. Grafik pada Gambar 4.7 berikut menunjukkan hasil dari pengumpulan data kecepatan air, beserta hasil toleransinya berdasarkan data yang telah diperoleh.



Gambar 4.7 Grafik hasil pengumpulan data

Berdasarkan Tabel 4.1 dan grafik pada Gambar 4.7, dapat disimpulkan bahwa waktu dan aliran air memiliki hubungan yang searah, sehingga ketebalan rol kertas mempengaruhi jumlah cairan yang dibutuhkan. Jika semakin lama atau rol semakin tebal, maka aliran air juga ikut naik.

4.1.3 Analisis *Setpoint* dan Toleransi

Analisis dilakukan menggunakan persamaan linier untuk menentukan *setpoint* dari kecepatan air pada tiap kecepatan putaran roda dan toleransi yang diizinkan. Analisis menggunakan regresi linier sederhana terhadap hasil pengumpulan data pada Tabel 4.1 dilakukan melalui tahapan berikut ini.

a. Mengelola hasil dari pengumpulan data terlebih dahulu untuk mendapatkan kenaikan yang signifikannya saja. Jika dilihat dari data yang telah dikumpulkan, perubahan signifikan terjadi di tiap 4 menit.
b. Setelah itu, menambahkan aliran air minimal dan maksimal dari tiap 4 menitnya berdasarkan pembahasan besaran toleransi di Bab 4.1.2.

c. Tabel 4.2 di bawah menjadi tabel yang mencangkup hasil dari poin a dan b di tiap 4 menitnya.

Tabel 4.2 Hasil olahan dari pengumpulan data

Menit	Kecepatan Air Aktual (mL/s)	Kecepatan Air Minimal (mL/s)	Kecepatan Air Maksimal (mL/s)
4	0,22	0,11	0,32
8	0,44	0,33	0,54
12	0,65	0,54	0,76
16	0,88	0,77	0,99
20	1,10	0,99	1,21

d. Menentukan jumlah waktu.

$$\sum x = 4 + 8 + 12 + 16 + 20 = 60$$

e. Menentukan jumlah kecepatan air aktual.

$$\sum y = 0.22 + 0.44 + 0.65 + 0.88 + 1.10 = 3.29$$

f. Jumlah kaudrat dari waktu.

$$\sum (x^2) = 16 + 64 + 144 + 256 + 400 = 880$$

g. Jumlah hasil kali waktu dan air Aktual.

$$\sum (xy) = 0.88 + 3.52 + 7.80 + 14.08 + 22.00 = 48.28$$

h. Menentukan nilai kemiringan (m).

$$m = \frac{5(48.28) - (60)(3.29)}{5(880) - (60)^2}$$

$$m = \frac{241.4 - 197.4}{4400 - 3600}$$

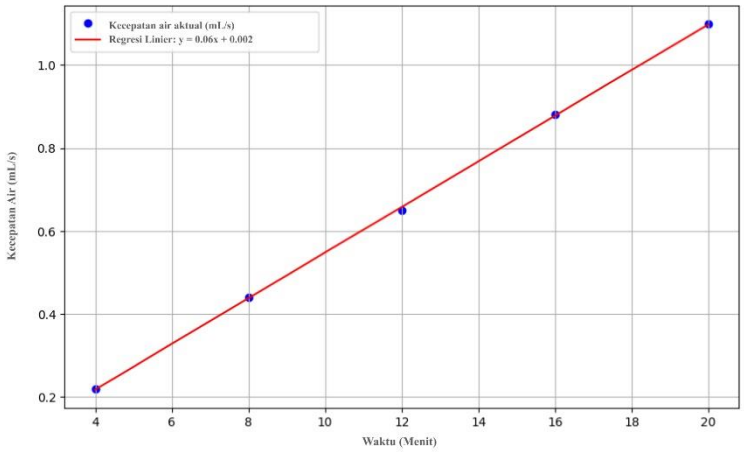
$$m = \frac{44}{800} = 0.055$$

i. Menentukan nilai intersep (b).

$$b = \frac{3.29 - (0.055)(60)}{5}$$

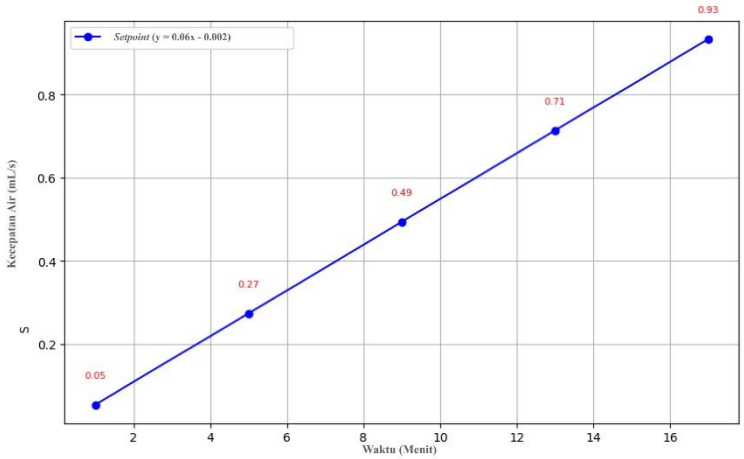
$$b = \frac{3.29 - 3.30}{5} = -0.002$$

Hasil perghitungan regresi linier menghasilkan nilai kemiringan (m) sebesar 0,055 atau jika dibulatkan menjadi 0,06 dengan nilai intersep (c) sebesar 0,002. Tabel berikut ini menyajikan *setpoint* yang ditentukan pada setiap putaran roda. Grafik pada Gambar 4.8 berikut menunjukkan hasil analisis dari kolerasi antara waktu dengan kecepatan air berdasarkan data yang telah diperoleh sebelumnya. Berdasarkan analisis, data yang diproses cukup tiap kelipatan 4 menit.



Gambar 4.8 Hasil persamaan linier

Setelah nilai kemiringan dan intersep telah ditentukan, persamaan tersebut digunakan untuk menentukan *setpoint* dari tiap menit waktu. Gambar 4.9 digunakan untuk memberikan visualisasi terkait *setpoint* dengan kenaikan waktu sebesar 4 menit sesuai dengan analisis terhadap data.



Gambar 4.9 Hasil *setpoint* berdasarkan persamaan linier waktu konstan

Selain melalui kenaikan waktu secara konstan, pengujian ini juga dilakukan berdasarkan nilai acak dari waktu yang ditentukan agar memberikan gambaran terkait perubahan yang terjadi dari waktu ke waktu. Hasil dari *setpoint* berdasarkan nilai acak ini ditampilkan pada Tabel 4.3 di bawah ini.

Tabel 4.3 Tabel hasil *setpoint* berdasarkan waktu

Waktu	***Setpoint* (mL/s)**
16.32	0,9
24,39	1,34
14,12	0,77
8,66	0,47
11,1	0,61

4.1.4 Pengujian Sistem

Setelah *setpoint* pada tiap rentang waktu ditentukan, *tuning* PSO dijalankan berdasarkan desain sistem yang telah dibuat di Bab 3.3 dan dievaluasi di setiap skenarionya. Pengujian akan berhenti apabila telah menemukan hasil optimal berdasarkan evaluasi kinerja PID dengan data yang didapatkan melalui pengumpulan data.

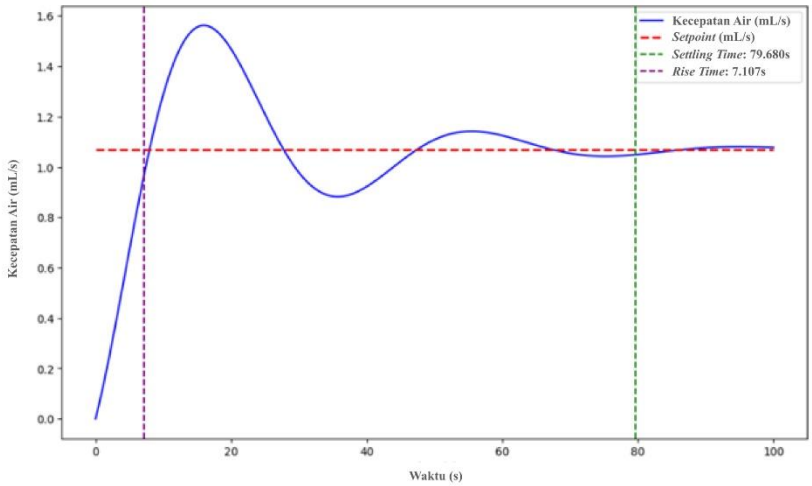
Iterasi skenario awal atau skenario 1 menggunakan parameter umum dari PSO, yakni jumlah populasi partikel berkisar 20 hingga 50, meski pada praktiknya semakin besar maka semakin baik (Piotrowski, A., Napiorkowski, J., & Piotrowska, A., 2020). Selanjutnya, setiap partikel diinisialisasi dengan tiga dimensi yang mewakili nilai-nilai *Kp*, *Ki*, dan *Kd*. Masing-masing rentang nilai pada *Kp*, *Ki*, dan *Kd* adalah 0 hingga 10, 0 hingga 5, dan 0 hingga 5. Ketiga rentang nilai tersebut didapatkan berdasarkan penelitian terdahulu yang memiliki sistem yang sama dengan kebutuhan dari desain sistem yang dirancang pada penelitian ini, yakni memiliki prioritas terhadap respons yang tinggi.

Dengan skenario tersebut, parameter yang digunakan pada skenario 1A dapat dilihat pada Gambar 4.10 di bawah ini yang menghasilkan optimal parameter Kp sebesar 9.7, Ki sebesar 2.7, Kd sebesar 0.11, dan nilai ITAE sebesar 269.2.

