SISTEM MONITORING KEKERUHAN AIR BERBASIS *SCADA* DAN *IOT* DENGAN ANTARMUKA *ANDROID* MENGGUNAKAN SENSOR *ENDRESS AND HAUSER*

LAPORAN TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat Untuk Menyelesaikan Studi

Jenjang Program Diploma Tiga

Oleh:

Nama : Septyawan Bagus Wibowo

NIM : 24014012

PROGRAM STUDI DIII TEKNIK ELEKTRONIKA

POLITEKNIK HARAPAN BERSAMA TEGAL

2025

HALAMAN PERNYATAAN KEASLIAN

Yang bertanda tangan dibawah ini :

Nama : Septyawan Bagus Wibowo

NIM : 24014012

Adalah mahasiswa Program Studi DIII Teknik Elektronika Politeknik Harapan Bersama Tegal, dengan ini menyatakan bahwa Laporan Tugas Akhir berjudul:

"SISTEM MONITORING KEKERUHAN AIR BERBASIS *SCADA* DAN *IOT* DENGAN ANTARMUKA *ANDROID* MENGGUNAKAN SENSOR *ENDRESS AND HAUSER*"

Merupakan hasil pemikiran saya sendiri secara orisinil dan saya susun secara mandiri dengan tidak melanggar kode etik hak karya cipta. Pada laporan Tugas Akhir ini juga bukan merupakan karya yang pernah diajukan untuk memperoleh gelar akademik tertentu di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila dikemudian hari ternyata Laporan Tugas Akhir ini terbukti melanggar kode etik karya cipta atau merupakan karya yang dikategorikan mengandung unsur plagiarisme, maka saya bersedia untuk melakukan penelitian baru dan menyusun laporannya sebagai Laporan Tugas Akhir sesuai dengan ketentuan yang berlaku.

Demikian pernyataan ini saya buat dengan sebenarnya dan sesungguhnya.

Yang membuat pernyataan

Semarang, 23 Juni 2026

Septyawan Bagus W.

24014012

HALAMAN PERNYATAAN PUBLIKASI

Sebagai sivitas akademik Politeknik Harapan Bersama, saya bertanda tangan dibawah ini :

Nama : Septyawan Bagus Wibowo

NIM : 24014012

Prodi Studi : DIII Teknik Elektronika

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Politeknik Harapan Bersama Tegal **Hak Bebas Royalti Nonekslusif** ***(Non-exclusive Royalty-Free Right)*** atas karya ilmiah yang berjudul :

" SISTEM MONITORING KEKERUHAN AIR BERBASIS SCADA DAN IOT DENGAN ANTRAMUKA ANDROID MENGGUNAKAN SENSOR *ENDRESS AND HAUSERR*"

Beserta perangkat yang ada. Dengan Hak Bebas Royalti non ekslusif ini Politeknik Harapan Bersama Tegal berhak menyimpan , mengalih media/format, mengelola dalam bentuk data pangkalan *(database),* merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat Di : Semarang

Pada tanggal : 30 Juni 2025

Yang Menyatakan

(Septyawan Bagus W.)

HALAMAN REKOMENDASI

Laporan Tugas Akhir (TA) yang berjudul **"SISTEM MONITORING KEKERUHAN AIR BERBASIS SCADA DAN IOT DENGAN ANTARMUKA ANDROID MENGGUNAKAN SENSOR *ENDRESS AND HAUSERR*"** yang disusun oleh Septyawan Bagus W. , NIM 24014012 telah mendapat persetujuan pembimbing dan siap dipertahankan di depan Tim Penguji Laporan Tugas Akhir (TA) Program Studi DIII Teknik Elektronika Politeknik Hrapan Bersama Tegal.

Semarang, 23 Juni 2025

Mengetahui,

Pembimbing I,	Pembimbing II,
Bahrun Niam M.T	Qirom S.Pd, M.T
NIPY. 09.015.277	NIPY. 09.015.281

HALAMAN PENGESAHAN

Judul	:	**SISTEM MONITORING KEKERUHAN AIR BERBASIS *SCADA* DAN *IOT* DENGAN ANTARMUKA *ANDROID* MENGGUNAKAN SENSOR *ENDRESS AND HAUSER***
Nama	:	Septyawan Bagus Wibowo
NIM	:	24014012
Program Studi	:	TeknikElektronika
Jenjang	:	Diploma Tiga

Dinyatakan LULUS setelah dipertahankan di depan Tim Penguji Laporan Tugas Akhir Program Studi DIII Teknik Elektronika Politeknik Harapan Bersama Tegal

Tegal, 6 Agustus 2025
Tim Penguji:

	Nama	Tanda Tangan
1. Ketua Penguji	: Much. Sobri Sungkar, M.Kom.	.....
2. Penguji I	: Rony Darpono, M.T.	.....
3. Penguji II	: Dany Sucipto, S.T.	.....

Mengetahui,
Ketua Program Studi DIII Teknik Elektronika
Politeknik Harapan Bersama Tegal

Rony Darpono, M.T.
NIPY.09.015.282

HALAMAN MOTO

1. "Jadilah baik. Sesungguhnya Allah menyukai orang-orang yang berbuat baik." - Q.S Al Baqarah: 195
2. Hiduplah Di Masa Kini Jangan Berpikir Terlalu Keras untuk Masa Lalu
3. Teknologi untuk kehidupan yang lebih baik
4. "Saat ini adalah milik mereka; masa depan, yang sebenarnya saya kerjakan adalah milik saya", Nikolai Tesla
5. "Tidak ada yang akan menuai kecuali apa yang mereka tabur." -QS Al-An'am: 164

HALAMAN PERSEMBAHAN

PERSEMBAHAN:

1. Keluarga khususnya istri yang memberikan semangat dan dukungan selama penyusunan tugas akhir.
2. Bapak Qirom S.Pd, M.T selaku Pembimbing I yang telah membimbing dalam penyusunan laporan ini.
3. Bapak Bahrun Niam selaku Pembimbing II yang telah membimbing dalam penyusunan laporan ini.
4. Bapak dan Ibu Dosen Prodi DIII Teknik Elektronika Politeknik Harapan Bersama Tegal.
5. Rekan-rekan Tim Maintenance PT Air Semarang Barat & Moya Holding yang telah memberikan masukkan dan semangat selama penyusunan Laporan Tugas Akhir.

KATA PENGANTAR

Puji syukur saya panjatkan ke hadirat Allah SWT atas segala limpahan rahmat dan karunia-Nya sehingga saya dapat menyelesaikan laporan tugas akhir yang berjudul "Sistem Monitoring Kekeruhan Air Berbasis *SCADA* dan *IoT* dengan Antarmuka *Android* Menggunakan Sensor *Endress and Hauser*" ini dengan baik. Laporan ini disusun sebagai salah satu syarat untuk menyelesaikan pendidikan pada program studi DIII Teknik Elektronika Politeknik Harapan Bersama Tegal.

Dalam penyusunan tugas akhir ini, saya banyak mendapat bantuan, bimbingan, dan dukungan dari berbagai pihak. Oleh karena itu, saya mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Bapak Qirom S.Pd, M.T, selaku pembimbing yang telah memberikan arahan, ilmu, dan motivasi selama proses penelitian berlangsung.
2. Bapak Bahrun Niam, yang telah memberikan masukan berharga sehingga laporan ini dapat tersusun dengan baik.
3. Semua pihak, keluarga, dan teman-teman yang telah memberikan dukungan moral maupun material selama pengerjaan tugas akhir.

Penelitian ini bertujuan untuk mengembangkan sistem monitoring kekeruhan air yang menggabungkan teknologi SCADA dan IoT agar dapat memudahkan pemantauan kualitas air secara real-time melalui antarmuka aplikasi Android. Penggunaan sensor Endress and Hauser diharapkan memberikan data yang akurat dan andal dalam pengukuran kekeruhan air.

Selama proses pengerjaan, penulis menyadari masih banyak kekurangan dan keterbatasan dalam laporan ini. Oleh karena itu, kritik dan saran yang membangun sangat penulis harapkan demi kesempurnaan laporan ini.Akhir kata, semoga laporan ini dapat memberikan manfaat dan kontribusi positif bagi pengembangan teknologi dan ilmu pengetahuan.

ABSTRAK

PT Air Semarang Barat merupakan perusahaan pengolahan air bersih yang bersumber dari aliran sungai dan air yang digunakan membawa banyak zat pengotor sehingga menyebabkan kendala dalam pengolahannya apabila tidak dipantau kualitasnya sehingga perlunya untuk memonitoring kualitas air pada grit chamber intake PT Air Semarang Barat menjadi kebutuhan dalam operasional pengolahan air bersih. Berdasarkan permasalahan diatas maka dirangcang sebuah alat untuk memontoring sensor kekeruhan air pada girt chamber intake PT Air Semarang Barat. Tujuan penelitian ini adalah untuk merancang Sistem monitoring kekeruhan air berbasis *Supervisory Control and Data Acquisition (SCADA)* dan *Internet of Things (IoT)* dengan antarmuka *Android* dirancang untuk memantau kualitas air secara real-time. Sistem ini menggunakan sensor *Endress and Hauser* untuk mengukur tingkat kekeruhan air pada grit chamber intake PT Air Semarang Barat guna memudahkan dalam memonitoring tingkat kekeruhan air.Penelitian ini menggunakan metode penelitian dan pengembangan (*R&D*) agar mempermudah dalam identifikasi masalah sampai dengan tahapan pembuatan alat dan *software*.Sistem monitoring ini menggunakan sensor *CUS52D* yang keluaran sinyal akan diproses oleh mikrokontroler *ESP32* dan ditransmisikan melalui protokol *MQTT* untuk komunikasi ke server dan setiap perangkat yang terhubung dapat meneriman data tersebut dengan menggunakan topic "data/turbid526/ntu/4456"

Kata kunci: ***IoT,Endress and Hauser*****, Antarmuka** ***Android, Software Haiwell SCADA.***

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Air bersih adalah kebutuhan dasar manusia dan merupakan fondasi penting bagi kesehatan, kesejahteraan, dan pembangunan berkelanjutan. Namun, ketersediaan air bersih yang memadai semakin terancam oleh berbagai faktor global dan lokal. Krisis air bersih bukan hanya masalah sumber daya, tetapi juga masalah kebersamaan dalam menjaga alam dan lingkungan. Kebutuhan air bersih yang semakin meningkat di kota-kota besar menuntut solusi inovatif dan berkelanjutan. Pemanfaatan air sungai sebagai sumber alternatif untuk penjernihan menawarkan potensi besar, namun memerlukan pengelolaan yang hati-hati dan teknologi yang tepat, agar lingkungan sekitar tetap lestari serta memenuhi ketersediaan air bersih yang aman dan berkelanjutan bagi semua penduduknya.

PT Air Semarang Barat yang terletak di Jl. Untung Suropati No. 01 A, Kelurahan Bambankerep, Kecamatan Ngaliyan, Kota Semarang, merupakan perusahaan yang berfokus pada pengelolaan air bersih dan memainkan peran krusial dalam memenuhi kebutuhan air bersih masyarakat. Sebagai bagian dari Proyek Strategis Negara, PT Air Semarang Barat memiliki tanggung jawab besar dalam mengolah air baku (air sungai) menjadi air yang aman untuk dikonsumsi oleh masyarakat. Proses pengolahan air bersih di perusahaan ini dilakukan melalui dua metode yaitu pengolahan fisika dan pengolahan kimia. Metode pengolahan fisika memanfaatkan sifat mekanis

air, seperti pengendapan dan penyaringan (filtrasi)[1], untuk menghilangkan partikel-partikel yang tidak diinginkan. Sementara itu, pengolahan kimia melibatkan penambahan zat kimia seperti PAC dan Gas Chlorine[2], yang berfungsi untuk menghilangkan logam-logam berat dan menjernihkan air, sehingga kualitas air yang dihasilkan memenuhi standar kesehatan.

Masalah dalam pengolahan sumber daya air sering kali disebabkan oleh pencemaran, pertumbuhan populasi, dan perubahan iklim. Faktor cuaca dan iklim memiliki dampak signifikan terhadap kualitas air baku yang tersedia. Selama musim kemarau, ketersediaan air baku menjadi terbatas, sementara pada musim hujan, meskipun ketersediaan air meningkat, kualitasnya sering kali menurun akibat pencampuran dengan lumpur, pasir, dan kerikil yang umum ditemukan di sungai. Bercampurnya partikel-partikel ini menjadi tantangan tersendiri dalam memastikan air baku yang digunakan memenuhi standar kualitas yang aman untuk dikonsumsi, maka dari itu perlunya pemantauan kualitas kekeruhan air merupaka bagian yang dibutuhkan dalam proses penjernihan air mengingat bahwa air baku sering mengandung partikel-partikel terlarut dalam air dan akan menentukan seberapa banyak bahan kimia yang perlu dicampurkan untuk menghilangkan partikel yang terlarut sehingga dapat memenuhi persyaratan *Keputusan Menteri Lingkungan Hidup No.115 Tahun 2003* Tentang Pedoman Penentuan Status Mutu Air. Keputusan ini memberikan pedoman dalam menentukan status mutu air, termasuk air baku yang akan dijernihkan.[3]

Water Treatment Plant (WTP) atau Instalasi Pengolahan Air (IPA) adalah suatu sistem yang dirancang untuk mengolah air baku (*raw water*) dan suplai air bersih untuk kegiatan domestik maupun industri.[4] Proses pengolahan air ini meliputi beberapa tahapan, seperti koagulasi, flokulasi, sedimentasi, filtrasi, dan desinfeksi, yang bertujuan untuk menghilangkan kontaminan seperti partikel padat, mikroorganisme, dan zat kimia berbahaya. Pemantauan kualitas air baku secara *real-time* menjadi krusial dalam proses ini untuk memastikan bahwa air yang diolah memenuhi persyaratan mutu sebelum masuk ke tahap penjernihan. Dalam upaya pemenuhan persyaratan mutu kualitas air baku yang digunakan perlunya sensor kekeruhan air terpasang pada bak penampungan air sementara sebelum masuk proses penjernihan air, menjadi poin yang penting karena adanya alat tersebut menjadi salah satu acuan layak tidak-nya air baku digunakan untuk proses penjernihan menjadi air bersih. Selama ini proses pemantauan kualitas air baku hanya dilakukan di area proses penjernihan melalui SCADA (*Supervisory Control and Data Acquisition*) sebelum air masuk pada bak penjernihan dan sensor yang berada di *grit chamber* belum terintegrasi dengan baik sehingga apabila kualitas air baku tidak sesuai persyaratan maka air dapat berpotensi gagal dalam proses penjernihan. Perlunya integrasi sistem monitoring sensor kekeruhan air pada bak penampung sementara di grit chamber PT Air Semarang Barat dalam operasional *Water Treatment Plant* (*WTP*) menggunakan *ESP32* dan *IoT*[5] menjadi salah satu solusi untuk mengatasi persoalan diatas dengan harapan adanya sensor kekeruhan pada bak penampung sementara yang sudah

terintegrasi menjadi sebuah peringatan apabila kualitas air baku menurun maka dapat segera bertindak dan dapat mengurangi resiko gagal proses.