Parameter	Value
Swarm Size	50
Max Iterations	100
Lower Bounds (Kp, Ki, Kd)	[9.37068355 1.53304006 0.11543324]
Upper Bounds (Kp, Ki, Kd)	[9.71961124 2.75895103 2.66823747]
w (Inertia)	0.5
c1 (Cognitive)	0.2
c2 (Social)	0.2

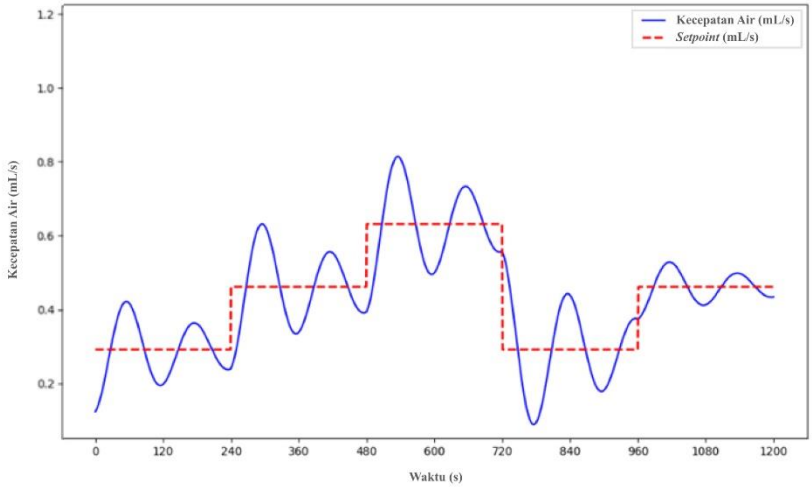
Gambar 4.10 Spesifikasi *tuning* PSO pada skenario 1A

Setelah parameter PID telah dioptimalkan menggunakan *tuning* PSO, pengujian terhadap evaluasi kinerja PID dengan dijalankan. Evaluasi tersebut menghasilkan *overshoot* sebesar 45.0%, *settling time* selama 79,67 detik, *rise time* selama 7,1 detik, dan *steady-state error* sebesar 0.007. Hasil ini terjadi pada evaluasi kinerja PID dengan *setpoint* statis yang dapat dilihat pada Gambar 4.11.



Gambar 4.11 Hasil evaluasi pada skenario 1A *setpoint* statis

Evaluasi kinerja PID pada *setpoint* dinamis dengan skenario ini menghasilkan rata-rata *overshoot* sebesar 94.48%, *settling time* selama 399,33 detik, *rise time* selama 8,34 detik, dan *steady-state error* sebesar 1.31. Hasil ini dapat dilihat pada Gambar 4.12.



Gambar 4.12 Hasil evaluasi pada skenario 1A *setpoint* dinamis

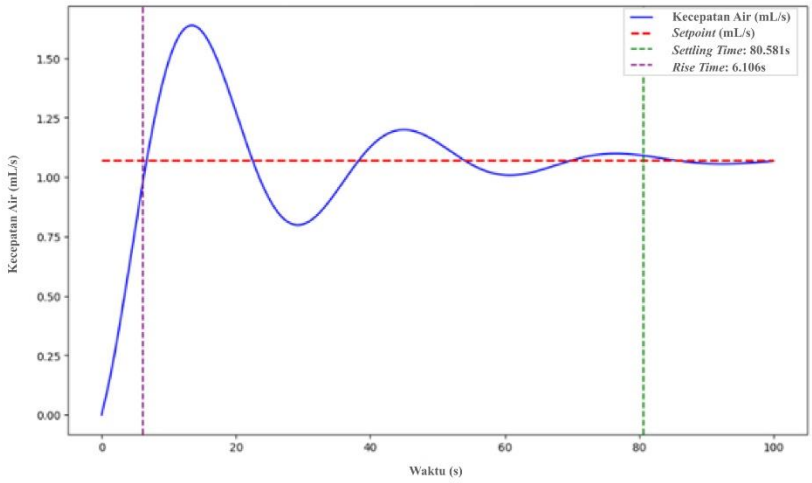
Dengan skenario yang sama, parameter yang digunakan pada skenario 1B dapat dilihat pada Gambar 4.13 di bawah ini yang menghasilkan optimal parameter

Kp sebesar 9.39, Ki sebesar 4.19, dan Kd sebesar 0.62, dan nilai ITAE sebesar 224.67.

Parameter	Value
Swarm Size	50
Max Iterations	100
Lower Bounds (Kp, Ki, Kd)	[9.13432614 1.52863836 0.43386308]
Upper Bounds (Kp, Ki, Kd)	[9.72913946 4.19203947 2.15430418]
w (Inertia)	0.5
c1 (Cognitive)	0.2
c2 (Social)	0.2

Gambar 4.13 Spesifikasi *tuning* PSO pada skenario 1B

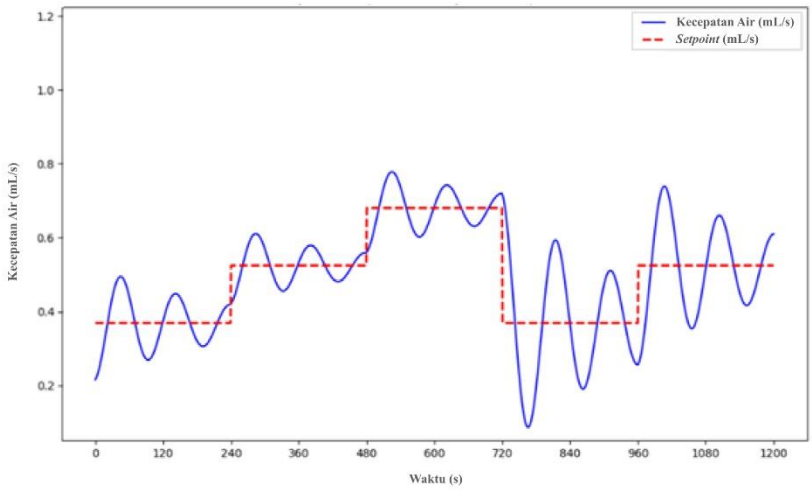
Setelah parameter PID telah dioptimalkan menggunakan *tuning* PSO, pengujian terhadap evaluasi kinerja PID dengan dijalankan. Evaluasi tersebut menghasilkan *overshoot* sebesar 53.14%, *settling time* selama 80,58 detik, *rise time* selama 6,1 detik, dan *steady-state error* yang kurang stabil di 0.98. evaluasi kinerja PID yang dapat dilihat pada Gambar 4.14.



Gambar 4.14 Hasil evaluasi pada skenario 1B *setpoint* statis

Evaluasi kinerja PID pada *setpoint* dinamis dengan skenario ini menghasilkan rata-rata *overshoot* sebesar 83.26%, *settling time* selama 399,3 detik,

rise time selama 70,39 detik, dan *steady-state error* sebesar 0.59. Hasil ini dapat dilihat pada Gambar 4.15.



Gambar 4.15 Hasil evaluasi pada skenario 1B *setpoint* dinamis

Hasil evaluasi kinerja PID pada skenario 1A dan 1B mengalami permasalahan besar. Keduanya mendapatkan hasil yang tidak sesuai dengan sistem kontrol yang akan dibuat. Terlihat pada hasil dengan simulasi *setpoint* statis, kedua skenario mendapatkan kinerja respons dan stabilitas jauh dari 1 detik. Selain itu, nilai *overshoot* cukup besar, meskipun *steady-state error* yang dihasilkan sangat rendah, yakni masing-masing 0.007 dan 0.59.

Pada skenario dengan *setpoint* dinamis, hasil yang diberikan sangat jauh dari *setpoint* untuk semua parameter evaluasi PID. Dengan *settling time* dan *rise time* yang didapatkan lebih dari 1 detik, serta nilai *overshoot* yang hampir mendekati 100%, skenario ini tidak dapat diterapkan pada sistem kontrol yang akan dibuat, meskipun dengan rata-rata *steady-state error* yang rendah.

Berdasarkan penjelasan pada skenario 1A dan 1B permasalahan besar dari keduanya adalah kecepatan respons dan ketidakcocokkan antara parameter satu

dengan yang lainnya sehingga mempengaruhi hasil nilai evaluasi. Hasil skenario 1 ini dapat dirangkum melalui Tabel 4.4 di bawah ini.

Tabel 4.4 Tabel hasil evaluasi skenario 1

Skenario	Kp	Ki	Kd	Hasil Statis	Hasil Dinamis
1A	9,7	2,7	0,11	Nilai Kp terlalu rendah jika dikombinasikan dengan Ki dan Kd	Kp kurang tinggi dan ketiga parameter masih belum seimbang
1B	9,39	4,19	0,62	Nilai Kp terlalu rendah jika dikombinasikan dengan Ki dan Kd	Kp kurang tinggi dan ketiga parameter masih belum seimbang

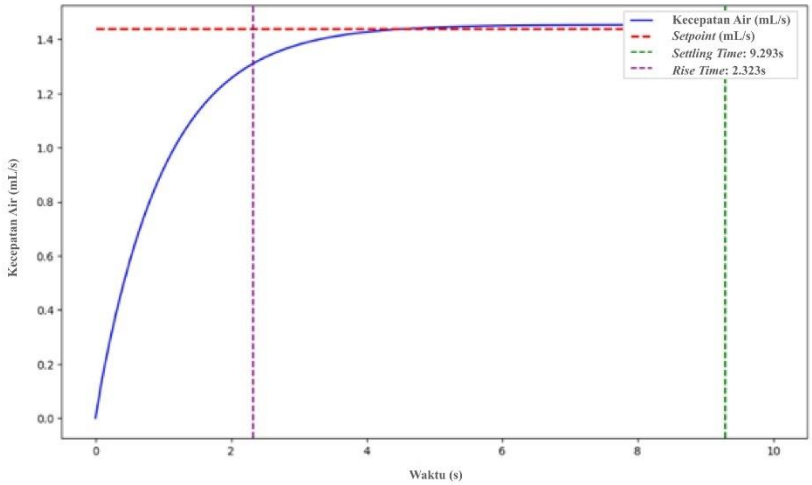
Iterasi skenario kedua atau skenario 2 menggunakan parameter yang disesuaikan berdasarkan skenario sebelumnya, yakni meningkatkan batas bawah Kp, menurunkan serta menyempitkan batas atas Ki dan Kd. Dengan jumlah populasi partikel tetap berkisar 20 hingga 50. Selanjutnya, setiap partikel diinisialisasi dengan tiga dimensi yang mewakili nilai-nilai *Kp, Ki,* dan *Kd*. Masing-masing rentang nilai pada *Kp, Ki,* dan *Kd* adalah 80 hingga 100, 0.1 hingga 1.5, dan 0.1 hingga 1.0.

Dengan skenario tersebut, parameter yang digunakan pada skenario 2A dapat dilihat pada Gambar 4.16 di bawah ini yang menghasilkan optimal parameter Kp sebesar 96.86, Ki sebesar 0.98, Kd sebesar 0.77, dan nilai ITAE sebesar 12.21.

Parameter	Value
Swarm Size	50
Max Iterations	100
Lower Bounds (Kp, Ki, Kd)	[88.46673036 0.97882083 0.43679419]
Upper Bounds (Kp, Ki, Kd)	[98.07263914 1.41427874 0.96191869]
w (Inertia)	0.5
c1 (Cognitive)	0.2
c2 (Social)	0.2

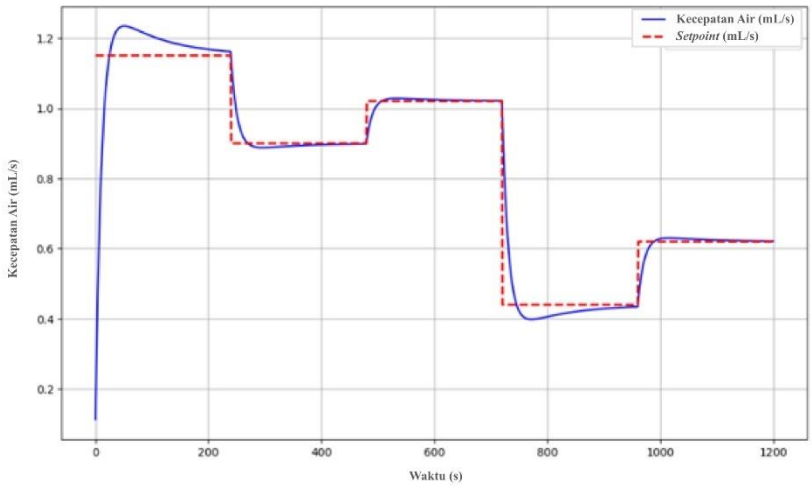
Gambar 4.16 Spesifikasi *tuning* PSO pada skenario 2A

Setelah parameter PID telah dioptimalkan menggunakan *tuning* PSO, pengujian terhadap evaluasi kinerja PID dengan *setpoint* statis dijalankan. Evaluasi tersebut menghasilkan *overshoot* sebesar 0.98%, *settling time* selama 9,20 detik, *rise time* selama 2,32 detik, dan *steady-state error* sebesar 0.01. Hasil ini terjadi pada evaluasi kinerja PID dengan *setpoint* statis, dapat dilihat pada Gambar 4.17.



Gambar 4.17 Hasil evaluasi pada skenario 2A *setpoint* statis

Evaluasi kinerja PID pada *setpoint* dinamis dengan skenario ini menghasilkan rata-rata *overshoot* sebesar 7.20%, *settling time* selama 119,09 detik, *rise time* selama 4,17 detik, dan *steady-state error* sebesar 0.38. Hasil ini dapat dilihat pada Gambar 4.18.



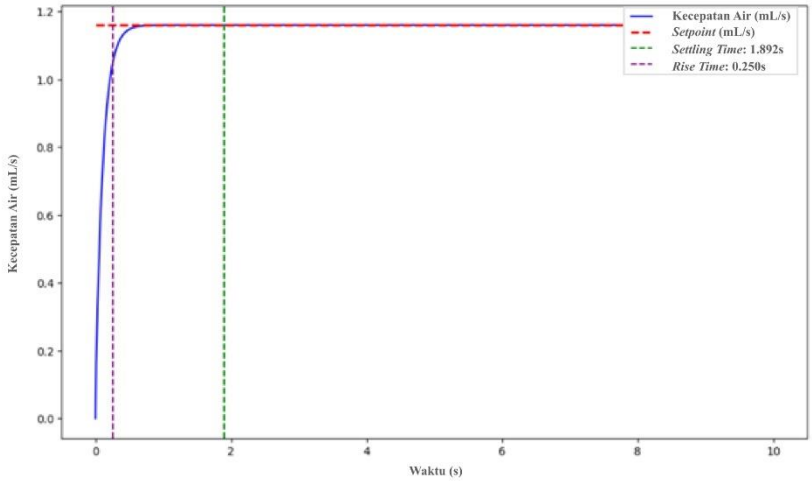
Gambar 4.18 Hasil evaluasi pada skenario 2A *setpoint* dinamis

Dengan skenario yang sama, parameter yang digunakan pada skenario 2B dapat dilihat pada Gambar 4.19 di bawah ini yang menghasilkan optimal parameter Kp sebesar 90.74, Ki sebesar 0.16, Kd sebesar 0.49, dan nilai ITAE sebesar 1.56.

Parameter	Value
Swarm Size	50
Max Iterations	100
Lower Bounds (Kp, Ki, Kd)	[82.20020529 0.1624574 0.23927951]
Upper Bounds (Kp, Ki, Kd)	[91.98288937 1.45632187 0.62897324]
w (Inertia)	0.5
c1 (Cognitive)	0.2
c2 (Social)	0.2

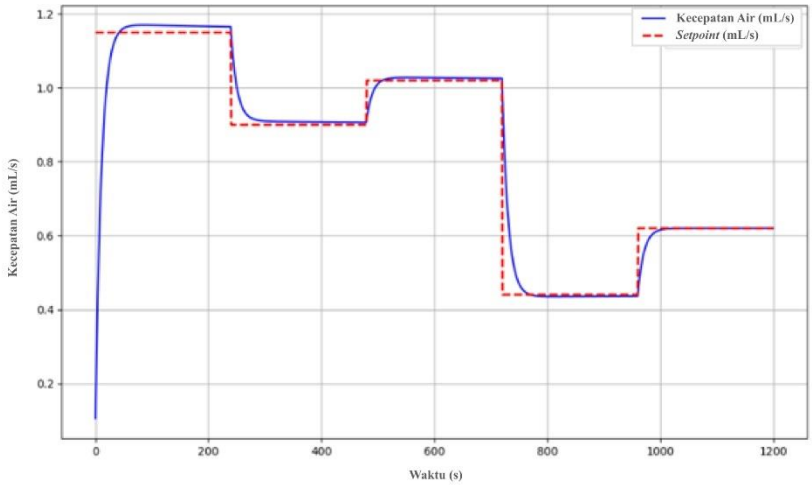
Gambar 4.19 Spesifikasi *tuning* PSO pada skenario 2B

Setelah parameter PID telah dioptimalkan menggunakan *tuning* PSO, pengujian terhadap evaluasi kinerja PID dengan dijalankan. Evaluasi tersebut menghasilkan *overshoot* sebesar 0.02%, *settling time* selama 1,89 detik, *rise time* selama 0,25 detik, dan *steady-state error* sebesar 0.0002. evaluasi kinerja PID yang dapat dilihat pada Gambar 4.20.



Gambar 4.20 Hasil evaluasi pada skenario 2B *setpoint* statis

Evaluasi kinerja PID pada *setpoint* dinamis dengan skenario ini menghasilkan rata-rata *overshoot* sebesar 6.57%, *settling time* selama 110,59 detik, *rise time* selama 4,83 detik, dan *steady-state error* sebesar 0.37. Hasil ini dapat dilihat pada Gambar 4.21.



Gambar 4.21 Hasil evaluasi pada skenario 2B *setpoint* dinamis

Hasil evaluasi kinerja PID pada skenario 2A dan 2B mengalami peningkatan signifikan di hampir semua bagian. Pada skenario 2A dengan *setpoint*

statis, mendapatkan nilai *overshoot* yang lebih kecil dan memiliki *settling time* serta *rise time* yang lebih cepat dibandingkan dengan skenario 1, meskipun hasulnya masih kurang sesuai dengan sistem kontrol yang diharapkan. Selain itu, *steady-state error* yang dihasilkan juga lebih kecil dari sebelumnya. Nilai ini juga bertahan pada pengujian dengan *setpoint* dinamis, meskipun rata-rata *settling time* nya masih besar.

Pada skenario 2B dengan *setpoint* statis, hasil yang diberikan sangat bagus, yakni *settling time* di bawah 2 detik dan *rise time* di bawah 1 detik serta menghasilkan *steady state error* yang stabil meskipun tidak lebih baik dari skenario 2A. Selain itu, *overshoot* yang diberikan sangat rendah. Hasil ini juga tetap berlaku pada *setpoint* dinamis, meskipun rata-rata *settling time* nya masih besar. Berdasarkan penjelasan pada skenario 2A dan 2B dapat dirangkum melalui melalui Tabel 4.5 di bawah ini.

Tabel 4.5 Tabel hasil evaluasi skenario 2

Skenario	**Kp**	**Ki**	**Kd**	**Hasil Statis**	**Hasil Dinamis**
2A	96,86	0,98	0,77	Hasilnya sudah sesuai	Hasilnya sudah sesuai
2B	90,74	0,16	0,49	Hasilnya sudah sesuai	Hasilnya sudah sesuai

Setelah semua skenario telah dievaluasi, peneliti perlu untuk mengumpulkan beberapa informasi terkait hasil dari skenario. Tabel 4.6 di bawah ini akan menunjukkan variabel PSO yang dipakai pada masing-masing skenario, termasuk juga batas atas dan bawah dari nilai acak yang digunakan pada sistem yang dibuat.

Tabel 4.6 Evaluasi kinerja PID dengan *tuning* PSO dan *setpoint* statis

Skenario	***Lower Bounds*** **(Kp, Ki, Kd)**	***Upper Bounds*** **(Kp, Ki, Kd)**	**Inersia**	**C1**	**C2**
1A	9,37, 1.53, 0.11	9.71, 2.75, 2.66	0.5	0.2	0.2
1B	9.13, 1.52, 0.43	9.72, 4.19, 0.5	0.5	0.2	0.2
2A	88.46, 0.97, 0.43	98.07, 1.41, 0.96	0.5	0.2	0.2
2B	82,2, 0.16, 0.23	91.98, 1.45, 0.62	0.5	0.2	0.2

Berdasarkan varibel PSO yang telah dijabarkan di setiap skenario, variable-variabel tersebut akan menjadi parameter dari *tuning* PSO yang dapat menghasilkan nilai optimal dari parameter PID. Hasil dari parameter optimal tersebut dievaluasi kinerjanya dan menghasilkan pilihan terbaik di antara parameter optimal yang telah didapatkan. Hasil evaluasi kinerja PID tersebut dapat dilihat pada Tabel 4.7.

Tabel 4.7 Evaluasi kinerja PID dengan tuning PSO

Skenario	**Kp**	**Ki**	**Kd**	**Rise Time (s)**	**Settling Time (s)**	**Overshoot (%)**	**SSE (%)**	**ITAE**
1A	9,70	2,70	0,11	7,1	79,67	45	0,007	269,2
1B	9,39	4,19	0,62	6,1	80,58	53,14	0.98	224,67
2A	96,8	0,98	0,77	0,3	9.2	0.98	0.01	12,21
2B	90.7	0,16	0,49	0,25	1,89	0.02	0.00002	1,56

Berdasarkan tabel evaluasi kinerja PID dengan *tuning* PSO, skenario 2B menjadi nilai paling optimal dibandingkan 3 skenario lainnya. Penilaian ini didasari oleh nilai ITAE yang lebih kecil dan seluruh parameternya yang lebih baik dari skenario lain.