1.2. Rumusan Masalah

1. Bagaimana rancangan sistem monitoring kekeruhan air pada *grit chamber* PT Air Semarang Barat?
2. Bagaimana membuat alat untuk monitoring kekeruhan air dengan *IoT* pada *grit chamber* PT Air Semarang Barat?
3. Bagaiman akurasi sistem monitoring tersebut?

1.3. Batasan Masalah

Dalam sistem monitoring kekeruhan air, telah ditetapkan batasan yang digunakan agar tidak meluas dalam penelitian, batasan tersebut yaitu seberapa akurat pembacaan yang didapat dari sensor, serta realtime pembacaan yang dikirim oleh sensor *Endress and Hauser* dengan tipe *Tubimax CUS-52D* melalui *ESP32* menggunakan protokol *MQTT* dan menampilkan pesan tersebut dalam *SCADA Haiwell* serta aplikasi *Android* di smartphone.

1.4. Tujuan

Tujuan dari penelitian ini adalah membuat system monitoring kekeruhan air dengan sensor turbidity yang sudah terpasang yaitu *Endress and Hauser* dengan seri *Turbimax CUS-52D* berbasis *IoT* dan dapat menampilkan hasil pembacaan sensor ke *SCADA Haiwell* di area *WTP* dan dapat dipantau juga melalui aplikasi berbasis *android.*

1.5. Manfaat

1.5.1. Manfaat Teoritis

Dengan adanya penelitian ini berharap dapat berkontribusi pada pengembangan teknologi *Internet of Things (IoT)* dalam pengelolaan sumber daya air. Dengan memanfaatkan sensor turbidity dan sistem *SCADA*, penelitian ini dapat menjadi model untuk penerapan teknologi serupa di bidang lain, seperti pertanian dan industri.

1.5.2. Manfaat Praktis

Sistem ini memungkinkan pengelolaan yang lebih efisien di *Water Treatment Plant (WTP)* dengan memberikan data yang akurat dan terkini mengenai kekeruhan air. Hal ini dapat mengurangi waktu dan biaya operasional. Data yang diperoleh dari sensor dapat digunakan untuk analisis lebih lanjut, membantu manajemen dalam membuat keputusan yang lebih baik terkait pengolahan air dan pengelolaan sumber daya air. Serta implementasi teknologi *IoT* dalam sistem monitoring ini menunjukkan kemajuan dalam pengelolaan sumber daya air, yang dapat menjadi model bagi daerah lain dalam penerapan teknologi serupa untuk meningkatkan kualitas air.

1.6. Sistematika Penulisan

Dalam penyusunan Laporan dan Penelitian Tugas Akhir ini, penyusunan diuraikan menjadi beberapa bagian berdasarkan masalah yang akan dibahas, antara lain:

BAB I : PENDAHULUAN

Membahas latar belakang dari Sistem Monitoring Kekeruhan Air Berbasis *SCADA* dan *IoT* dengan Antarmuka *Android* menggunakan Sensor

Endress and Hauser dengan seri *Turbimax CUS-52D*, dari mulai rumusan masalah, batasan masalah, tujuan dan manfaat hingga sistematika penulisan.

BAB II LANDASAN TEORI

Bab ini membahas terkait teori apa saja yang digunakan dalam penyusunan Laporan dan Penelitian Tugas Akhir.

BAB III METODOLOGI PENELITIAN

Dalam bab ini membahas mengenai model penelitian terdahulu sebagai acuan dalam membuat projek, prosedur penelitian, teknik pengumpulan data, instrumen penelitian dan tahap *Research And Development*.

BAB IV PEMBAHASAN

Membahas terkait cara kerja sistem komunikasi dan sensor yang diterapkan pada monitoring kekeruah air, serta analisa lebih lanjut mengenai penelitian yang dilakukan.

BAB V PENUTUP

Menyimpulkan secara singkat dari pembahasan yang telah diuraikan dan memberikan saran untuk pengembangan selanjutnya.

BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Pada penelitian Umar Katu, Yuniarti, dan Nuraeni Umar dengan judul "MODUL PEMBELAJARAN PRAKTIKUM MIKROPROSESOR DAN IOT BERBASIS ESP32 MENGGUNAKAN PROTOKOL *CLOUD* MQTT" mereka menggunakan ESP32 dan sensor DHT11 sebagai inputan data yang kemudian akan diteruskan ke MQTT *Cloud* . hasil pengujian menunjukan keberhasilsan dalam metode pengiriman pesan dari ESP32 menggunakan MQTT Cloud yang dirancang khusus untuk komunikasi aplikasi machine-to-machine yang ringan dan efisien dan dapat menghemat bandiwth. [6]

Pada penelitian Syafriyadi Nor dan Sarifudin dengan judul "*Integration of the XY-MD02 Module in an IoTBased Humidity and Temperature Monitoring System with Graphic Display on Nextion LCD*".memadukan sensor suhu industrial XY-MD02 dengan ESP32 dan MQTT hasil pengujian menunjukan suhu dan kelembapan yang terbaca di sensor XY-MD02 dapat diteruskan dengan baik lewat MQTT berformat *JSON*, dan dapat ditampilkan pada aplikasi android dengan baik, namun cara komunikasi antara ESP32 dan sensor XY-MD02 menggunakan protokol *Modbus*.[7]

Pada penelitian Sri Mahfuza, Uray Ristian, dan Dwi Marisa Midyanti dengan judul "Penerapan Protokol MQTT pada Sistem Otomatisasi Fotosintesis Tanaman *Aquascape* Berbasis *Internet of Things*". Mereka

menggunakan sensor suhu berseri DS18B20, sensor turbidity, ESP32,Relay dan tentunya menggunakan MQTT. Komunikasi sensor DS18B20 menggunakan *Wire-Communication* selayaknya sensor kelas industri dan hasil dari riset tersebut menunjukan performa yang baik dalam pembacaan pengiriman data dan *delay* untuk mengirim data tersebut ke MQTT *server*.

Pada penelitian Muhammad Revaldi Frizky, Seno D. Panjaitan, dan Hendro Priyatman dengan judul "RANCANG BANGUN SCADA DENGAN TEKNOLOGI *INDUSTRIAL IoT* DIIMPLEMENTASIKAN PADA SISTEM KENDALI BERBASIS PLC" yang menggunakan teknologi IoT dan MQTT serta integrase MQTT ke SCADA, Hasil pengujian menunjukan hasil yang baik antara data yang dikirim dan diterima tidak terjadi data hilang, namun kekurangan dari riset tersebut ialah menggunakan PLC yang memiliki harga cukup mahal dan tidak menggunakan mikro-kontroler.

Pada penelitian Heru Prasetyo dan Muhammad Agung Raharjo dengan judul "Pembuatan Sistem Timbangan *Online* Berbasis *Internet of Things*", yang menggunakan sensor timbangan *load cell* HX711 dan ESP32. Hasil dari riset tersebut juga menunjukan hasil yang baik antara timbangan digital dan tradisional, namun kekurangan-nya ialah hasil data yang terekam disimpan pada *SQL-Server* dan penggunaan *server* tersebut memakan cukup biaya.

2.2 Dasar Teori

2.1.1 *WTP (Water Treatment Plant)*

Water Treatment Plant (WTP) atau biasa disebut juga sebagai instalasi Pengolahan Air (IPA), merupakan salah satu teknik manajemen pengolahan air dan suplai air bersih untuk kegiatan domestik maupun kegiatan industri.Water Treatment Plant atau lebih populer dengan akronim WTP adalah bangunan utama pengolahan air bersih dengan cara tertentu dengan tujuan agar mendapatkan air dengan kualitas yang bagus dan bersih.

2.1.2 Golongan Air Minum

Air sering kali mengandung polutan, bahkan tetesan hujan yang jatuh dari langit pun tercemar oleh karbon dioksida dan debu. Air permukaan, yang biasanya menjadi sumber air minum, berasal dari danau, sungai, waduk, mata air, dan saluran pengairan. Sebagian besar pencemar dalam air permukaan berasal dari limbah industri dan rumah tangga.

Peraturan Menteri Kesehatan Republik Indonesia Nomor 32 Tahun 2017 menetapkan standar kualitas air minum untuk kebutuhan hidup manusia, memastikan bahwa air tersebut tidak mengganggu kesehatan dan memiliki nilai estetika yang dapat diterima. Selain itu, Peraturan Pemerintah Republik Indonesia No. 22 Tahun 2021 mengklasifikasikan mutu air menjadi empat golongan:

1. Golongan I : Air yang dapat digunakan sebagai air baku untuk minum dan keperluan lain yang memerlukan mutu serupa.

2. Golongan II : Air untuk sarana rekreasi, budidaya ikan air tawar, irigasi, peternakan, dan keperluan lain yang memerlukan kualitas yang sama.
3. Golongan III: Air untuk pembibitan ikan air tawar, pengairan tanaman, peternakan, dan penggunaan lain yang memerlukan kualitas serupa.
4. Golongan IV: Air untuk mengairi pertanaman dan keperluan lain yang memerlukan mutu yang sama.

Berdasarkan peraturan tersebut, kualitas air dalam kategori pertama dapat digunakan sebagai air baku minum, dengan parameter yang harus diperhatikan meliputi fisika, kimia, dan biologi. Dalam parameter fisika, elemen penting yang diperhatikan adalah turbiditas, padatan terlarut, warna, dan suhu. Untuk parameter kimia, yang diperhatikan adalah pH, sulfida, dan senyawa organik (seperti logam). Sedangkan dalam parameter biologi, elemen yang harus diperhatikan adalah coliform yaitu mikoorganisme yang biasa tumbuh di air yang dan dapat dijadikan patokan bahwa air tidak terkandung patogen.[8]

2.1.3 Air Baku

Air baku adalah air yang diambil dari sumber alami, seperti sungai, danau, atau sumur, yang belum mengalami proses pengolahan. Air ini berfungsi sebagai bahan dasar untuk berbagai keperluan, termasuk kebutuhan domestik, industri, dan pertanian. Kualitas air baku sangat penting untuk menentukan proses pengolahan yang diperlukan

sebelum air tersebut dapat digunakan. Beberapa parameter yang sering dianalisis dalam kualitas air baku meliputi pH, kekeruhan, kandungan bakteri, dan zat terlarut.

Pengolahan air baku biasanya diperlukan untuk menghilangkan kontaminan dan memenuhi standar kualitas air bersih. Proses pengolahan ini dapat mencakup koagulasi dan flokulasi, yang bertujuan untuk menggumpalkan partikel kecil menjadi lebih besar agar mudah diendapkan, diikuti dengan filtrasi untuk menghilangkan partikel padat melalui media filter. Disinfeksi juga merupakan langkah penting dalam pengolahan air, di mana klorin, ozon, atau sinar UV digunakan untuk membunuh mikroorganisme patogen.

Di Indonesia, standar kualitas untuk air baku diatur oleh Kementerian Kesehatan dan Badan Pengawas Obat dan Makanan (BPOM), yang menetapkan batasan-batasan tertentu yang harus dipenuhi sebelum air dapat digunakan. Air baku memiliki peran yang sangat penting dalam kehidupan sehari-hari, digunakan untuk kebutuhan domestik seperti minum, memasak, dan mandi, serta dalam industri untuk proses produksi dan pendinginan. Dalam pertanian, air baku digunakan untuk irigasi dan pemeliharaan tanaman.

Namun, pengelolaan air baku menghadapi berbagai tantangan, termasuk pencemaran dari limbah industri, pertanian, dan domestik, yang dapat mengancam kualitas sumber air. Selain itu, perubahan iklim dapat mempengaruhi ketersediaan air baku, sementara pertumbuhan

populasi meningkatkan permintaan akan air bersih, menambah tekanan pada sumber daya air yang ada. Oleh karena itu, pemahaman yang mendalam tentang kualitas, pengolahan, dan tantangan dalam pengelolaan air baku sangat penting untuk menjaga keberlanjutan sumber daya air dan memastikan ketersediaan air bersih bagi masyarakat.