4.1.5 Hasil Pengujian

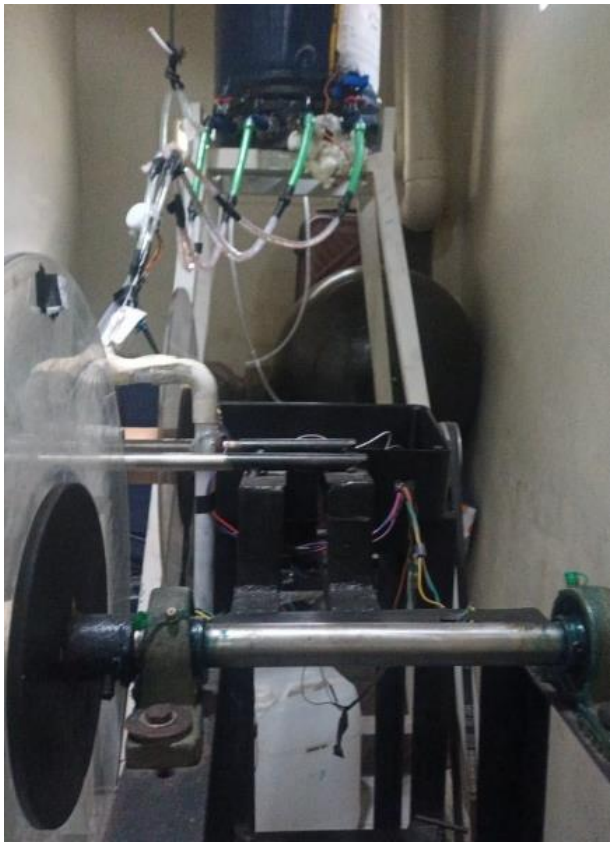
Setelah sistem dengan performa optimal ditentukan melalui evaluasi kinerja PID dengan *tuning* PSO, langkah selanjutnya adalah melakukan pengujian

akhir. PID yang telah dioptimalkan akan menjadi parameter yang akan dijalankan oleh *Arduino* untuk menentukan sudut bukaan keran yang tepat.

Rangkaian pengujiannya hampir sama dengan rangkaian dari pengumpulan data. Perbedaannya hanya terletak pada *Arduino* yang telah diintegrasikan PID dengan nilai yang telah dioptimasi dan aktuator motor *servo* yang dipasang pada keran. Hasil pemasangan keran ini dapat dilihat pada Gambar 4.22.

Gambar 4.22 Hasil pemasangan servo pada keran

Hasil rangkaian secara keseluruhan alat dan mesinnya dapat dilihat pada Gambar 4.23 di bawah ini. Rangkaian keseluruhan ini, mirip dengan desain rangkaian yang telah dibahas di Bab 4.1.1. Dengan *servo* yang telah dipasang, sensor-sensor yang sudah disesuaikan, dan penanda yang sudah ditempelkan pada kaca pelindung roda.



Gambar 4.23 Tampilan mesin dan hasil rangkaian akhir

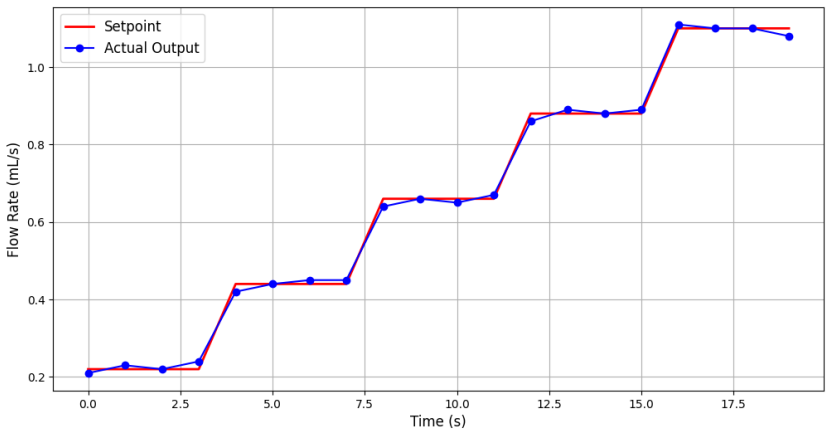
Pengujian ini diobservasi langsung oleh pihak perusahaan dalam kondisi nyata dari mesin pelapis kertas. Kolom hasil dari pengujian ini dibagi menjadi kriteria sesuai dengan yang tertulis pada Bab 3.6, yakni sesuai, toleransi, dan gagal. Pengujian ini menghasilkan Tabel 4.8 berikut.

Tabel 4.8 Pengujian desain sistem

Menit	Setpoint (mL/s)	Roda aktual (RPM)	Kecepatan air aktual (mL/s)	Hasil
1	0.22	49.6	0.21	Sesuai
2	0.22	50	0.23	Sesuai
3	0.22	50	0.22	Sesuai
4	0.22	49.7	0.24	Toleransi
5	0.44	49.7	0.42	Sesuai
6	0.44	49.8	0.44	Sesuai
7	0.44	50	0.45	Sesuai
8	0.44	50	0.45	Sesuai
9	0.66	49.8	0.64	Toleransi

Menit	Setpoint (mL/s)	Roda aktual (RPM)	Kecepatan air aktual (mL/s)	Hasil
10	0.66	50	0.66	Sesuai
11	0.66	50	0.65	Sesuai
12	0.66	50	0.67	Toleransi
13	0.88	50	0.86	Toleransi
14	0.88	49.9	0.89	Toleransi
15	0.88	49.9	0.88	Sesuai
16	0.88	50	0.89	Sesuai
17	1.10	50	1.11	Sesuai
18	1.10	50	1.10	Sesuai
19	1.10	50	1.10	Sesuai
20	1.10	49.9	1.08	Toleransi

Hasil ini merupakan rata-rata dari keseluruhan data di tiap menit. Grafik pada Gambar 4.24 menunjukkan fluktuasi kecepatan air aktual dibandingkan dengan *setpoint* pada tiap interval waktu di tiap menit menggunakan skenario 2B dari PID dan *tuning* PSO yang telah dianalisis.



Gambar 4.24 Grafik hasil pengujian akhir

Adapun hasil dari implementasi sistem dengan *Dashboard Blynk* dapat dilihat pada Gambar 4.25. Terintegrasinya sistem dengan *Cloud* seperti *Blynk* akan mempermudah penggunaan sistem kontrol yang telah dibuat karena dapat diakses kapanpun dan dimanapun.

Gambar 4.25 Hasil tampilan *Dashboard Blynk*

4.2 Pembahasan

Berdasarkan hasil yang diperoleh dari pengujian sistem pengontrol PID yang dioptimalkan dengan metode *tuning* PSO, dapat disimpulkan bahwa sistem ini berhasil memenuhi semua permintaan dan spesifikasi yang telah ditetapkan sebelumnya. Hal ini terbukti melalui evaluasi yang dilakukan selama tahap pengumpulan data hingga pengujian akhir. Meskipun memerlukan waktu yang cukup lama untuk melakukan rangkaian pengujian tersebut, sistem mampu menjaga kinerjanya secara konsisten dan efektif sepanjang proses.

Hasil dari skenario pertama (skenario 1A) dengan hasil optimal Kp sebesar 9.70, Ki sebesar 2.70, dan Kd sebesar 0.11 menghasilkan *rise time* 7,1 detik, *settling time* selama 79,67 detik, *overshoot* sebesar 45%, *steady-state error* sebesar 0,0007% dan hasil ITAE nya sebesar 269,2. Pada skenario kedua (skenario 1B)

dengan hasil optimal Kp sebesar 9.39, Ki sebesar 4.19, dan Kd sebesar 0.62 menghasilkan *rise time* 6,1 detik, *settling time* selama 80,58 detik, *overshoot* sebesar 53,14%, *steady-state error* sebesar 0,98% dan hasil ITAE nya sebesar 224,67. Pada skenario ketiga (skenario 2A) dengan hasil optimal Kp sebesar 96.86, Ki sebesar 0.98, dan Kd sebesar 0.77 menghasilkan *rise time* 0,3 detik, *settling time* selama 9,2 detik, *overshoot* sebesar 0,98%, *steady-state error* sebesar 0,001% dan hasil ITAE nya sebesar 12,21. Pada skenario keempat (2B) dengan hasil optimal Kp sebesar 9.57, Ki sebesar 0.17, dan Kd sebesar 0.17 menghasilkan *rise time* 0,3 detik, *settling time* selama 0,5 detik, *overshoot* sebesar 0,37%, *steady-state error* sebesar 0,02% dan hasil ITAE nya sebesar 0,3.

Melalui tabel evaluasi kinerja PID dengan *tuning* PSO tersebut, skenario 2ABdengan nilai optimal Kp sebesar 90.74, Ki sebesar 0.16, dan Kd sebesar 0.49 menjadi nilai paling optimal dibandingkan 3 skenario lainnya. Penilaian ini didasari oleh nilai ITAE yang lebih kecil, yakni 1,56 dan seluruh parameter dari skenario 2B lebih baik dari yang lain. Parameter dari skenario 2B ini menghasilkan *rise time* dalam waktu 0,25 detik, *settling time* selama 1,89 detik, *overshoot* sebesar 0,02%, dan *steady-state error* sebesasr 0,00002%.

Pengujian ini mengonfirmasi bahwa sistem pada skenario 2B dengan menggunakan metode PSO untuk *tuning* parameter PID dapat memberikan performa yang stabil. Dengan mengoptimalkan parameter *rise time*, *settling time*, *overshoot*, dan *steady-state error*, sistem ini mampu menghasilkan respons yang responsif dan stabil berdasarkan yang tertera pada Tabel 4.8 dan Gambar 4.21 menunjukkan bahwa setiap pengujian akhir mengindikasikan bahwa sistem berhasil

mempertahankan *output* sesuai dengan nilai *setpoint* yang diberikan. Ini menunjukkan bahwa hampir tidak ada kegagalan atau penyimpangan signifikan selama pengujian. Hasil dari pengujian akhir ini menghasilkan kesesuaian sebesar 85.71%, pada ambang batas minimal ataupun maksimal sebesar 10.16%, dan mengalami kegagalan sebesar 4.13% dalam waktu kurang lebih 20 menit.