Parameter fisika air meliputi beberapa aspek penting yang dapat mempengaruhi kualitas dan keamanan air. Salah satunya adalah turbiditas, yang mengukur sejauh mana air keruh akibat partikel padat yang terlarut dari zat organik maupun non-organik.[9] Tingkat turbiditas yang tinggi dapat mengindikasikan adanya kontaminan dan dapat mempengaruhi proses pengolahan air. Selain itu, padatan terlarut juga menjadi parameter penting, yang mengacu pada jumlah total zat padat yang terlarut dalam air. Padatan terlarut yang tinggi dapat mempengaruhi rasa dan kualitas air, serta berpotensi mengganggu kesehatan. Warna air juga merupakan indikator penting, di mana perubahan warna dapat menunjukkan adanya zat organik atau anorganik yang terlarut, yang dapat mempengaruhi kualitas air secara keseluruhan.

Dari segi parameter kimia air meliputi beberapa parameter seperti pH adalah tingkat keasaman atau kebasaan air.[10] pH yang ideal untuk air minum biasanya berkisar antara 6,5 hingga 8,5. pH yang tidak sesuai dapat mempengaruhi kesehatan dan rasa air. Kemudian ada sulfida

kehadiran sulfida dalam air dapat menyebabkan bau tidak sedap dan berpotensi berbahaya bagi kesehatan dan ada juga senyawa organik adalah senyawa yang mengandung unsur karbon sebagai penyusun utamanya. Senyawa organik dapat berasal dari alam atau kegiatan manusia. Logam berat dan bahan organik termasuk dalam kategori ini dan kehadiran senyawa ini harus diminmalkan untuk menjaga kualitas air.

Semua parameter diatas dapat ditemukan dalam *Keputusan Menteri Lingkungan Hidup No.115 Tahun 2003* Tentang Pedoman Penentuan Status Mutu Air, selain parameter yang telah disebutkan masih banyak parameter air yang diatur oleh pedoman diatas sehingga air yang kita olah telah sesuai dengan pedoman yang dikeluarkan oleh pemerintah.

2.1.4 Proses Dalam Pengolahan Air

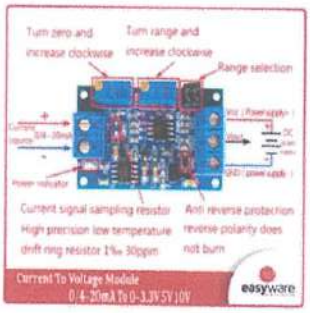
Proses pengolahan air dimulai dengan *koagulasi* dan *flokulasi*, di mana bahan kimia seperti tawas ditambahkan ke dalam air untuk menggumpalkan partikel-partikel kecil dan kotoran, membentuk flok yang lebih besar. Setelah itu, air dialirkan ke bak sedimentasi, di mana flok-flok tersebut mengendap ke dasar karena *gravitasi*, memisahkan partikel padat dari air. Selanjutnya, air melewati tahap filtrasi, di mana partikel-partikel kecil yang masih tersisa disaring menggunakan lapisan pasir, kerikil, atau media filter lainnya. Untuk memastikan air bebas dari bakteri, virus, dan *mikroorganisme* berbahaya, dilakukan proses

desinfeksi dengan menambahkan klorin, ozon, atau menggunakan sinar *ultraviolet*. Setelah itu, pengaturan pH dilakukan dengan menambahkan bahan kimia seperti kapur atau soda ash agar tingkat keasaman air sesuai dan aman untuk dikonsumsi. Terakhir, air yang telah bersih dan aman didistribusikan ke rumah-rumah atau industri melalui jaringan pipa. Proses-proses ini menjamin bahwa air yang kita gunakan sehari-hari memenuhi standar kesehatan dan layak konsumsi.

2.1.5 *Current to Voltage 0/4-20mA to 0-3.3V5V10V Signal Converter*

Dalam pemrosesan sebuah sinyal transmisi, terkadang jarak antara alat pengirim dan penerima bisa membuat voltase sinyal menurun, untuk menghindari hal tersebut maka dapat menggunakan metode mengirim arus dari pada voltase dengan adanya modul ini dapat menangani masalah tersebut. Modul ini mengubah keluaran arus dari sensor dengan rentang antara 0 – 20mA, ataupun 4 – 20mA menjadi bentuk voltase yang dapat dibaca oleh mikrokontroler atapun alat sejenisnya untuk mengolah hasil keluaran sensor tersebut, penggunanaan modul ini menjadi salah satu alternative dalam pengiriman sinyal transmisi. Keluaran voltase modul ini di rentang 0 - 3.3V, 0 – 5V, dan 0 – 10V untuk penggunaan daya sendiri modul ini menggunakan sumber daya antara 7 – 36V jika ingin keluaran voltase diatas 10V maka dapat menggunakan sumber daya bertegangan diatas 12V, input minimum tegangan dan *output* voltase pun dapat di sesuaikan dengan kebutuhan terdapat potensio untuk mengatur

kebutuhan ini dan ada fungsi jumper pin untuk memilih kebutuhan komunikasi dalam alat ini.



Gambar 2. 1. *Voltage Converter*

2.1.6 ESP32-S2

ESP32 merupakan modul *microcontroller* terintegrasi yang dikembangangkan oleh *Espressif Systems*, mikrokontroler ini dapat digunakan untuk membuat berbagai sistem aplikasi berbasis *Internet of Things* (IoT). ESP32 memiliki modul WiFi dan Bluetooth yang terintegrasi di dalamnya.Modul ini mempunyai dua prosesor komputasi, *prosesor* pertama berfungsi untuk mengelola komunikasi via jaringan bluetooth dan wifi sedangkan prosesor kedua untuk menjalankan aplikasi, prosesor ESP32 berjenis *CPU dual-core Tensilica Xtensa LX6*. Dilengkapi memori RAM yang cukup besar untuk menyimpan data. Untuk proyek kali ini menggunakan versi ESP32-S2 yang menggunakan prosesor *single-core 32-bit LX7 microprocessor up to 240 MHz*

Tabel 2. 1 Perbedaan *ESP32* dan *ESP32-S2*

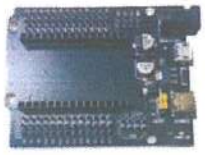
ESP32	ESP32-S2
ESP32 adalah mikrokontroler SOC berbiaya rendah dan berdaya rendah termasuk Wi-Fi &	ESP32-S2 adalah mikrokontroler berbasis Wi-Fi inti tunggal, berdaya rendah, dan sangat

Bluetooth mode ganda.	terintegrasi.
Diluncurkan pada bulan September 2016.	Diluncurkan pada September 2019.
Prosesor utama yang digunakan adalah Tensilica Xtensa LX6.	Prosesor utama yang digunakan adalah Tensilica Xtensa LX7 .
ESP32 tidak hemat energi dibandingkan dengan ESP32-S2.	ESP32-S2 lebih hemat energi dibandingkan ESP32 dalam konsumsi daya RF & CPU.
SRAM berukuran 520KB.	SRAM berukuran 320KB.
ROM berukuran 448KB.	ROM berukuran 128KB.
Cachenya 64KB	Cache-nya 8/16KB.
Bluetooth yang digunakan adalah BLE 4.2.	Tidak memiliki Bluetooth.
Tidak memiliki koprosesor ULP.	Memiliki koprosesor ULP-RISC-V ULP.
Memiliki akselerator kriptografi seperti; SHA, RNG, AES & RSA.	Memiliki akselerator kriptografi seperti; RSA, SHA, AES, HMAC, RNG, dan Tanda Tangan digital.
Memiliki dua I2S.	Ia memiliki satu I2S .
Ia memiliki tiga UART .	Memiliki dua UART.
Termasuk 34 pin GPIO.	Termasuk 43 pin GPIO.
ADC – SAR-2 12-bit dan hingga 18 saluran.	ADC – SAR-2 13-bit dan hingga 20 saluran.
Frekuensi jamnya adalah 160/240 MHz.	Frekuensi jamnya adalah 240 MHz.
Memiliki enkripsi flash boot aman OTP 1024-bit.	Memiliki enkripsi flash boot aman OTP 4096-bit.
Flash eksternal berukuran hingga 16 MB perangkat dan 11 MB alamat + 248 KB setiap waktu.	Flash eksternal berkapasitas perangkat hingga 1 GB dan alamat 11,5 MB setiap waktu.
RSA hingga 4096 bit.	RSA hingga 4096 bit dengan opsi akselerasi yang ditingkatkan dibandingkan dengan ESP32.
OTP adalah 1024-bit.	OTP adalah 4096-bit.

Gambar 2. 2. *ESP32-S2*

2.1.7 ESP32-S2 *Shield*

Menurut penelitian (Jamaludin, Sultan, and Shah 2020) ESP32 Shield adalah sebuah alat yang di gunakan untuk mempermudah penggunaan ESP32. Dengan menggunakan ESP32 Shield maka dapat memperbanyak jumlah pin yang terdapat pada ESP32 dan pada saat merancang sebuah alat tidak lagi membutuhkan sebuah papan bredboard. Shield ini juga terdapat beberapa macam pin headar, di antara nya 5 Volt, 3,3 Volt dan di lengkapi dengan Groud. Cara menggunakan shield juga cukup mudah, hanya dengan memasangkan mikrokontroler ESP32 di tempat yang sudah di sediakan di atas ESP32 Shield. Terdapat variant untuk setiap jenis ESP32 dan tidak setiap varian shield akan cocok dengan ESP32, semisal ESP32 shield tidak akan cocok dipasangkan dengan versi ESP32-S2 karena jumlah pin dan penempatan yang berbeda maka dari itu pemilihan jenis shield dan ESP sangat penting.

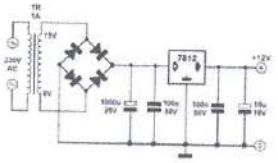


Gambar 2. 3. *ESP32-S2 Shield*

2.1.8 *DC Power Supply*

Catu daya *DC (*power supply*)* suatu rangkaian elektronik yang mengubah arus listrik bolak-balik menjadi arus listrik searah. Catu daya merupakan bagian terpenting dalam elektronika yang memiliki fungsi sebagai sumber tenaga listrik. *Power supply* adalah perangkat yang memasok energi listrik ke satu atau lebih beban listrik. *Power supply* menggunakan modul AC 220V ke DC 12V 2A, yang dilengkapi dengan *overcurrent protection*, *overload protection* dan *short circuit protection.*

Gambar 2. 4. *Power Supply DC 12V*



Gambar 2. 5. Rangkaian *Power Supply 12V*

2.1.9 Kabel Jumper

Kabel jumper merupakan kabel elektrik yang mempunyai pin konektor di setiap ujungnya dan memungkinkan untuk menghubungkan

dua komponen yang melibatkan Arduino tanpa memerlukan solder. Intinya, kegunaan kabel jumper ini digunakan sebagai konduktor listrik untuk menyambungkan rangkaian listrik.

Kabel jumper biasanya digunakan pada breadboard atau alat prototyping lainnya supaya lebih mudah untuk mengutak-atik rangkaian. Konektor yang terdapat pada ujung kabel terdiri dari konektor jantan (*male connector*) dan konektor betina (*female connector*). Konektor *female* berfungsi untuk menusuk dan konektor *male* berfungsi untuk ditusuk.

1. **Kabel Jumper *Male to Male***

Gambar 2. 6. Kabel *Jumper Male to Male*

Kabel jumper jenis ini merupakan kabel yang sangat cocok untuk yang ingin membuat rangkaian elektronik di breadboard.

2. **Kabel Jumper *Male to Female***

Gambar 2. 7. Kabel Jumper Male to Female

Kabel jenis ini mempunyai ujung konektor yang berbeda di tiap ujungnya, yaitu male dan female. Biasanya digunakan untuk menghubungkan komponen elektronika selain dari Arduino ke *breadboard.*

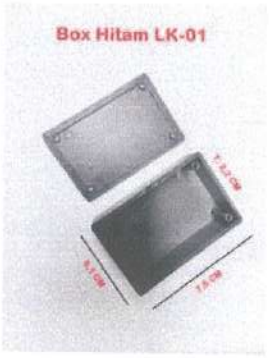
3. **Kabel Jumper *Female to Female***

Gambar 2. 8. Kabel *Jumper Female to Female*

Kabel jenis ini merupakan kabel yang sangat cocok untuk menghubungkan antar komponen yang mempunyai header male. Misalnya, sensor ultrasonik HC-SR04, sensor suhu DHT dan lain sebagainya.

2.1.10 *Box Case*

Merupakan sebuah box berwarna hitam yang memiliki fungsi sebagai sebuah case pelindung untuk komponen-komponen yang akan dipasang, ukuran box sendiri bervariatif untuk yang akan dipakai memiliki ukuran yaitu 21,5x14,5x8,5 cm, box ini sendiri tahan akan induksi arus listrik sehingga komponen-komponen didalam-nya akan aman.



Gambar 2. 9. Box Case

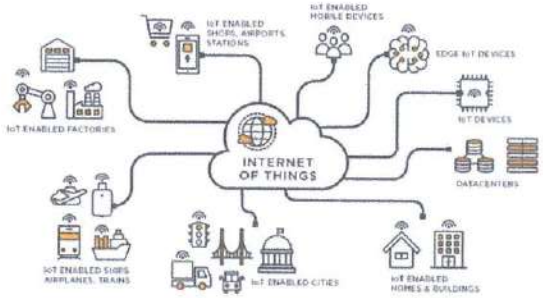
2.1.11 *IoT (Internet of Things)*

Diambil dari situs resmi *IBM* yang berjudul Apa itu *IoT*? *Internet of Things (IoT)* mengacu pada jaringan alat-alat fisik, kendaraan, perangkat, dan objek fisik lainnya yang dilengkapi dengan sensor, perangkat lunak, serta konektivitas jaringan, yang memungkinkan pengumpulan dan pertukaran data. Perangkat *IoT*, sering disebut sebagai "objek cerdas," dapat bervariasi dari alat "rumah cerdas" yang sederhana seperti termostat pintar, hingga perangkat yang dikenakan seperti jam tangan pintar dan pakaian berbasis *RFID*, serta sistem transportasi dan mesin industri yang rumit. Beberapa pengembang bahkan membayangkan "kota cerdas" yang sepenuhnya berfokus pada teknologi *IoT*. Teknologi *IoT* memungkinkan alat-alat cerdas ini untuk saling berinteraksi dan berkomunikasi dengan perangkat lain yang terhubung ke internet. Mirip dengan *smartphone* dan *gateway*, *IoT* membangun jaringan besar perangkat yang berinteraksi satu sama lain, yang mampu bertukar informasi dan melakukan berbagai fungsi secara mandiri, seperti memantau keadaan lingkungan di lahan pertanian,

mengatur pola lalu lintas dengan kendaraan cerdas dan alat otomotif pintar lainnya, mengelola mesin dan proses di pabrik, serta memantau persediaan dan pengiriman di tempat penyimpanan.

Gambar 2. 10. Skema *IoT*

2.1.12 *MQTT (Message Queue Telemetry Transport)*



Dikutip dari halaman resmi *hivemq.com* dan *mqtt.org*, *MQTT* adalah protokol pengiriman pesan yang paling umum digunakan dalam konteks *Internet of Things (IoT). MQTT* merupakan singkatan dari *MQ Telemetry Transport*. Protokol ini terdiri dari serangkaian aturan yang mengatur bagaimana perangkat *IoT* dapat menerbitkan dan berlangganan data melalui *internet. MQTT* digunakan untuk pengiriman pesan dan pertukaran data antara perangkat *IoT* dan *perangkat IoT industri (IIoT),* seperti mikrokontroler, sensor, dan PLC industri. Protokol ini beroperasi dengan menghubungkan perangkat melalui pola publikasi/berlangganan *(Pub/Sub)*. Pengirim (*Publisher*) dan penerima (*Subscriber*) berkomunikasi melalui topik yang memisahkan keduanya. Koneksi antara pengirim dan penerima dikelola oleh *broker MQTT*, yang menyaring semua pesan masuk dan mendistribusikannya dengan tepat kepada pelanggan atau alat yang terhubung.

Protokol ini awalnya dirancang oleh Andy Stanford-Clark (*IBM*) dan Arlen Nipper pada tahun 1999 untuk menghubungkan sistem telemetri pipa minyak melalui satelit. Meskipun awalnya merupakan protokol milik perusahaan, *MQTT* dirilis bebas royalti pada tahun 2010 dan diakui sebagai standar *OASIS* pada tahun 2014.

2.1.13 Versi Protokol *MQTT*

MQTT sendiri memiliki beberapa versi yang sudah berjalan sampai sekarang.

1. ***MQTT 3.1***

   Dirilis pada tahun 2010 kemudian menjadi versi pertama yang banyak diadopsi secara luas. Memperkenalkan konsep dasar seperti *publish/subscribe, topics*, dan *Quality of Service (QoS)*.

2. ***MQTT 3.1.1***

   Dirilis pada tahun 2014, memperbaiki beberapa masalah dari versi 3.1 dan menambahkan beberapa fitur baru .Menyediakan dukungan untuk *UTF-8* dalam topic names dan payload .Memperjelas beberapa aspek dari spesifikasi, termasuk pengelolaan koneksi dan pengaturan *keep-alive*.

3. ***MQTT 5.0***

   Ini adalah versi *MQTT* yang sering kita pakai sekarang, memiliki beberapa spesfikasi seperti, dirilis pada tahun 2019. Menambahkan banyak fitur baru untuk meningkatkan fungsionalitas dan fleksibilitas. Memperkenalkan User Properties, yang memungkinkan pengiriman metadata tambahan dengan pesan. Menyediakan *Reason Codes* untuk memberikan umpan balik yang

lebih jelas tentang status operasi. Memperkenalkan *Session Expiry*, yang memungkinkan pengelolaan sesi yang lebih baik. Menyediakan dukungan untuk *shared subscriptions*, yang memungkinkan beberapa pelanggan untuk berbagi langganan pada topik yang sama.

2.1.14 Perangkat Lunak *SCADA Haiwell*

SCADA (Supervisory Control and Data Acquisition) merupakan struktur sistem pengendali yang meliputi komputer, komunikasi data jaringan, dan antarmuka pengguna grafis untuk memantau mesin serta proses tingkat atas. Ini juga mencakup sensor serta perangkat lainnya, seperti pengontrol logika terprogram, yang berinteraksi dengan pabrik proses atau mesin. Antarmuka operator yang memungkinkan pemantauan dan pengeluaran perintah proses, seperti perubahan titik pengaturan pengontrol, ditangani melalui sistem komputer *SCADA*. Operasi subordinasi, seperti kontrol logika waktu nyata atau perhitungan pengendali, dilakukan oleh modul jaringan yang terhubung ke sensor lapangan dan aktuator.

Scada sendiri memiliki beberapa jenis atau versi yang sudah dipakai dalam industri seperti :

1. ***SCADA* Berbasis *PC* (*PC-Based SCADA*)**

   Sistem *SCADA* ini berjalan di komputer pribadi atau workstation. Biasanya digunakan untuk aplikasi yang lebih kecil dan tidak memerlukan infrastruktur yang kompleks. Kelebihan dari jenis

ini ialah biaya lebih rendah mudah diinstal dan dikelola. Sedangkan kekurangan-nya ialah terbatas dalam hal skalabilitas dan kemampuan pemrosesan.

2. ***SCADA* Berbasis *Web* (*Web-Based SCADA*)**

Sistem ini menggunakan teknologi *web* untuk memberikan akses ke data dan kontrol dari mana saja melalui *browser*. Ini memungkinkan pemantauan jarak jauh yang lebih fleksibel. Kelebihan jenis ini adalah aksesibilitas tinggi dari berbagai perangkat, memudahkan kolaborasi dan pemantauan jarak jauh. Sedangkan untuk kekurangan sendiri yaitu bergantung pada koneksi internet yang stabil dan memiliki potensi risiko keamanan siber yang lebih tinggi.

3. ***SCADA* Berbasis *Cloud* (*Cloud-Based SCADA*)**

Sistem *SCADA* ini *dihosting* di *cloud*, memungkinkan penyimpanan data dan pemrosesan di server jarak jauh. Ini memberikan fleksibilitas dan skalabilitas yang lebih besar.Kelebihan jenis ini adalah Skalabilitas yang tinggi dan biaya pemeliharaan yang lebih rendah.Akses data secara real-time dari lokasi mana pun. Sedangkan kekurangan-nya terdapat ketergantungan pada penyedia layanan cloud dan beberapa masalah privasi dan keamanan data.

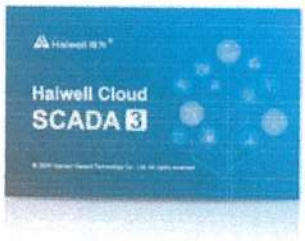
4. ***SCADA* Terdistribusi (*Distributed SCADA*)**

Sistem ini terdiri dari beberapa unit *SCADA* yang terhubung, yang memungkinkan pengelolaan dan pemantauan dari lokasi yang

berbeda. Cocok untuk aplikasi yang mencakup area geografis yang luas.Kelebihan dari system ini mampu mengelola sistem yang lebih besar dan kompleks, redundansi dan keandalan yang lebih baik. Namun dari segi kekurangan memerlukan infrastruktur yang lebih kompleks dan biaya yang lebih tinggi.

5. ***SCADA* Berbasis *PLC* (*PLC-Based SCADA*)**

Sistem ini mengintegrasikan *SCADA* dengan *Programmable Logic Controllers (PLC)* untuk kontrol otomatisasi industri. *PLC* bertindak sebagai pengumpul data dan pengendali proses. Kelebihan system ini sendiri dari segi kontrol yang lebih presisi dan responsive, integrasi yang baik dengan perangkat keras industri. Sedangkan kekurangan dari system ini memerlukan pemahaman teknis yang lebih dalam untuk pengaturan dan pemeliharaan.



Gambar 2. 11. Software *SCADA Haiwell*

2.1.15 *Android* Studio

Dikutip dari halaman resmi android developer, *Android Studio* [11]*adalah Integrated Development Environment (IDE)* resmi yang dikembangkan oleh *Google* untuk pengembangan aplikasi *Android*.

IDE ini dirancang untuk memberikan alat dan fitur yang diperlukan bagi pengembang untuk membuat aplikasi *Android* yang berkualitas tinggi. Ada beberapa fitur utama dalam *IDE* ini seperti dilengkapi dengan editor kode yang canggih, yang mendukung penyorotan sintaks, penyelesaian otomatis, dan *refactoring*[12] kode. Ini membantu pengembang menulis kode dengan lebih efisien. Menyediakan alat desain visual yang memungkinkan pengembang untuk merancang antarmuka pengguna (*UI*)[13] aplikasi dengan *drag-and-drop.*[14]

Dilengkapi dengan *emulator* yang memungkinkan pengembang untuk menguji aplikasi mereka di berbagai perangkat virtual tanpa memerlukan perangkat fisik, mendukung integrasi dengan sistem kontrol versi seperti *Git*[15], memudahkan kolaborasi tim dan manajemen kode sumber.

Mendukung pengembangan aplikasi menggunakan bahasa pemrograman *Java dan Kotlin. Kotlin*[16], yang diperkenalkan sebagai bahasa resmi untuk pengembangan *Android*, menawarkan sintaks yang lebih bersih dan fitur modern. Menyediakan akses ke berbagai *library* dan *framework*[17], seperti *Android Jetpack*, yang membantu pengembang dalam mengimplementasikan fitur-fitur umum dengan lebih cepat dan efisien. Dengan adanya *Android Studio*, pengembang dapat membuat aplikasi untuk berbagai versi *Android* dan perangkat, termasuk *smartphone, tablet, dan* perangkat *wearable*, *Android Studio* sendiri memiliki komunitas pengembang yang besar dan aktif, serta

dokumentasi resmi yang lengkap hal ini memudahkan pengembang untuk menemukan solusi dan mendapatkan bantuan saat menghadapi masalah.

Gambar 2. 12. *Android Studio*

2.1.16 Bahasa Pemrograman *Flutter*

Dikutip dari halaman codingstudio.id *Flutter*[18] adalah *cross-framework* [19] aplikasi mobile yang diciptakan oleh Google yang populer di kalangan *developer* karena penggunaannya lebih mudah.

Mobile flutter adalah *platform* yang kini sudah banyak digunakan oleh *flutter developer* untuk menciptakan aplikasi dengan desain yang menarik dengan hanya memanfaatkan satu jenis *base coding (codebase)*. Dengan begitu, aplikasi tersebut bisa diunduh dan digunakan di berbagai platform, dari *Android, iOS, website*, hingga *desktop. Flutter* sebenarnya sudah dikembangkan oleh *Google* sejak tahun 2015.

Dengan menggunakan *flutter*, *developer* bisa melakukan kostumisasi penampilan *UI* dan desain sesuai keinginan, sehingga bisa

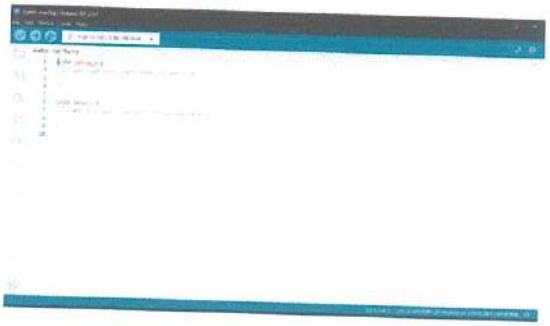
didesain dengan bagus dan unik dan berbeda dengan aplikasi mobile lain.

Gambar 2. 13. *Flutter Program*

2.1.17 *Arduino IDE*

Arduino IDE (*Integrated Development Environment*) adalah perangkat lunak yang digunakan untuk menulis, mengedit, dan mengunggah kode ke papan pengembangan *Arduino*. *IDE* ini dirancang untuk memudahkan pengembang, baik pemula maupun berpengalaman, dalam membuat proyek berbasis Arduino maupun mikrokontroler yang memiliki Bahasa pemrograman yang sama dengan *Arduino*. *IDE* ini memiliki beberapa keunggulan seperti antarmuka pengguna *Arduino IDE* dirancang agar mudah digunakan, dengan area untuk menulis kode, mengakses menu, dan melihat output dari papan *Arduino*.

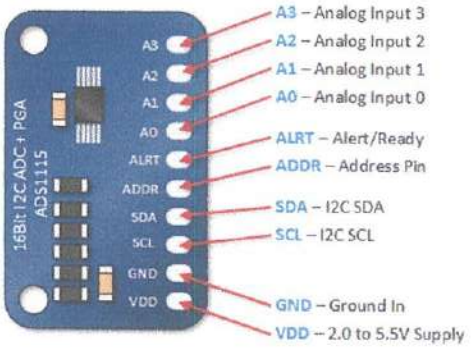
Arduino IDE menggunakan bahasa pemrograman *C/C++* dengan beberapa pustaka tambahan yang disediakan oleh Arduino. *Arduino* IDE menyediakan berbagai pustaka yang memungkinkan pengguna untuk dengan mudah mengakses dan menggunakan berbagai sensor, modul, dan perangkat keras lainnya. Serta mendukung berbagai jenis papan Arduino, termasuk Arduino Uno, Mega, Nano, dan banyak lagi sehingga pengguna dapat memilih papan yang sesuai dengan proyek mereka.