Secara keseluruhan, desain sistem yang menggunakan pengontrol PID yang dioptimalkan oleh PSO ini telah terbukti efektif dalam menjaga kestabilan sistem operasional, meskipun terdapat variasi pada *setpoint* yang diberikan seiring dengan waktu. Semua hasil pengujian mengindikasikan bahwa nilai *output* tidak melebihi batas atas atau bawah secara signifikan, yang menandakan bahwa sistem sangat responsif dan adaptif terhadap perubahan *setpoint* dalam lingkungan operasional yang dinamis.

Meskipun, terdapat beberapa kendala yang perlu diperhatikan terutama pada rangkaian fisik dari sistem. Mulai dari daya *servo* yang dibutuhkan, maupun *source code* yang digunakan oleh *Arduino IDE* yang perlu diringkas performanya. Karena pada penelitian ini, terjadi pembengkakan performa pada *Arduino IDE* sehingga mempengaruhi hasil dari besaran bukaan dari motor *servo*.

Selain itu, simulasi dari pengujian sistem untuk nilai *fitness* dengan ITAE beserta evaluasi kinerja PID yang telah dibuat tidak menghasilkan nilai yang 100% akurat, dan tidak serta merta dapat diterapkan pada sistem kontrol yang berbeda bahkan untuk yang sejenis. Jadi, perlu dilakukan konfigurasi ulang.

Dengan demikian, penerapan metode PSO dalam optimasi parameter PID telah berhasil menciptakan sebuah sistem yang dapat mempertahankan stabilitas

operasional yang tinggi, memastikan performa yang optimal sepanjang waktu, dan sangat cocok untuk aplikasi pelapisan kertas yang membutuhkan ketepatan aliran cairan sesuai dengan setpoint yang diinginkan.

4.3 Integrasi Islam

Integrasi Islam mencakup harmoni antara *hablumminallah* (hubungan manusia dengan Allah), *hablumminannas* (hubungan manusia dengan sesama), dan hubungan manusia dengan lingkungan sebagai manifestasi dari peran manusia sebagai khalifah di bumi. Ketiga hubungan ini saling melengkapi dan membutuhkan. Adapun integrasi islam yang diimplementasikan pada penelitian ini dapat dijabarkan pada tiga sub bab di bawah ini.

4.3.1 *Hablumminallah*

Penerapan teknologi efisiensi pada mesin pelapis kertas tidak hanya mencerminkan ketaatan pada prinsip efisiensi dalam Islam, tetapi juga secara langsung mendukung manusia dalam memfokuskan waktu dan energi untuk beribadah kepada Allah SWT. Teknologi ini membantu mengurangi beban kerja manual yang memakan waktu dan tenaga, sehingga memberi ruang lebih bagi pengguna untuk fokus melaksanakan ibadah, mendekatkan diri kepada Allah, dan memperbanyak amal saleh. Allah SWT berfirman dalam surat *Adz-Dzariyat* Ayat 56:

وَمَا خَلَقْتُ ٱلْجِنَّ وَٱلْإِنسَ إِلَّا لِيَعْبُدُونِ

"Dan Aku tidak menciptakan jin dan manusia melainkan supaya mereka beribadah kepada-Ku." (*QS.\ Adz-Dzariyat: 56)*

Tafsir *Hidayatul Insan bi Tafsiril Qur'an* menjelaskan bahwa Allah menciptakan jin dan manusia untuk beribadah kepada-Nya, yang mencakup mengenal, mencintai, dan kembali kepada-Nya sambil berpaling dari selain-Nya. Kesempurnaan ibadah tergantung pada pengenalan (ma'rifat) kepada Allah; semakin seseorang mengenal-Nya, semakin sempurna ibadahnya. Allah tidak membutuhkan makhluk-Nya, tetapi makhluk-Nya selalu membutuhkan-Nya dalam segala hal.

Ayat ini mengingatkan bahwa tujuan utama penciptaan manusia adalah untuk beribadah. Dengan menggunakan teknologi yang mengoptimalkan waktu dan tenaga, manusia dapat lebih leluasa memenuhi tujuan ini tanpa terganggu oleh tugas-tugas yang sifatnya repetitif atau membebani.

Selain itu, teknologi ini juga membantu manusia memanfaatkan waktu secara efektif, yang merupakan bagian dari syukur atas nikmat waktu yang diberikan Allah. Nabi s*hallallahu 'alaihi wa sallam* bersabda,

نِعْمَتَانِ مَغْبُونٌ فِيهِمَا كَثِيرٌ مِنَ النَّاسِ ، الصِّحَّةُ وَالْفَرَاغُ

"*Ada dua kenikmatan yang banyak manusia tertipu, yaitu nikmat sehat dan waktu senggang".* (*HR. Bukhari no. 6412, dari Ibnu 'Abbas*)

Hadits tersebut berkaitan dengan Ibnu Qayyim dalam sebuah tulisannya, beliau berkata, "Pemikiran yang paling cemerlang dan mempunyai kedudukan tinggi di sisi Allah di akhirat bermacam-macam. Di antaranya, ialah berpikir bagaimana mengisi dan menggunakan waktu seefisien mungkin dan mencurahkan segala perhatian untuk mengukur waktu. Orang yang paham adalah orang yang tahu

akan nilai waktu, apabila ia menyia-nyiakannya berarti ia telah menyia-nyiakan kemaslahatan hidupnya. Sebab, setiap kemaslahatan dan keberhasilan diperoleh karena penggunaan waktu yang efisien, dan waktu yang telah berlalu mustahil untuk diulang kembali" (Ahmad Ubaidillah, 2013).

Hadits ini mengingatkan bahwa waktu senggang adalah nikmat besar yang sering disia-siakan. Dengan teknologi optimasi, waktu yang sebelumnya dihabiskan untuk pengawasan manual dapat dimanfaatkan untuk hal-hal yang lebih bermanfaat, seperti shalat sunnah, membaca Al-Qur'an, atau memperbanyak dzikir.

Teknologi adalah sarana ikhtiar untuk mempermudah kehidupan. Setelah ikhtiar dilakukan melalui optimasi teknologi, manusia dapat lebih tenang menyerahkan hasilnya kepada Allah, sehingga mampu lebih khusyuk dalam menjalankan ibadah.

Keseluruhan penerapan teknologi ini menunjukkan bahwa efisiensi bukan hanya soal kemajuan teknis, tetapi juga bagian dari upaya manusia untuk mengatur kehidupan yang selaras dengan prinsip Islam. Dengan teknologi yang mendukung efisiensi, manusia tidak hanya melaksanakan tugas duniawi dengan lebih baik, tetapi juga memperkuat hubungan dengan Allah melalui peningkatan kualitas ibadah dan pemanfaatan waktu yang lebih bijak.

4.3.2 *Hablumminannas*

Hablumminannas, atau hubungan manusia dengan manusia lainnya, tercermin dalam penerapan sistem optimasi pada mesin pelapis kertas yang memberikan manfaat langsung bagi kehidupan sehari-hari. Sistem ini dirancang untuk mengurangi beban kerja manual, yang sebelumnya memerlukan pengawasan

terus-menerus. Dengan otomatisasi, pekerja dapat lebih fokus pada tugas lain atau menikmati waktu istirahat, sehingga meningkatkan kesejahteraan mereka. Hal ini sejalan dengan sabda Rasulullah SAW dalam hadits yang diriwayatkan oleh Imam Muslim dari Abu Hurairah:

مَنْ نَفَّسَ عَنْ مُؤْمِنٍ كُرْبَةً مِنْ كُرَبِ الدُّنْيَا نَفَّسَ اللَّهُ عَنْهُ كُرْبَةً مِنْ كُرَبِ يَوْمِ الْقِيَامَةِ وَمَنْ يَسَّرَ عَلَى مُعْسِرٍ يَسَّرَ اللَّهُ عَلَيْهِ فِي

"Siapa yang menyelesaikan kesulitan seorang mukmin dari berbagai kesulitan dunia, Allah akan memudahkan urusan-urusan kesulitannya di akhirat. Siapa yang mempermudah orang yang dalam kesulitan, Allah akan mempermudah urusan dunia dan akhirat baginya." (HR. Muslim, no. 2699)

Tafsir pada kitab *al-Arba'in al-Nawawiyah* oleh An-Nawawi menegaskan pentingnya saling membantu dalam kebaikan tanpa memandang status, pangkat, atau derajat seseorang, dan tanpa mengharapkan imbalan apa pun (Muhammad Rusmin B., & Abduzar Al Qifari., 2023).

Penerapan teknologi ini juga menunjukkan bahwa ilmu yang diberikan Allah SWT kepada manusia harus digunakan untuk kebaikan bersama. Sebagaimana firman Allah dalam QS. *Al-Alaq* ayat 4-5:

الَّذِيْ عَلَّمَ بِالْقَلَمِۙ عَلَّمَ الْاِنْسَانَ مَا لَمْ يَعْلَمْۗ

"Yang mengajar (manusia) dengan kalam. Dia mengajarkan manusia apa yang tidak diketahuinya" (QS. Al-Alaq : 4-5).

Menurut tafsir *al-Muyassar*, ayat ini menegaskan bahwa Allah memberikan kemampuan kepada manusia untuk menulis, membaca, dan memahami ilmu, sebagai nikmat yang harus dimanfaatkan sebaik-baiknya (Muji, 2020). Dengan memanfaatkan ilmu untuk kebaikan, seperti peningkatan

kesejahteraan manusia dengan teknologi, memberikan kemaslahatan yang baik untuk ruang yang lebih luas.

4.3.3 Hubungan manusia dengan lingkungan

Dalam Islam, manusia dipandang sebagai khalifah atau pemimpin di bumi yang memiliki tanggung jawab untuk menjaga dan memelihara alam serta segala isinya. Hal ini tercermin dalam ajaran Islam yang mengingatkan umatnya untuk tidak merusak atau mencemari lingkungan. Salah satu ajaran utama yang mendasari hubungan ini terdapat dalam QS. *Al-Baqarah* ayat 205:

وَاللَّهُ لَا يُحِبُّ الْفَسَادَ

"Dan Allah tidak menyukai kerusakan." (QS. Al-Baqarah: 205).

Ahmad Mustafa al-Maraghi dalam tafsirnya menekankan bahwa manusia harus menghindari perilaku yang merusak lingkungan, baik secara langsung maupun tidak langsung, karena tindakan tersebut merugikan manusia dan alam.