Gambar 2. 14. Arduino IDE

2.1.18 Modul *ADS1115*

Modul *ADS1115*[20] adalah sebuah konverter analog ke digital (*ADC*) dengan resolusi *16-bit* yang digunakan untuk mengubah data. sinyal analog menjadi data digital dengan presisi tinggi. Modul ini memiliki empat input analog yang bisa dipilih, dan berkomunikasi dengan mikrokontroler melalui protokol *I2C*, sehingga mudah diintegrasikan dengan berbagai perangkat seperti *Arduino* atau *Raspberry Pi*. *ADS1115* juga dilengkapi dengan *programmable gain amplifier (PGA)* yang memungkinkan penguatan sinyal kecil agar dapat diukur dengan lebih akurat. Modul ini sangat cocok digunakan untuk aplikasi yang membutuhkan pengukuran sinyal analog dengan tingkat



Gambar 2. 15. Modul *ADS1115*

2.1.19 Sensor Turbidity Arduino

Dikutip dari halaman resmi wikidfrobot,sensor kekeruhan gravitasi Arduino mendeteksi kualitas air dengan mengukur tingkat kekeruhan, atau kepekatannya. Sensor ini menggunakan cahaya untuk mendeteksi partikel tersuspensi dalam air dengan mengukur transmitansi cahaya dan laju hamburan, yang berubah seiring dengan jumlah padatan tersuspensi total (TSS) dalam air. Seiring meningkatnya TTS, tingkat kekeruhan cairan juga meningkat.

Sensor cairan ini menyediakan mode keluaran sinyal analog dan digital. Ambang batas dapat disesuaikan saat dalam mode sinyal digital.

Gambar 2. 16. Sensor *Turbidity Arduino*

BAB III
METEDOLOGI PENELITIAN

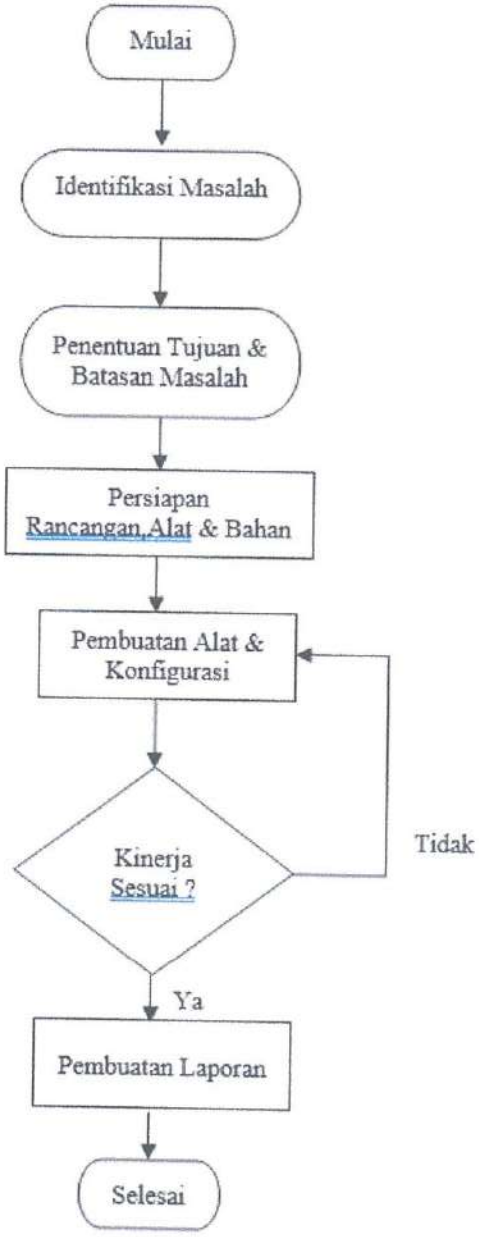
3.1.Model Penelitian

Penelitian dan Pengembangan (*R&D*) adalah proses atau tahap-tahap untuk menghasilkan produk baru atau memperbaiki produk yang sudah ada. Penelitian pengembangan adalah salah satu tipe penelitian yang bisa berfungsi sebagai penghubung atau penghubung antara penelitian dasar dan penelitian terapan.

Dari penjelasan tersebut, dapat disimpulkan bahwa Riset dan Pengembangan merupakan suatu cara penelitian yang bertujuan untuk menciptakan produk-produk tertentu. Melalui penelitian, sebuah produk akan dengan tepat mengetahui kekurangan dalam kinerja produk tersebut dan dapat segera diperbaiki untuk meningkatkan kualitas kinerja produk, sehingga pada penelitian ini metode tersebut akan digunakan untuk menguji bagaimana kinerja sensor dan kontroler dalam menangani sebuah informasi yang didapat dan diolah agar menjadi informasi berbentuk visual serta bagaimana akurasi sensor, apabila kurang memuaskan akan diperbaiki ulang untuk mencapai tingkat akurasi yang diharapkan

3.2.Prosedur Penelitian

Pada tahapan ini akan memaparkan bagaiaman penelitian berjalan dari awal perencanaan sampai penelitian berakhir dalam bentuk diagram kerja sehingga alur penelitian berjalan sesuai dengan rancangan.



Gambar 3. 1. Alur Proses Penelitian

Berikut penjelasan Langkah-langkah penelitian dan pengembangan :

1. Identifikasi Masalah

Tahapan identifikasi masalah adalah dasar dari penelitian ini dengan adanya tahapan ini permasalahan dapat ditentukan dengan tepat dengan melihat langsung ke lapangan dan melihat kondisi sekeliling. Kegiatan ini

dilakukan di PT Air Semarang Barat yang berlokasi di Jl Untung Suropati No. 01A Bambankerep, Ngaliyan Kota Semarang Provinsi Jawa Tengah.

2. Penetuan Tujuan dan Batasan Masalah

   Di tahap kedua ini masalah yang sudah terindentifikasi akan lebih dipersempit agar dapat ditentukan tujuan pembuatan penelitian dan batasan penelitian yang akan dilakukan, agar tidak keluar dari topik utama penellitian dan mencapai hasil yang diharapkan dari penelitian ini.

3. Persiapan Rancangan, Alat & Bahan

   Tahap ketiga ini adalah mempersiapkan rancangan produk, alat dan bahan yang akan digunakan, setelah tujuan dan batasan selesai ditentukan maka akan keluar keperluan alat dan bahan apasaja yang dibutuhkan untuk membuat produk dari penelitian ini dan bertujuan juga untuk efisiensi anggaran dalam pembuatan produk karena alat telah dirancang sebelumnya.

4. Pembuatan Alat Dan Konfigurasi

   Tahap keempat ini adalah membuat produk alat yang sudah dirancang dan dipersiapkan maka tahap pembuatan alat dilakukan mulai dari perakitan di sisi perangkat keras sampai pembuatan program di sisi perangkat lunak, tidak lupa juga produk dikonfigurasi sedemikian rupa agar dapat berjalan semestinya, konfigurasi dapat berupa kalibrasi produk yang akan dipakai agar sesuai dengan harapan penelitian ini.

5. Pengujian Alat

Tahapan kelima ini berupa pengujian alat di lapangan, apakah alat bekerja sesuai dengan rancangan awal atau tidak, atau perlu dikonfigurasi ulang agar alat dapat bekerja sesuai rancangan yang diharapkan, tahapan ini memakan waktu sebanyak 30 menit tiap pengujian dan pengujian dilakukan sesuai dengan keperluan agar nilai akurasi pembacaan alat yang diharapkan tercapai. Apabila hasil tidak memuaskan maka tahapan ini akan diulang, bila memuaskan akan dilanjutkan ketahapan selanjutnya.

6. Pembuatan Laporan

Tahapan ini merupakan pelaporan hasil kinerja alat selama penelitian berlangsung, bagaimana kinerja dan akurasi alat dilaporkan di tahapan ini, beserta tahapan pembuatan alat dari mulai merangkai sampai pembuatan perangkat lunak juga disertakan dalam laporan.

3.3.Teknik Pengumpulan Data

3.2.1 Metode Observasi

Pengumpulan data dilakukan dengan cara observasi dan pencatatan sehingga dapat dilakukan monitoring bagaimana kinerja alat dengan membandingkan hasil pembacaan alat dengan hasil pembacaan sensor aktual.

3.2.2 Metode Wawancara

Metode wawancara dilakukan dilakukan dengan cara menanyakan langsung kepada operator serta petugas LAB PT Air Semarang Barat untuk mendapatkan informasi dan data yang dibutuhkan sesuai dengan penelitan sistem monitoring kekeruhan air berbasis *SCADA* dan *IoT*

dengan antarmuka android menggunakan sensor *Enderss+Hauserr,* data yang dibutuhkan berupa pembacaan aktual sensor serta hasil uji LAB dihari itu untuk melihat kinerja alat.

3.2.3 Metode Literature/Studi Kepustakaan

Studi pustaka disebut juga studi literatur, kajian pustaka, tinjauan pustaka, kajian teoritis, dan tinjauan teoritis. Studi literatur menggunakan cara mencari sumber data ilmiah dan digunakan untuk menjelaskan teori-teori penelitian sebelumnya yang relevan dengan topik sistem monitoring kekeruhan air berbasis *SCADA* dan *IoT* dengan antarmuka android menggunakan sensor *Enderss+Hauserr* , sumber bisa berasal dari jurnal elektronik, perpustakaan online, dan sumber terpercaya lainnya.

3.4.Instrument Penelitian

3.4.1. Alat

Dalam membuat sistem monitoring kekeruhan air berbasis *SCADA* dan *IoT* dengan antarmuka android menggunakan sensor *Enderss+Hauserr* dibutuhkan beberapa alat diantaranya sebagai berikut:

1. Solder
2. Obeng
3. Tang potong khusus kabel
4. Timah
5. Pasta Flux
6. Box Hitam sebagai wadah
7. Multimeter

8. PC untuk membuat program
9. Kabel *USB type-C*

3.4.2. Bahan

Pembuatan sistem monitoring kekeruhan air berbasis *SCADA* dan *IoT* dengan antarmuka *android* menggunakan sensor *Endress and Hauserr* menggunakan bahan-bahan sebagai berikut :

1. *ESP 32S2*

   Sebagai mikrokontroler yang memiliki konektivitas *WiFi* dan sebagai perangkat pemrosesan utama dari output sensor dan akan menerjemahkan baik ke smartphone maupun *SCADA* hasill pembacaan sensor.

2. *ESP32 Shield*

   Papan konektivitas sebagai tempat dimana sensor dan *EPS32* dirangkai dan saling berbagi sumber tegangan, papan ini menyediakan terminal *power* serta inputan ke *ESP32* yang lebih lengkap daripada hanya menggunakan terminal bawaan dari *ESP32*.

3. *Power Supply*

   Sebagai catu daya utama baik untuk *ESP32* maupun sensor agar keduanya dapat bekerja dengan baik.

4. Kabel *Jumper male to female*

   Kabel ini difungsikan untuk mengkoneksikan antara sensor dan *ESP32*, untuk kearah *ESP32* menggunakan terminal female, sedangkan arah sensor menggunakan terminal male

5. *Current to Voltage 0/4-20mA to 0-3.3V5V10V Signal Converter*

Sebagai sensor utama dalam menjalankan fungsinya yaitu mengubah arus menjadi voltase agar dapat dibaca oleh *ESP32*, voltase yang dikeluarkan sendiri memiliki nilai tegangan sebesar 3.3V.

6. Modul *ADS1115*

Adalah modul analog to digital atau *ADC* yang berfungsi mengubah sinyal analog dari sensor ke bentuk digital yang siap diproses oleh *ESP32*, modul ini bekerja menggunakan jalur komunikasi *SCL* dan *SDA* pada *ESP32*.

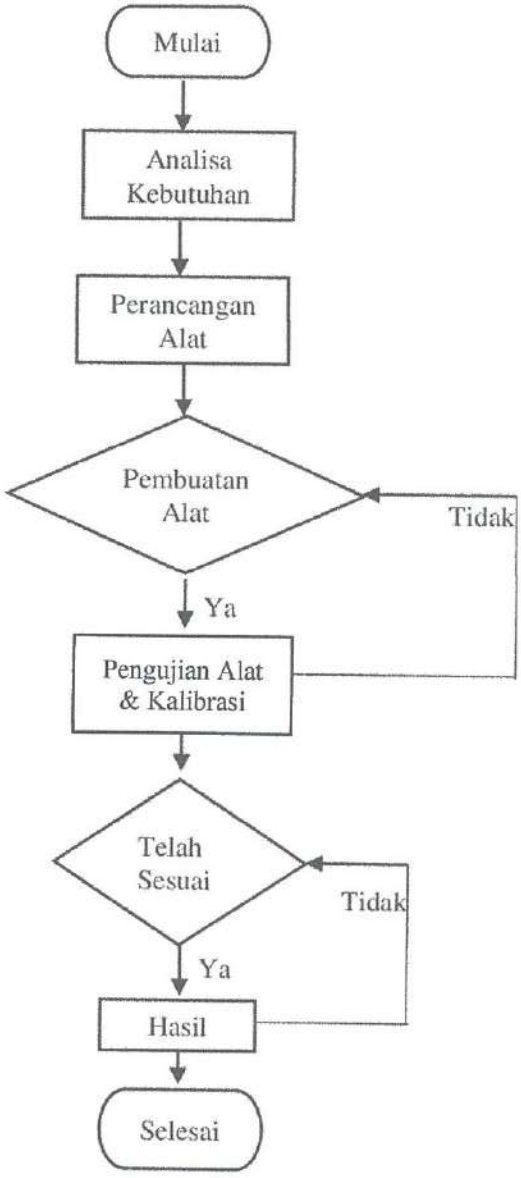
Selain bahan-bahan diatas berikut adalah table keperluan untuk perangkat lunak untuk mengembangkan alat tersebut :

Tabel 3.1. Kebutuhan Perangkat Lunak

No	Perangkat Lunak	Fungsi
1	Arduino IDE	Program Alat
2	Perangkat Lunak *Haiwell Cloud Scada*	Program Dan Desain *SCADA*
3	Perangkat Lunak *Android Studio*	Desain Antarmuka *Android*
4	Perangkat Lunak *Flutter*	Bahasa Program Untuk *Android*

3.5.Tahap Perancangan Alat

Tahapan perancangan alat ini dibagi menjadi dua bagian yaitu pertama dari perancangan sisi perangkat keras kemudian perancangan sisi perangkat lunak dan setiap sisi akan diwakili oleh flowchart bagaimana cara kerja-nya.



Gambar 3. 2. *Flowchart* Perancangan Alat

Dari tahapan diatas dapat dijelaskan sebagai berikut :

1. Analisa kebutuhan

   Merencanakan kebutuhan alat sesuai dengan hasil identifikasi dan batasan yang telah didapat.

2. Perancangan Alat

Mulai merancang alat sesuai dengan desain dan kebutuhan dilapangan dengan tetap memperhitungkan keefektifan produk, efisiensi dan akurasi produk sesuai dengan tujuan awal.

3. Pembuatan Alat

Pembuatan alat beserta perangkat lunak akan dirangkai dan diprogram sesuai dengan rancangan awal untuk mengumpulkan data dan tahapan pengujian dilapangan, apabila dirasa dalam pembuatan masih kurang maka akan kembali ke tahapan perancangan alat.

4. Pengujian Alat

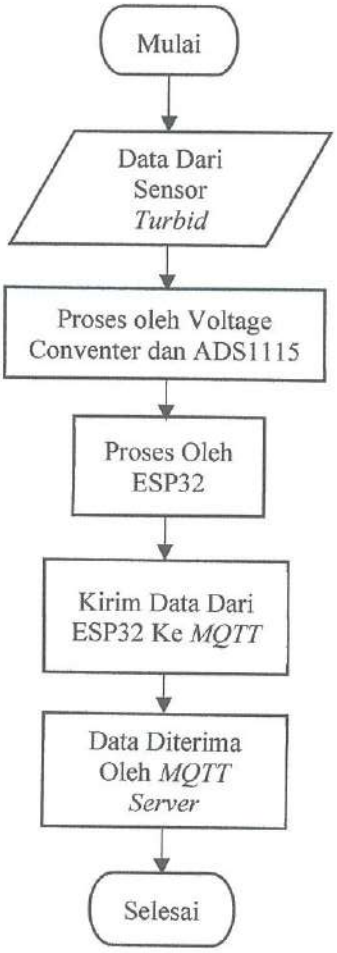
Setelah alat dibuat dan perangkat lunak selesai diprogram maka tahapan selanjutnya adalah pengujian, yang akan menentukan hasil alat sudah sesuai dengan tujuan, kriteria, dan akurasi yang diinginkan atau tidak, pengujian ini memakan waktu 30 menit tiap tahapan pengujiannya.

5. Hasil

Hasil pengujian berupa data dan akan dievaluasi kembali apakah alat sudah siap digunakan atau perlu perbaikan dalam sisi perangkat keras maupun perangkat lunak-nya.

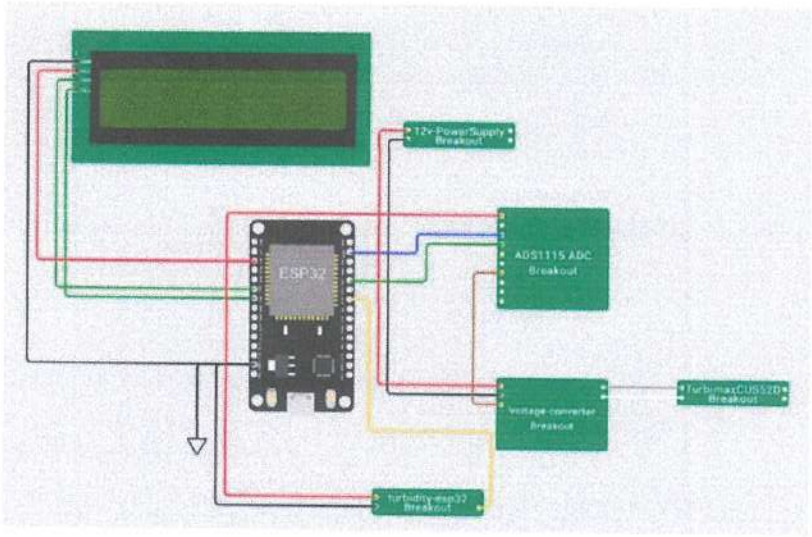
3.5.1 Perakitan Alat

Pada tahapan ini alat akan digambarkan bagaimana sistem kerja serta seperti apa pengkabelan tiap-tiap alat yang terhubung dengan mikrokontroler serta bagaimana alat akan berkomunikasi dengan *server MQTT.*



Gambar 3. 3. Alur Proses Kerja Alat

Gambar diatas adalah alur proses kerja alat mulai dari data yang terkirim oleh sensor turbidity kemudian diolah oleh *Voltage Conventer* dan modul *ADS1115* ESP32 yang telah terhubung ke internet dan kemudian data tersebut akan diterima oleh *MQTT server* yang sudah dikonfigurasi sebelumnya.

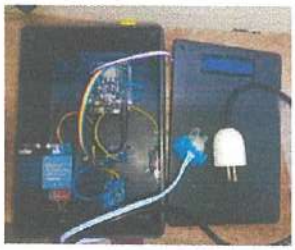


Gambar 3. 4. *Wiring* Diagram Sensor dan Alat

Dari gambar 3.3 dapat dilihat bagamana pengkabelan sensor beserta *ESP32* yang saling terhubung ke pin *I/O ESP32,* Terdapat sebuah *power supply 12 VDC* yang digunakan untuk sumber tegangan *ESP32* maupun sensor.

Tabel 3.2 Konfigurasi pin ESP32

Komponen	PIN	Jalur ESP32
PSU 12 VDC	+	Pin VCC ESP32
	-	Pin GND ESP32
Voltage Con-verter	VCC	Pin VCC PSU 12VDC
	GND	Pin GND PSU 12VDC
	Vout	Pin pin A0 ADS1115
ADS1115	VCC	Pin VCC ESP32
	GND	Pin GND ESP32
	SCL	Pin SCL ESP32
	SDA	Pin SDA ESP32

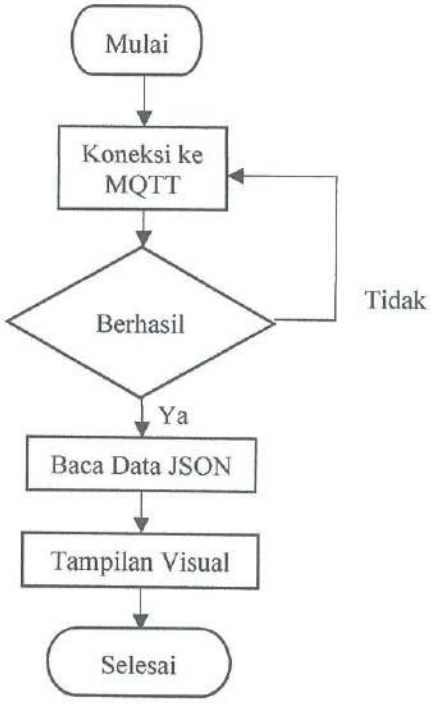


Gambar 3. 5. *Wiring* Pada Alat

Gambar 3.5 menunjukan hasil jadi produk yang telah dirakit sesuai dengan perencanaan awal dan produk siap untuk diuji dilapangan dari fungsi sampai akurasi.

3.5.2 Pemograman *Android* Dan *SCADA Haiwell*

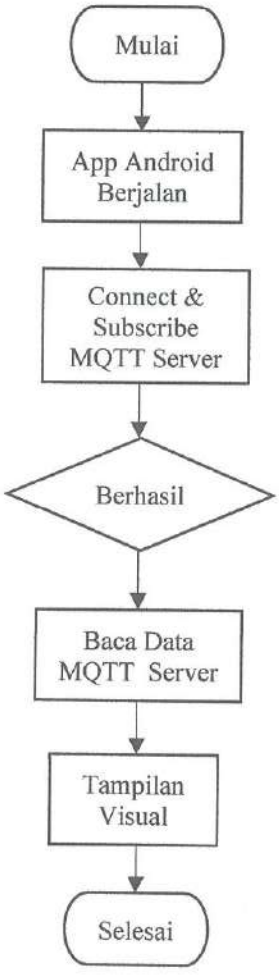
Pada tahapan ini akan dijelaskan langkah kerja dari antra muka *Android* dan *SCADA Haiwell* dalam bentuk flowchart beserta penjelasannya.



Gambar 3. 6. *Flowchart SCADA Haiwell*

Dari *flowchart* diatas dapat dilihat dari alur pertama yaitu *SCADA* akan mencoba untuk membuat koneksi ke *MQTT* server, kemudian bila berhasil maka akan membaca data yang tersimpan pada topik yang sudah disetting sebelumnya yaitu "data/turbid526/ntu/4456", topik yang dibaca berformat *JSON* sesuai dengan konfigurasi yang ada pada *SCADA Haiwell.*

Bila data telah terbaca dengan benar maka kemudian data akan diolah menjadi tampilan visual yang lebih mudah dibaca.



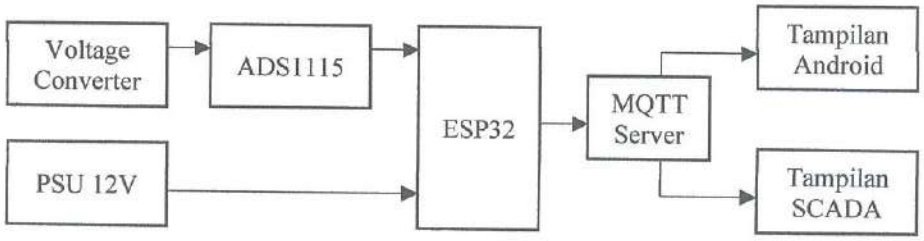
Gambar 3. 7. Flowchart Aplikasi Android

Gambar 3.7 adalah flowchart alur kerja dari aplikasi android mulai dari aplikasi berjalan kemudian akan mencoba menyambung ke mqtt server dan subscribe topik yang sudah disetting, apabila gagal proses akan diulang sampai berhasil setelah berhasi maka aplikasi akan membaca data yang tersedia di server dan akan menampilkan hasil informasi yang berada di mqtt server dalam tampilan visual yang mudah dibaca, setiap aplikasi berjalan maka kita sebagai user harus memastikan bahwa aplikasi sudah terkoneksi dengan benar ke mqtt server dan subscribe dengan topik yang sama yaitu "data/turbid526/ntu/4456"

BAB IV
PEMBAHASAN

4.1.Hasil Penelitian

Setelah semua tahap perancangan dan perencanaan selesai, tahapan selanjutnya adalah pengerjaan alat dan implementasi sistem monitoring kekeruhan air berbasis *IoT*. Sistem ini menggunakan sensor *turbidity* (kekeruhan) tipe *CUS52D* dari merk *Endress and Hauserr* yang berfungsi untuk mendeteksi tingkat kekeruhan air secara *real-time*. Data yang diperoleh dari sensor akan diproses oleh mikrokontroler ESP32S2 dan kemudian dikirimkan ke aplikasi *Android* serta sistem *SCADA Haiwell* untuk monitoring dan kontrol.

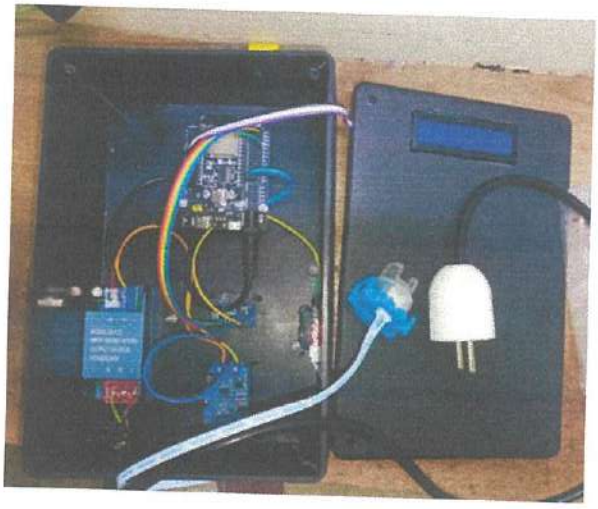


Gambar 4. 1. Diagram Blok Sistem

4.1.1Pembuatan Alat

Alat yang dibuat terdiri dari beberapa komponen utama, yaitu sensor kekeruhan *EH CUS52D*, mikrokontroler *ESP32S2*, modul komunikasi *Wi-Fi*, serta perangkat lunak pendukung berupa aplikasi *Android* dan *SCADA* Haiwell. Sensor *EH CUS52D* bekerja dengan prinsip pengukuran intensitas cahaya yang diteruskan atau tersebar oleh partikel dalam air, sehingga dapat menghasilkan nilai kekeruhan dalam

satuan *NTU* (*Nephelometric Turbidity Unit*).Data diterima dan akan dikonversi menjadi nilai voltase dengan modul *Voltage Converter* kemudian voltase akan diubah menjadi nilai *ADS* oleh modul *ADS1115* dan hasilnya akan diproses oleh Mikrokontroler *ESP32S2*, setelah data berhasil diubah maka kemudian data akan dikirim ke server *MQTT* dan dari server *MQTT* juga tugas aplikasi *android* dan *SCADA Haiwell* akan menampilkan data tersebut dalam bentuk visual.