Dalam konteks teknologi yang diterapkan pada sistem pengontrol cairan, efisiensi yang dihasilkan tidak hanya berfokus pada pengurangan biaya dan peningkatan produktivitas perusahaan, tetapi juga pada pengurangan dampak negatif terhadap lingkungan. Penggunaan cairan yang tepat tanpa pemborosan mencegah terjadinya limbah yang dapat mencemari lingkungan, yang sejalan dengan ajaran Islam untuk tidak melakukan kerusakan di bumi.

Konsep ini juga selaras dengan prinsip Islam tentang keseimbangan (*wasatiyyah*) yang mengajarkan untuk tidak berlebihan (*israf*) dan tidak kikir (*taqtir*), sebagaimana dijelaskan dalam QS. Al-Furqan ayat 67:

وَٱلَّذِينَ إِذَآ أَنفَقُواْ لَمۡ يُسۡرِفُواْ وَلَمۡ يَقۡتُرُواْ وَكَانَ بَيۡنَ ذَٰلِكَ قَوَامٗا

*"*Dan orang-orang yang apabila membelanjakan (harta), mereka tidak berlebihan, dan tidak (pula) kikir, dan adalah (pembelanjaan itu) di tengah-tengah antara yang demikian*." (QS. Al-Furqan: 68).*

Menurut Tafsir *Al-Wajiz* oleh Syaikh Prof. Dr. Wahbah az-Zuhaili pada kajian surat *Al-Furqan* ayat 67 memaparkan, "Dan orang-orang yang apabila membelanjakan harta, mereka tidak berlaku boros dan tidak pelit. Pembelanjaan mereka itu sedang-sedang saja, tidak lebih dan tidak kurang". Hal ini tercermin dalam sistem kontrol cairan yang mengoptimalkan penggunaan cairan pada mesin, memastikan bahwa tidak ada pemborosan yang terjadi. (Mansour Abu Zaina, 2013).

Sistem ini juga dapat dilihat sebagai bentuk nyata dari pemanfaatan ilmu pengetahuan dalam menjaga kelestarian lingkungan, yang sejalan dengan prinsip Islam tentang pentingnya ilmu dan teknologi dalam kehidupan manusia yang tercermin pada firman Allah SWT yang sebelumnya dibahas, yakni QS. *Al-Alaq* ayat 4-5.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil pengumpulan data, analisis, dan pengujian yang telah dilakukan, dapat disimpulkan bahwa metode optimasi yang diterapkan berhasil meningkatkan performa sistem kontrol pada mesin pelapis kertas sesuai dengan kebutuhan operasional perusahaan. Penelitian ini membuktikan bahwa penerapan metode PSO secara signifikan meningkatkan efisiensi dan efektivitas sistem kontrol yang menggunakan PID. Hasil pengujian menunjukkan bahwa:

1. Skenario terbaik adalah Skenario 2B dengan nilai parameter PID optimal Kp = 90.74, Ki = 0.16, dan Kd = 0.49.
    - Nilai ITAE terkecil sebesar 1.56, *rise time* 0.25 detik, *settling time* 1.89 detik, *overshoot* 0.02%, dan *steady-state error* 0.00002% menunjukkan kinerja dinamik sistem yang optimal.
    - Hasil dari pengujian akhir ini adalah kesesuaian sebesar 85.71%, ambang batas sebesar 10.16%, dan mengalami kegagalan sebesar 4.13% dalam waktu kurang lebih 20 menit.
2. Dampak operasional:
    - Sistem yang sebelumnya berjalan secara manual kini telah terotomasi, membutuhkan pengawasan minimal pada mesin, sehingga meringankan tugas pegawai.
    - Tingkat kesalahan pada proses pengendalian cairan berkurang meskipun ada sedikit inkonsistensi tidak signifikan dibandingkan

metode manual. Namun, hasil tetap berada dalam batas wajar dari perusahaan dan tidak menyebabkan kerugian pada bahan baku.

- Kualitas tetap terjaga dengan jumlah produk yang berhasil relatif stabil, tanpa kerusakan kertas atau cairan meluber, meskipun terjadi kesalahan kecil dalam ambang batas.

Penelitian ini memberikan kontribusi nyata terhadap proses produksi yang lebih efisien, konsisten, dan berkualitas tinggi. Dengan sistem yang telah dioptimasi, perusahaan berhasil mendukung tujuan pengurangan beban kerja pegawai sekaligus memaksimalkan sumber daya yang tersedia. Hasil penelitian ini juga dapat dijadikan referensi untuk implementasi metode optimasi serupa pada mesin industri lainnya yang membutuhkan efisiensi dan konsistensi tinggi.

5.2 Saran

Untuk pengembangan lebih lanjut, disarankan agar ukuran sensor *waterflow* dipilih sesuai dengan kebutuhan spesifik untuk memastikan akurasi pembacaan. Selain itu, pemasangan aktuator pada keran perlu dilakukan dengan lebih cermat, mengingat akutasi posisi dapat memengaruhi putaran. Implementasi kode pada PID juga perlu dirancang seefisien mungkin dapat diproses dengan baik oleh *Arduino*. Terakhir, parameter dalam metode PSO sebaiknya divariasikan sesuai kebutuhan untuk menghasilkan performa yang optimal di berbagai kondisi operasional.

DAFTAR PUSTAKA

Atina, V. Z., Mahmudi, A. Y., & Abdillah, H. (2020). Ceper Foundry Industries, Technology Management Readiness for Industrial Revolution 4.0. In *Brawijaya International Conference on Multidisciplinary Sciences and Technology* (Vol. 1, No. 2020, pp. 14-17).

Wahyuningsih, S. (2019). Pengaruh pelatihan dalam meningkatkan produktivitas kerja karyawan. *Warta Dharmawangsa*, *13*(2).

Prasetyo, A., Abdillah, H., Wawan, W., Nurtanto, M., Kholifah, N., & Suyitno, S. (2021, April). How to the Need for Personal Protective Equipment (PPE) during the current Covid 19 Pandemic: Smart Products Solution. In *2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT)* (pp. 309-313). IEEE.

Kalsoom, T., Ramzan, N., Ahmed, S., & Ur-Rehman, M. (2020). Advances in sensor technologies in the era of smart factory and industry 4.0. *Sensors*, *20*(23), 6783.

Anaam, I. K., Hidayat, T., Pranata, R. Y., Abdillah, H., & Putra, A. Y. W. (2022, June). Pengaruh trend otomasi dalam dunia manufaktur dan industri. In *Vocational Education National Seminar (VENS)* (Vol. 1, No. 1).

Mustafa, N., & Hashim, F. H. (2020). Design of a Predictive PID *Controller* Using Particle Swarm Optimization. *International Journal of Electronics and Telecommunications*, 737–743. https://doi.org/10.24425/ijet.2020.134035

YUMURTACI, M., & VERİM, Ö. (2020). Liquid level control with different control methods based on Matlab/Simulink and Arduino for the control systems lesson. *International Advanced Researches and Engineering Journal*, *4*(3), 249–254. https://doi.org/10.35860/iarej.750664

Agil Haikal, M., Tulus Herlambang, D., & Ali, M. (2021). *Desain Optimasi PID Controller Pada Heating Furnace Temperature Menggunakan Metode Particle Swarm Optimization (PSO)* (Vol. 2). www.elektro.itn.ac.id

Argo, B. D., Hendrawan, Y., Firmanda, D., Riza, A., Nugroho, A., & Laksono, J. (2015). *Optimization of PID Controller Parameters on Flow Rate Control System Using Multiple Effect Evaporator Particle Swarm Optimization*. *5*(2).

Moch Febriawa, H., & Yuliyanto Agung, P. (2021). Sistem Kontrol Pemanas Air Kamar Mandi Menggunakan PID *Controller. Prosiding Seminar Nasional Teknik Elektro, Sistem Informasi, dan Teknik Informatika*, 155-160. Surabaya, 26 Juni 2021. https://doi.org/10.31284/p.snestik.2021.1777

Khan, T. (2018). Design of an Autonomous Smart Shower With Sensors and Actuators. *International Journal of Embedded Systems and Applications*, *8*(1/2/3), 01–17. https://doi.org/10.5121/ijesa.2018.8301

Moch Umar, H., & Usman Abdul, H. (2021). Purwarupa Alat Untuk Memantau Kecepatan Putaran Pada Mesin Berputar Berbasis IoT. *Jurnal Teknik Elektro*, (Vol 5, No 2). http://dx.doi.org/10.31000/jte.v5i2.6997

Siswanto, S., Anif, M., Hayati, D. N., & Yuhefizar, Y. (2019). Pengamanan pintu ruangan menggunakan arduino mega 2560, mq-2, dht-11 berbasis android. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, *3*(1), 66-72.

Tahir, M., Ismat, N., Rizvi, H. H., Zaffar, A., Nabeel Mustafa, S. M., & Khan, A. A. (2022). Implementation of a smart energy meter using blockchain and Internet of Things: A step toward energy conservation. *Frontiers in Energy Research*, *10*, 1029113.

Moon, M. A. M., Kaimujjaman, M., Hossain, M. M., Islam, M. M., & Hossain, M. S. (2023). Seamless real-time thermal imaging system with ESP8266: wireless data transfer and display using UDP. *SN Applied Sciences*, *5*(11), 296.

Kusumadiarti, R. S., & Qodawi, H. (2021). Implementasi Sensor Water Level Dalam Sistem Pengatur Debit Air Di Pesawahan. *PETIK: Jurnal Pendidikan Teknologi Informasi Dan Komunikasi*, *7*(1), 19-29.

Daoud, A. M. I. N. E. (2021). An Arduino-based low-cost hardware for temperature control. *WSEAS Trans. Syst*, *20*, 54-66.

Hasan, D., & Ismaeel, A. (2020). Designing ECG monitoring healthcare system based on internet of things blynk application. *Journal of applied science and technology trends*, *1*(2), 106-111.

Z. Xiaodong, L. Yongqiang & X. Anke, "The Study of the Generalized PID Algorithm," in Proc. of the International Forum on Computer Science-Technology and Applications, Chongqing, 2009, pp. 255-258.

Mary, A. H. (2011). Generalized PID *Controller* Based On Particle Swarm Optimization. *Iraqi Journal of Computers, Communication and Control & Systems Engineering*, *11*, 114-122.

Wang, L. (2020). *PID control system design and automatic tuning using MATLAB/Simulink*. John Wiley & Sons.