Gambar 4. 2. Hasil Pembuatan Alat

Gambar diatas merupakan hasil perangkaian alat yang sudah siap digunakan dengan menggunakan pin-pin pada ESP32S2 sebagai berikut:

1. Pin VCC pada ADS1115 tersambung ke Pin 5V pada ESP32
2. Pin GND pada ADS1115 tersambung pada pin GND di ESP32
3. Pin SCL pada ADS1115 tersambung ke pin SCL pada ESP32
4. Pin SDA pada ADS1115 tersambung ke pin SDA pada ESP32
5. Pin VCC pada Converter tersambung ke 12V pada board

6. Pin GND pada Converter tersambung ke GND pada board
7. Pin Vout dari *Converter* masuk ke dalam pin A0 pada ADS1115
8. Pada *socket* A dan B di *Converter* akan tersambung ke kabel *Analog Sensor*
9. Pin VCC sensor *turbidity* masuk ke 5V pada ESP32
10. Pin GND sensor *turbidity* masuk ke GND pada ESP32
11. Pin Vout sensor turbidity masuk ke A1 pada modul ADS1115
12. Pin VCC LCD masuk ke 5V pada ESP32
13. Pin GND LCD masuk ke GND pada ESP32
14. Pin SCL LCD masuk ke pin 32 pada ESP32
15. Pin SDA LCD masuk ke pin 31 pada ESP32

4.1.2 Pembuatan Program ESP32S2

Program pada ESP32S2 dikembangkan menggunakan *Arduino IDE* dengan bahasa pemrograman *C/C++*. Program ini mengatur pembacaan data sensor *EH*, pengolahan sinyal, dan pengiriman data melalui protokol *MQTT* atau *HTTP* ke platform *IoT*. Program juga mengatur koneksi *Wi-Fi* dan penanganan error agar sistem dapat berjalan stabil dan data dapat dikirim secara kontinu. Dengan alur kerja data dari sensor akan diproses oleh *voltage converter* kemudian diproses juga oleh modul *ADS1115* dan diolah oleh ESP32, setelah diolah maka data akan dikirim ke server *MQTT*, server sendiri menggunakan *broker* dari *EMQX* dengan alamat *broker.emqx.io*, *port*

1883, dan *topic*" data/turbid526/ntu/4455", data yang telah dikirim akan siap dibaca oleh aplikasi *android* dan juga *SCADA Haiwell*.

Revisi-UAS.ino

```
#include <WiFi.h>
#include <WebServer.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <Adafruit_ADS1X15.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Pin I2C Configuration
#define SDA_ADS 21      // Pin SDA untuk ADS1115 (standar)
#define SCL_ADS 22      // Pin SCL untuk ADS1115 (standar)
#define SDA_LCD 33      // Pin SDA untuk LCD
#define SCL_LCD 32      // Pin SCL untuk LCD

// WiFi dan MQTT Configuration
const char *ssid = "TselhomeHAP";
const char *password = "Embul@8888";
const char *mqtt_broker = "broker.emqx.io";
const int mqtt_port = 1883;
const char *mqtt_username = "emqx";
const char *mqtt_password = "public";

// MQTT Topics
const char *topic_ntu = "data/turbid526/ntu/4456";

// Sensor Configuration
const int16_t ZERO_ADC_LOW = 1560;      // Nilai ADC saat 0 NTU (batas bawah)
const int16_t ZERO_ADC_HIGH = 1580;     // Nilai ADC saat 0 NTU (batas atas)
const float ZERO_VOLTAGE = 0.1975;      // Tegangan saat 0 NTU (dihitung dari ZERO_ADC_HIGH)
const float MAX_VOLTAGE = 4.09;         // Tegangan maksimal sensor
const float maxNTU = 388.0;             // Skala maksimal NTU
const float ADC_LSB = 0.125;            // 4.096V / 32767 (untuk gain GAIN_ONE)
const float CAL_VOLTAGE = 0.5;
const float CAL_NTU = 6.4;
```

Gambar 4. 3. Pemrograman *ESP32*

```
564  void readsensor() {
565      // Baca nilai ADC
566      setI2CPins(SDA_ADS, SCL_ADS); // Set I2C pins for ADS1115
567      currentADC0 = ads.readADC_SingleEnded(0);
568      Serial.print("Raw ADC Value: ");
569      Serial.println(currentADC0);
570
571      // Hitung tegangan aktual
572      currentVoltage0 = currentADC0 * ADC_LSB / 1000;  // Konversi ke volt
573      Serial.print("Voltage 1 : ");
574      Serial.print(currentVoltage1, 4);
575      Serial.println(" V");
576
577      // Kalibrasi nilai NTU
578      if (currentADC0 >= ZERO_ADC_LOW && currentADC0 <= ZERO_ADC_HIGH) {
579          ntu1 = 0.0;  // Dianggap 0 NTU dalam range ini
580      }else if (currentADC0 < ZERO_ADC_LOW) {
581          // Jika nilai di bawah range zero
582          ntu1 = 0.0;
583          Serial.println("Warning: ADC value below zero point!");
584      }else if (currentVoltage0 <= CAL_VOLTAGE ){
585          ntu1 = (currentVoltage0 / CAL_VOLTAGE) * CAL_NTU;
586      }
587      else {
588          // Hitung tegangan terkoreksi (dikurangi offset)
589          float correctedVoltage = currentVoltage0 - ZERO_VOLTAGE - CAL_VOLTAGE ;
590          float maxCorrectedVoltage = MAX_VOLTAGE - ZERO_VOLTAGE - CAL_VOLTAGE ;
591
592          // Hitung NTU (proporsional terhadap tegangan)
593          ntu1 = CAL_NTU + (correctedVoltage / maxCorrectedVoltage) * (maxNTU - CAL_NTU);
594
595          // Batasi nilai antara 0-maxNTU
596          ntu1 = constrain(ntu1, 0.0, maxNTU);
597      }
598
```

Gambar 4. 4. Program Untuk Membaca Sensor

```
773  void loop() {
774      // Handle web server
775      if (wifiConnected) {
776          server.handleClient();
777      }
778
779      // Read sensor every 500ms
780      if (millis() - lastSensorRead >= 500) {
781          readsensor();
782          lastSensorRead = millis();
783      }
784
785      // Update LCD display
786      updateLCD();
787
788      // Publish to MQTT every 500ms
789      if (millis() - lastPublish >= 500) {
790          if (mqttConnected) {
791              publishSensorDataJSON(topic_ntu, ntu1);
792              publishSensorData(topic_ntu, ntu1);
793          }
794          lastPublish = millis();
795      }
796
797      // Check WiFi connection
798      if (WiFi.status() != WL_CONNECTED) {
799          wifiConnected = false;
800          mqttConnected = false;
801      } else {
802          wifiConnected = true;
803
804          // Try to reconnect MQTT if disconnected
805          if (!mqtt_client.connected()) {
806              mqttConnected = false;
```

Gambar 4. 5. *Void Loop* Untuk Sensor

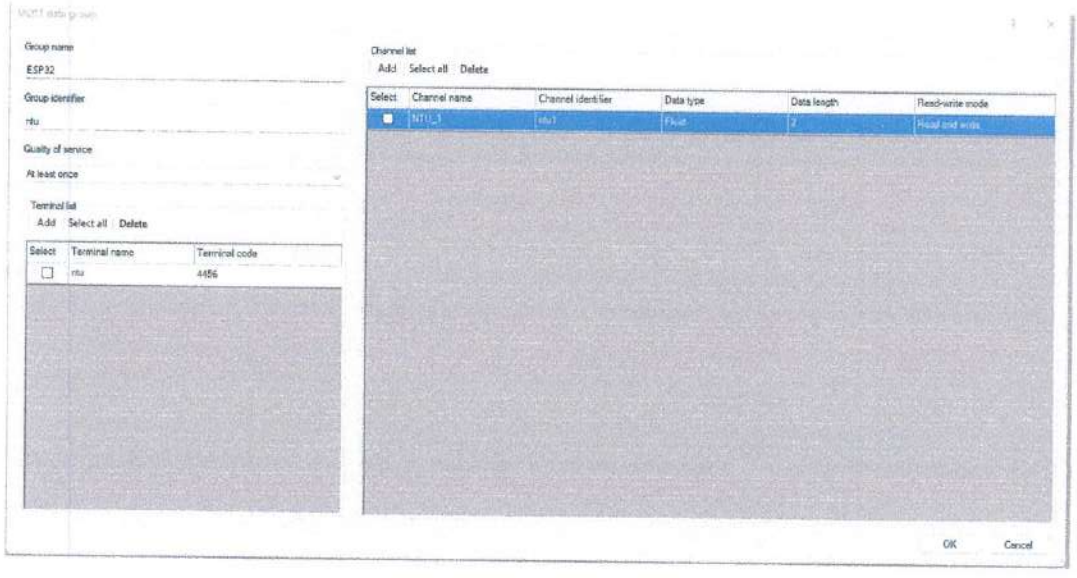
4.1.3Program Dan Desain Tampilan *SCADA Haiwell*

SCADA Haiwell digunakan sebagai *platform* monitoring dan kontrol berbasis *PC* atau *cloud*. Program *SCADA* dikonfigurasi untuk menerima data dari ESP32S2 melalui protokol *Modbus TCP/IP* atau *MQTT*. Desain tampilan *SCADA* menampilkan grafik real-time nilai kekeruhan, status sensor, dan alarm jika nilai kekeruhan melewati batas ambang yang telah ditentukan. Dashboard *SCADA* dirancang *user-friendly* dengan visualisasi data yang jelas dan mudah dipahami oleh operator.Langkah awal dalam pembuatan *SCADA Haiwell* tentunya kita perlu menyiapkan *software* tersebut, setelah *terinstal* maka dapat mengikuti langkah-langkah dibawah ini:

1. Tahapan pertama kita perlu device mqtt yang akan terkoneksi dengan mqtt broker yang akan kita gunakan dengan konfigurasi alamat *broker* adalah *broker.emqx.io*, *port* 1883, dan *topic* "*turbid526*"

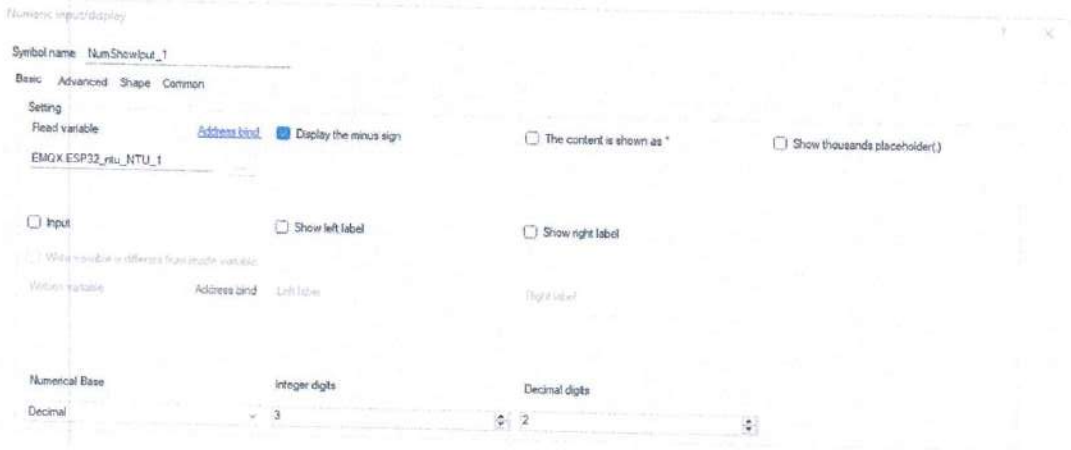
Gambar 4. 6. *Setting MQTT Device*

2. Selanjutnya adalah membuat *external variable* yang akan membaca data dari *mqtt server* yang dengan keterangan *group name* adalah ESP32 *group identifier* adalah *ntu* dan kita akan memberikan fungsi terminal yang akan menjadi kode akses ke *SCADA* kita dengan nama terminal *ntu* dan kode aksesnya adalah 4456. Setelah selesai maka kita akan membuat juga variable yang akan dapat dibaca oleh *SCADA* dari data *mqtt* dengan membuat *channel list* dengan nama channel *NTU_1*, *channel identifier* adalah *ntu1*, dan tipe data *float*. Dari penjelasan diatas maka didapatkan *topic* dengan standar dari *haiwell* sendiri yaitu "*data/turbid526/ntu/4456*" , dengan penjelasan data adalah nama awalan bawaan dari *haiwell* sendiri, *turbid526* adalah *topic* utama yang kita buat,*ntu* adalah nama terminal yang kita buat dan 4456 adalah kode terminalnya.