Li, X., Chen, M., & Tsutomu, Y. (2013, April). A method of searching PID controller's optimized coefficients for buck converter using particle swarm optimization. In *2013 IEEE 10th International Conference on Power Electronics and Drive Systems (PEDS)* (pp. 238-243). IEEE.

Piotrowski, A., Napiorkowski, J., & Piotrowska, A. (2020). Population size in Particle Swarm Optimization. Swarm Evol. Comput., 58, 100718. https://doi.org/10.1016/j.swevo.2020.100718.

Zaina, Mansour. (2013). Al-Tafsir Al-Muneer of Al-Zuhaili: Study and Evaluation. An-Najah University Journal for Research - B (Humanities). 27. 2043-2076. 10.35552/0247-027-010-002.

Muji, M. (2020). Fundamental Taksonomi Bloom dalam Sistem Pendidikan Menurut QS Al-'Alaq Ayat 1-5 (Telaah Tafsir Ibnu Katsir, Al-Qurthubi dan Salman). *TADRIBUNA: Journal of Islamic Education Management*, *1*(1), 81-106.

UBAIDILLAH, A. (2016). *KAJIAN ATAS AYAT-AYAT TENTANG WAKTU DAN KORELASINYA DENGAN PELAKSANAAN SHALAT* (Doctoral dissertation, Institut PTIQ Jakarta).

Al Hazmi, M. A., Azizah, F. H. N., Hajar, S., Ahmad, H. J., & Al Abrar, M. R. (2024). Kerusakan Alam dan Mitigasi Krisis Lingkungan (Kajian Surat Al-Baqarah Ayat 205-207 dalam Tafsir Al-Maraghi). *Ulumul Qur'an: Jurnal Kajian Ilmu Al-Qur'an dan Tafsir*, *4*(1), 75-92.

Rusmin, M. B., & Al Qifari, A. (2023). Nilai-nilai pendidikan akhlak dalam kitab *Al-Arba'in al-Nawawiyah* karya Imam Nawawi. *Volume XII*, *Nomor 2*, *Juli-Desember*, 763. UIN Alauddin Makassar.

LAMPIRAN

Lampiran I Kode sumber PID *Controller*

```
def simulate_pid(Kp, Ki, Kd, time_data, flow_setpoints):
    dt = 0.1  # Interval waktu
    itae = 0  # Inisialisasi ITAE
    process_var = 0  # Nilai awal posisi keran (0 derajat)

    for i in range(len(time_data)):
        setpoint = flow_setpoints[i]
        integral = 0
        prev_error = 0

      for t in range(200):  # 200 iterasi per titik data
      time = t * dt  # Waktu simulasi saat ini
      error = setpoint - process_var
      integral += error * dt

      derivative = (error - prev_error) / dt
      output = Kp * error + Ki * integral + Kd * derivative

      # Perbarui variabel proses
      process_var += output * 0.01
      process_var = np.clip(process_var, 0, 90)

      # Hitung ITAE
      itae += time * abs(error)  # Bobot waktu absolute error

      prev_error = error

    return itae

def fitness(params):
    Kp, Ki, Kd = params
    return simulate_pid(Kp, Ki, Kd, time_data, flow_setpoints)
```

Lampiran II Kode sumber *tuning* PSO

```
class PSO:
    def __init__(self, fitness, lower_bounds, upper_bounds,
swarm_size, max_iter, w, c1, c2):
        """
      Inisialisasi algoritma PSO (Particle Swarm Optimization).
      :param fitness: Fungsi fitness yang ingin diminimalkan.
      :param lower_bounds: Batas bawah untuk ruang pencarian.
      :param upper_bounds: Batas atas untuk ruang pencarian.
      :param swarm_size: Jumlah partikel dalam swarm.
      :param max_iter: Jumlah iterasi maksimum.
      :param w: Koefisien inersia (inertia weight).
      :param c1: Koefisien akselerasi personal (kognitif).
      :param c2: Koefisien akselerasi global (sosial).
        """
      self.fitness = fitness
      self.lower_bounds = lower_bounds
      self.upper_bounds = upper_bounds
      self.swarm_size = swarm_size
      self.max_iter = max_iter
      self.w = w
      self.c1 = c1
      self.c2 = c2

    def optimize(self):
      # Inisialisasi partikel
      # Init partikel secara acak dalam ruang pencarian.
      particles = np.random.uniform(self.lower_bounds,
      self.upper_bounds, (self.swarm_size,
      len(self.lower_bounds)))
      velocities = np.zeros_like(particles)
      # Kecepatan awal semua partikel diatur ke nol.
      personal_best_positions = np.copy(particles)
      # Posisi pbest awal adalah posisi awal partikel.
      personal_best_values =
      np.array([self.fitness(p) for p in particles])
      # Evaluasi fitness awal untuk setiap partikel.
      global_best_position =
      personal_best_positions[np.argmin(personal_best_values)]
      # Posisi gbest awal.
      global_best_value = np.min(personal_best_values)
      # Nilai fitness gbest awal.
```

```
        # Loop utama optimasi
        for iter_num in range(self.max_iter):
            for i in range(self.swarm_size):
                # Update kecepatan partikel sesuai rumus PSO.
                  velocities[i] = (self.w * velocities[i] +
                  self.c1 * np.random.random() *
                  (personal_best_positions[i] - particles[i]) +
                  self.c2 * np.random.random() *
                  (global_best_position - particles[i]))

                # Update posisi partikel berdasarkan kecepatan.
                particles[i] += velocities[i]

                # Partikel tetap dalam batas ruang pencarian.
                  particles[i] = np.clip(particles[i],
                  self.lower_bounds, self.upper_bounds)

                # Evaluasi nilai fitness untuk posisi baru.
                current_value = self.fitness(particles[i])
                if current_value < personal_best_values[i]:
                    personal_best_values[i] = current_value
                    personal_best_positions[i] =
                    particles[i]  # Perbarui posisi gbest.
            new_global_best_value =
            np.min(personal_best_values)
            if new_global_best_value < global_best_value:
                global_best_value = new_global_best_value
                  global_best_position
                  personal_best_positions[np.argmin(personal_be
                  st_values)]
            print(f'End of iteration {iter_num + 1},
            Best Score: {global_best_value}')
            print(f'Personal Best Values:
            {personal_best_values}')
            print(f'Global Best Position:
            {global_best_position}')

        # Kembalikan posisi gbest dan nilai fitness terbaik.
        return global_best_position, global_best_value

pso = PSO(fitness, lower_bounds, upper_bounds,
swarm_size=swarm_size, max_iter=max_iter, w=w, c1=c1, c2=c2)
optimal_params, optimal_value = pso.optimize()
Kp_opt, Ki_opt, Kd_opt = optimal_params
print(f'Optimal Parameters: Kp = {Kp_opt}, Ki = {Ki_opt}, Kd =
{Kd_opt}')
print(f'Optimal ITAE Fitness Value: {optimal_value}')
```

Lampiran III Kode sumber olahan dari pengumpulan data

```
df = pd.read_csv('hasil_pengumpulan_data.csv')
# Pastikan data terurut berdasarkan Menit
df = df.sort_values(by='Menit')

# Membuat plot dengan data per menit
plt.figure(figsize=(12, 6))

# Plot setiap variabel terhadap waktu (Menit)
plt.plot(df['Menit'], df['Aliran Air Aktual (mL/s)'],
label="Aliran Air Aktual (mL/s)", color='green', marker='o')
plt.plot(df['Menit'], df['Aliran Air Minimal (mL/s)'],
label="Aliran Air Minimal (mL/s)", color='red', linestyle='--')
plt.plot(df['Menit'], df['Aliran Air Maksimal (mL/s)'],
label="Aliran Air Maksimal (mL/s)", color='orange',
linestyle='--')

# Menambahkan label dan judul
plt.xlabel('Menit')
plt.ylabel('Nilai')
plt.legend()

# Menambahkan grid untuk kejelasan
plt.grid(True)

# Menampilkan plot
plt.show()

# Calculate 4-minute averages
df['Group'] = (df['Menit'] - 1) // 4  # Group rows into blocks
of 4 minutes
averages = df.groupby('Group').mean()

# Format the resulting DataFrame
averages['Menit'] = averages.index * 4 + 4  # Assign
appropriate 'Menit' values
averages = averages.reset_index(drop=True)[[
    "Menit", "Roda Target (RPM)", "Roda Aktual (RPM)",
    "Aliran Air Aktual (mL/s)", "Aliran Air Minimal (mL/s)",
"Aliran Air Maksimal (mL/s)"
]]

# Save to CSV
filename = "hasil_olahan_pengumpulan_data.csv"
averages.to_csv(filename, index=False)
```

Lampiran IV Kode sumber analisis pengumpulan data

```
x = df['Menit'].values.reshape(-1, 1)
y = df['Aliran Air Aktual (mL/s)'].values

model = LinearRegression()

model.fit(x, y)

m = model.coef_[0]  # Kemiringan (slope)
b = model.intercept_  # Intersep

print(f"Persamaan linier: y = {m:.2f}x + {b:.3f}")

plt.figure(figsize=(10, 6))
plt.scatter(x, y, color='blue', label='Kecepatan Air Aktual
(mL/s)')
plt.plot(x, model.predict(x), color='red', label=f'Regresi
Linier: y = {m:.2f}x + {b:.3f}')
# plt.xlabel('Kecepatan Putaran Roda (RPM)')
plt.xlabel('Waktu (Menit)')
plt.ylabel('Kecepatan Air Aktual (mL/s)')
plt.legend()
plt.grid(True)
plt.show()

if 'Iterasi' in df.columns:
    print("\nAnalisis per Iterasi:")
    iterasi_unique = df['Iterasi'].unique()
    for iterasi in iterasi_unique:
        # Ambil subset data untuk iterasi tertentu dan hapus
nilai kosong
        subset = df[df['Iterasi'] ==
iterasi].dropna(subset=['Menit', 'Aliran Air Aktual (mL/s)'])

        # Cek jika subset memiliki data yang cukup untuk
regresi
        if len(subset) > 1:  # Memastikan setidaknya ada 2 data
untuk regresi
            x_iter = subset['Menit'].values.reshape(-1, 1)
            y_iter = subset['Aliran Air Aktual (mL/s)'].values

            # Model untuk setiap iterasi
            model_iter = LinearRegression()
            model_iter.fit(x_iter, y_iter)

            m_iter = model_iter.coef_[0]
```

Lampiran V Kode sumber penentuan variabel PSO

```
# Define controlled bounds for PID parameters
lower_bound_kp = np.random.uniform(80.0, 90.0)
upper_bound_kp = np.random.uniform(91.0, 100.0)

lower_bound_ki = np.random.uniform(0.1, 1.0)
upper_bound_ki = np.random.uniform(1.1, 1.5)

lower_bound_kd = np.random.uniform(0.1, 0.5)
upper_bound_kd = np.random.uniform(0.6, 1.0)

# Store bounds in arrays
lower_bounds = np.array([lower_bound_kp, lower_bound_ki,
lower_bound_kd])
upper_bounds = np.array([upper_bound_kp, upper_bound_ki,
upper_bound_kd])

# PSO parameters
swarm_size = 50  # Moderate size for optimization
max_iter = 100   # Sufficient iterations for convergence

# Inertia and acceleration coefficients for PSO
w = 0.5  # Lower inertia weight to encourage convergence
c1 = 0.2
c2 = 0.2

# Create a table to display parameter settings
table = PrettyTable()
table.field_names = ["Parameter", "Value"]

# Populate table with PSO and bounds values for display
table.add_row(["Swarm Size", swarm_size])
table.add_row(["Max Iterations", max_iter])
table.add_row(["Lower Bounds (Kp, Ki, Kd)", lower_bounds])
table.add_row(["Upper Bounds (Kp, Ki, Kd)", upper_bounds])
table.add_row(["w (Inertia)", w])
table.add_row(["c1 (Cognitive)", c1])
table.add_row(["c2 (Social)", c2])

# Display the table
print(table)
```

Lampiran VI Kode sumber *ESP8266*

```
#include <Servo.h> // Include library Servo

Servo myServo; // Buat objek Servo

float currentAngle = 0;      // Posisi awal servo
float flowRate = 0.00;       // Laju aliran air dalam mL/s
float volume = 0;            // Total volume air dalam mL
unsigned long previousMillis = 0;      // Untuk interval 1
detik
unsigned long servoMillis = 0;         // Untuk interval 2
detik
unsigned long lastUpdateMillis = 0;    // Untuk pembaruan
setiap 10 detik
const unsigned long interval = 1000;   // Interval perhitungan
(1 detik)
const unsigned long servoInterval = 2000; // Interval
pergerakan servo (2 detik)

// Rentang awal untuk currentAngle dan flowRate
float minAngle = 1;
float maxAngle = 15;
float minFlowRate = 0.15;
float maxFlowRate = 0.25;

void setup() {
    Serial.begin(9600);
    myServo.attach(4);
    myServo.write(0);
}

void loop() {
    unsigned long currentMillis = millis();

    // Lakukan perhitungan setiap 1 detik
    if (currentMillis - previousMillis >= interval) {
        previousMillis = currentMillis;

        // Hasilkan nilai random untuk flowRate dalam rentang
        flowRate = random((minFlowRate * 100), (maxFlowRate *
100)) / 100.0;

        // Tambahkan flowRate ke volume total
        volume += flowRate;
        currentAngle = random((minAngle * 10), (maxAngle * 10))
/ 10.0;
```

```
        // Atur posisi servo secara perlahan ke sudut yang
diinginkan
        myServo.write(currentAngle);

        // Tampilkan sudut servo
        Serial.print("Sudut Servo: ");
        Serial.print(currentAngle); // Tampilkan currentAngle
dengan 1 digit desimal
        Serial.println(" Derajat");

        // Tampilkan data ke Serial Monitor
        Serial.print("Kecepatan Air: ");
        Serial.print(flowRate, 2); // Tampilkan flowRate dengan
2 digit desimal
        Serial.println(" mL/s");

        Serial.print("Volume: ");
        Serial.print(volume, 2); // Tampilkan volume total
dengan 2 digit desimal
        Serial.println(" mL");

        Serial.print("Waktu: ");
        Serial.print(currentMillis / 1000); // Waktu dalam
detik sejak program mulai
        Serial.println(" detik");
    }

    // Update rentang setiap 10 detik
    if (currentMillis - lastUpdateMillis >= 10000) {
        lastUpdateMillis = currentMillis; // Perbarui waktu
terakhir rentang diubah

        minAngle += 5;
        maxAngle += 5;
        minFlowRate += 0.22;
        maxFlowRate += 0.22;

        // Batasi nilai maksimal untuk flowRate
        if (maxFlowRate > 1.1) {
            maxFlowRate = 1.1;
            minFlowRate = 0.88;
            // Pastikan minFlowRate tetap sesuai
        }
    }
}
```

Lampiran VII Kode sumber *Atmega2560*

```
#include <PID_v1.h>
#include <Servo.h>

// Deklarasi variabel
const int pinSensorAir = 2;
const int pinSensorRoda = 3;

// Konstanta simulasi
float actualFlow = 0.22;  // Flowrate aktual awal (mL/s)
float minFlow = 0.11;     // Flowrate minimal awal (mL/s)
float maxFlow = 0.32;     // Flowrate maksimal awal (mL/s)

volatile int pulseCount = 0;                 // Hitungan pulsa
untuk sensor air
volatile unsigned long previousPulseTime = 0; // Waktu pulsa
sebelumnya untuk sensor roda
volatile unsigned long timeDifference = 0;   // Perbedaan waktu
antar pulsa roda

float flowRate = 0;      // Debit air (mL/s)
float rpm = 0;           // Putaran per menit (RPM) roda saat
ini
float avgRpm = 0;        // Rata-rata RPM roda dengan smoothing

// Konstanta kalibrasi dan konfigurasi
const float conversionFactor = 0.219;       // Faktor konversi
untuk flow rate
const float alpha = 0.3;                    // Faktor smoothing
untuk perhitungan RPM

// Konfigurasi roda
const int markers = 1;                      // Jumlah penanda
pada roda

// Interval waktu dan konfigurasi komunikasi Serial
const unsigned long minInterval = 1000000;  // Interval minimum
antar pulsa dalam µs
unsigned long previousMillis = 0;           // Waktu loop
sebelumnya dalam ms
const unsigned long interval = 1000;        // Interval
pembaruan data (ms)c
int elapsedSeconds = 0;              // Waktu yang sudah
berlalu dalam detik
const unsigned long noPulseTimeout = 10000; // Timeout jika
tidak ada pulsa roda (ms)
```

Lampiran VIII Hasil pengumpulan data

12/11/2024 1:36	15.2	0.23
12/11/2024 1:36	49.8	0.23
12/11/2024 1:36	49.8	0.24
12/11/2024 1:36	49.9	0.21
12/11/2024 1:36	49.8	0.23
12/11/2024 1:36	49.8	0.23
12/11/2024 1:36	49.8	0.21
12/11/2024 1:36	49.8	0.23
12/11/2024 1:36	49.8	0.23
12/11/2024 1:36	49.9	0.22
12/11/2024 1:36	49.9	0.2
12/11/2024 1:36	49.9	0.23
12/11/2024 1:36	49.8	0.23
12/11/2024 1:36	49.8	0.21
12/11/2024 1:36	49.8	0.23
12/11/2024 1:36	49.9	0.22
12/11/2024 1:36	49.9	0.22
12/11/2024 1:36	49.9	0.2
12/11/2024 1:36	49.8	0.2
12/11/2024 1:36	49.9	0.2
12/11/2024 1:36	49.8	0.21
12/11/2024 1:36	49.8	0.2
12/11/2024 1:36	49.9	0.2
12/11/2024 1:36	49.9	0.23
12/11/2024 1:36	49.9	0.21
12/11/2024 1:36	49.8	0.2
12/11/2024 1:36	49.8	0.23
12/11/2024 1:36	49.9	0.24
12/11/2024 1:36	49.9	0.23
12/11/2024 1:36	49.9	0.2
12/11/2024 1:36	49.9	0.21
12/11/2024 1:36	49.9	0.2
12/11/2024 1:36	49.9	0.21
12/11/2024 1:36	49.8	0.2

Data lengkapnya dapat dilihat pada tautan berikut https://drive.google.com/file/d/1E9XgwIDhMahe7rhsPY3StP2TKtnwkdq8/view?usp=sharing

Lampiran IX Hasil olahan pengumpulan data

Menit	Roda Target (RPM)	Roda Aktual (RPM)	Aliran Air Aktual (mL/s)	Aliran Air Minimal (mL/s)	Aliran Air Maksimal (mL/s)
4	50	49.77	0.22	0.209	0.231
8	50	49.98	0.44	0.429	0.451
12	50	49.91	0.65	0.649	0.671
16	50	49.8	0.88	0.861	0.891
20	50	49.57	1.1	1.009	1.111

Lampiran X Hasil akhir pengujian data

12/11/2024 1:36	15.2	0.23
12/11/2024 1:36	49.8	0.23
12/11/2024 1:36	49.8	0.24
12/11/2024 1:36	49.9	0.21
12/11/2024 1:36	49.8	0.23
12/11/2024 1:36	49.8	0.23
12/11/2024 1:36	49.8	0.21
12/11/2024 1:36	49.8	0.23
12/11/2024 1:36	49.8	0.23
12/11/2024 1:36	49.9	0.22
12/11/2024 1:36	49.9	0.2
12/11/2024 1:36	49.9	0.23
12/11/2024 1:36	49.8	0.23
12/11/2024 1:36	49.8	0.21
12/11/2024 1:36	49.8	0.23
12/11/2024 1:36	49.9	0.22
12/11/2024 1:36	49.9	0.22
12/11/2024 1:36	49.9	0.2
12/11/2024 1:36	49.8	0.2
12/11/2024 1:36	49.9	0.2
12/11/2024 1:36	49.8	0.21
12/11/2024 1:36	49.8	0.2
12/11/2024 1:36	49.9	0.2
12/11/2024 1:36	49.9	0.23
12/11/2024 1:36	49.9	0.21
12/11/2024 1:36	49.8	0.2
12/11/2024 1:36	49.8	0.23
12/11/2024 1:36	49.9	0.24
12/11/2024 1:36	49.9	0.23
12/11/2024 1:36	49.9	0.2
12/11/2024 1:36	49.9	0.21
12/11/2024 1:36	49.9	0.2
12/11/2024 1:36	15.2	0.23
12/11/2024 1:36	49.8	0.23

Data lengkapnya dapat dilihat pada tautan berikut

https://docs.google.com/spreadsheets/d/1QuiUxhBTHCATYfW_hCYRJVPYKrgyEQTn/edit?usp=sharing&ouid=112846875622497242472&rtpof=true&sd=true