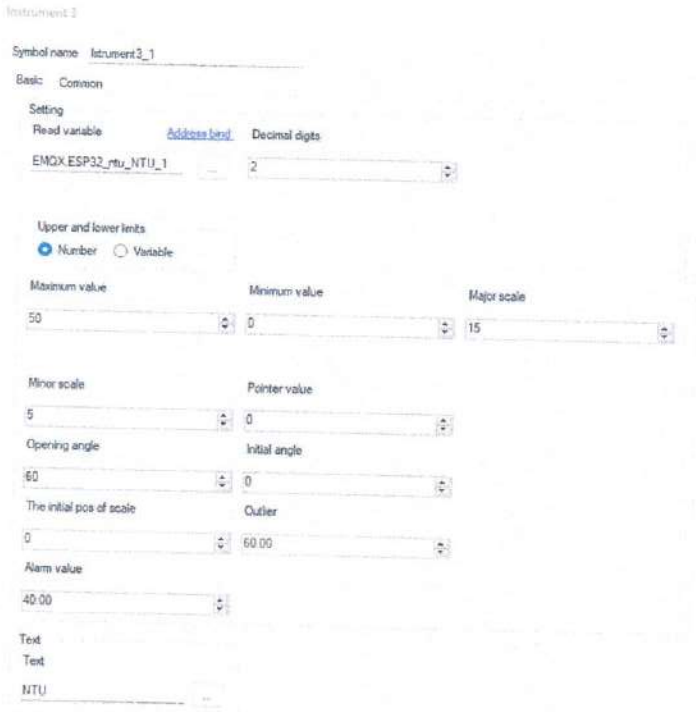


Gambar 4. 7. *Setting External Variable*

3. Selanjutnya adalah membuat tampilan dari *SCADA* itu sendiri, pada menu *display* kita akan membuat tampilannya, menggunakan *widget instrument* dan *numeric display* yang berfungsi sebagai alat visualiasai dalam menyajikan data. Dalam konfigurasi nya sendiri baik *numeric display* maupun *widget instrument* mengambil data dari *external variable* yang sudah kita buat tadi. Pada konfigurasi *numeric display* menggunakan *integer digits* 3 dan *decimal digits* 2, sedangkan pada *widget instrument* menggunakan *decimal digits* 2, karena berupa *gauge widget* maka ketika membaca data maka jarum akan naik mengikuti data yang terbaca.



Gambar 4. 8. Konfigurasi Numeric Display



Gambar 4. 9. Konfigurasi *Widget Instrument*

Berikut adalah hasil tampilan display yang sudah dibuat dengan konfigurasi seperti diatas.

Gambar 4. 10. Hasil Pembuatan *Display*

4.1.4Program Dan Desain Tampilan Aplikasi Android

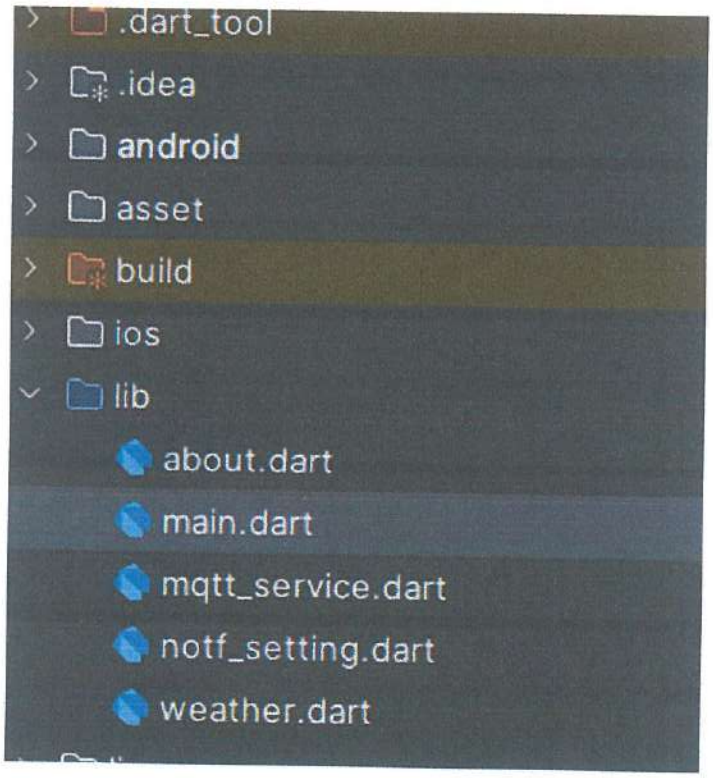
Aplikasi *Android* dikembangkan untuk memberikan kemudahan monitoring secara *mobile*. Aplikasi ini menampilkan data kekeruhan secara *real-time*, grafik tren kekeruhan, serta notifikasi jika terjadi perubahan signifikan pada kualitas air. Desain aplikasi dibuat sederhana dan responsif agar pengguna dapat dengan mudah mengakses informasi kapan saja dan di mana saja. Data aplikasi diperoleh dari *server cloud* yang menerima data dari ESP32S2.Dalam pengembangannya menggunakan 2 pemograman utama yaitu *android studio* dan bahasa pemograman *flutter*, bahasa ini dipilih karena popular, lebih ringkas dan banyak *tools* yang disediakan dalam penelitian ini 2 program diatas sudah terinstal dan dikonfigurasi agar dapat dipakai. Langkah-langkah pembuatannya sebagai berikut:

1. Persiapkan library yang akan kita gunakan dan sudah terinstall dalam program *flutter* yaitu, *mqtt_client: ^10.5.1, font_awesome_flutter: ^10.8.0 ,shared_preferences: ^2.5.2, flutter_local_notifications: ^18.0.1, http: ^0.13.5, flutter_launcher_icons: ^0.14.3, flutter_native_splash: ^2.4.4, syncfusion_flutter_charts: ^28.2.9* , semua library ditulis dalam file *pubspec.yaml* yang berada di folder utama proyek pembuatan *android* dengan cara mengetikan perintah *flutter pub_get* di *terminal* maka semua *library* akan otomasti terinstal pada proyek pembuatan *android* kita.

```
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.8
  get_time_ago: ^2.3.0
  syncfusion_flutter_charts: ^28.2.9
  intl: ^0.20.2
  flutter_launcher_icons: ^0.14.3
  flutter_native_splash: ^2.4.4

flutter_launcher_icons:
  android: "launcher_icon"
  ios: true
  image_path: "asset/images/icon2.png"
  min_sdk_android: 24
  adaptive_icon_background: "#ffffff"
  adaptive_icon_foreground: "asset/images/icon2.png"

flutter_native_splash:
  color: "#eba134"
  image: asset/images/emblem-logo.png
  branding: asset/images/branding-.png
  color_dark: "#121212"
  image_dark: asset/images/emblem-logo.png
  branding_dark: asset/images/branding-.png

  web: false

dev_dependencies:
  flutter_test:
    sdk: flutter
  mqtt_client: ^10.5.1
  font_awesome_flutter: ^10.8.0
  shared_preferences: ^2.5.2
  flutter_local_notifications: ^18.0.1
  http: ^0.13.5
```

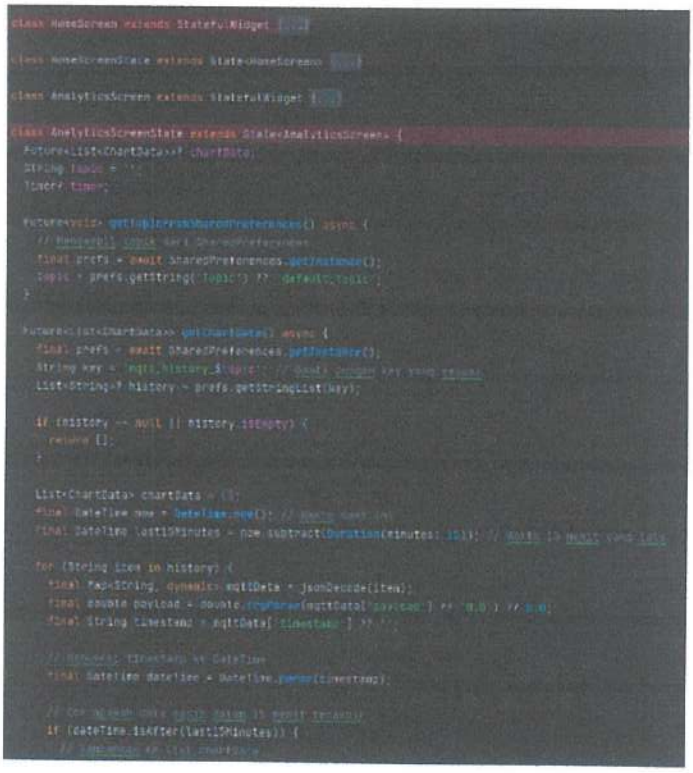
Gambar 4. 11. *Install Library Flutter*

2. Setelah terinstal semua ada 4 file utama dalam pembuatan halaman antarmuka android *yaitu main.dart, about.dart* sebagai file untuk tampilan visual dan *mqtt_service.dart, notf_setting.dart, weather.dart* sebagai file konfigurasi untuk koneksi ke *mqtt*, notifikasi, dan ramalan cuaca hari ini. Dalam tampilan visual di *main.dart* menggunakan tampilan *bottom nav* dan didalam file *main.dart* sendiri menyimpan konfigurasi dari tampilan halaman utama, *trend chart*, dan konfigurasi *mqtt server*.

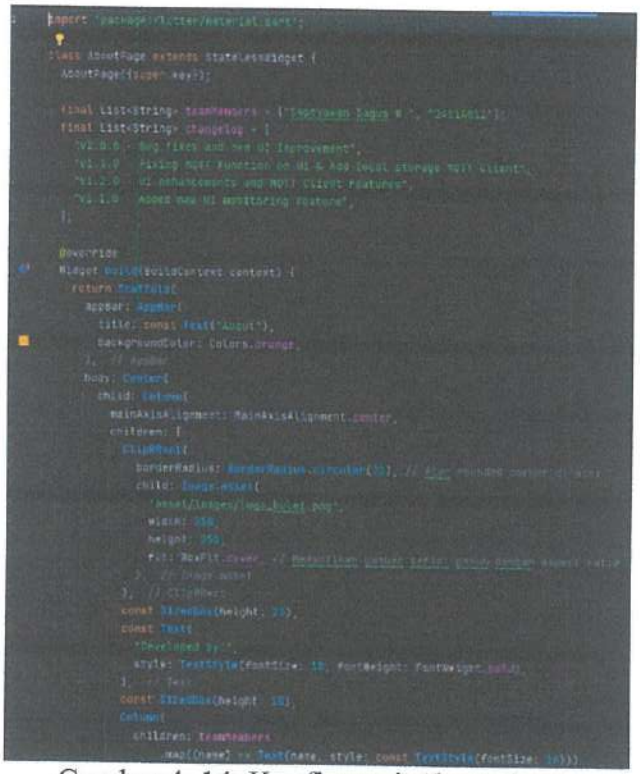


Gambar 4. 12. File Utama *Android*

3. Konfigurasi *main.dart* berisi *class-class* berikut yaitu *HomeScreen Class,AnalyticsScreen,SettingsScreen,MqttPage,MqttSettingsScreen* untuk *about.dart* berisi tentang versi aplikasi *android* serta pembuat aplikasi, untuk *mqtt_service.dart* berisi konfigurasi untuk koneksi ke *mqtt* , untuk *notf_service.dart* berisi konfigurasi untuk *pop_up* notifikasi, serta untuk *weather.dart* berisi konfigurasi untuk mengambil ramalan cuaca hari ini.



Gambar 4. 13. Konfigurasi *Main.dart*



Gambar 4. 14. Konfigurasi *About.dart*

Gambar 4. 15. Konfigurasi *mqtt_service.dart*

Gambar 4. 16. Konfigurasi *notf_service.dart*

```dart
import 'dart:convert';
import 'package:http/http.dart' as http;

class WeatherService {
  final String apiKey = '8ede7f8b710ac896e472a47f83700ac9'; // Ganti dengan API key Anda
  final String city = 'Semarang'; // Ganti dengan kota yang diinginkan

  Future<Map<String, dynamic>> getWeather() async {
    final url = Uri.parse(
      'https://api.openweathermap.org/data/2.5/weather?q=$city&appid=$apiKey&units=metric',
    );

    try {
      final response = await http.get(url);
      if (response.statusCode == 200) {
        return jsonDecode(response.body);
      } else {
        throw Exception('Failed to load weather data');
      }
    } catch (e) {
      print('Error fetching weather data: $e');
      throw Exception('Failed to load weather data');
    }
  }
}
```

Gambar 4. 17. Konfigurasi *weather.dart*

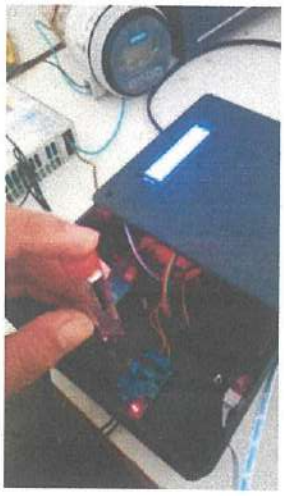
4. Setelah semua file selesai dibuat dapat menggunakan fungsi compile pada android studio dengan menggunakan menu build, kemudian flutter, dan pilih jenis APK.

Gambar 4. 18. *Build APK Flutter*

5. Aplikasi Selesai dibuat dan siap untuk diuji coba, untuk kode lengkap berada pada halaman lampiran kode *android studio*.

4.1.5 Kalibrasi Sensor Dan ESP32

Setelah semua selesai dibuat maka tahapan selanjutnya adalah kalibrasi alat baik sensor pembacaan maupun dari program sendiri, mengingat bahwa tahapan ini penting maka sebelum dilakukan pengujian dilapangan maka sebaiknya dikalibrasi dulu, dalam tahapan ini nilai kalibrasi beracuan pada nilai yang terbaca oleh *transmitter* karena kita ingin mengirimkan hasil pembacaan dari transmitter sensor ke *mqtt* maka ini menjadi point yang penting. Yang perlu disesuaikan adalah dari keluaran sensor *voltage converter* karena menggunakan modul *ADS1115* disini maka tegangan maksimal di set pada 4.096V sehingga nilai *ADC* yang akan terbaca di ESP32 adalah 32767, selanjutnya nilai *ADC* akan diproses oleh ESP32, dalam kalibrasi ini kita dapat mengetahui nilai turbiditas dari transmitter adalah 2,8 *NTU* dengan nilai voltase yang sudah dikonversi dari *ADC* adalah 0,8230V sehingga nilai ini dapat menjadi acuan dalam pemrograman kita dan kita hanya perlu menyesuaikan nilai yang terbaca *ADC* dengan nilai yang diolah oleh ESP32 sehingga sama dengan sensor.



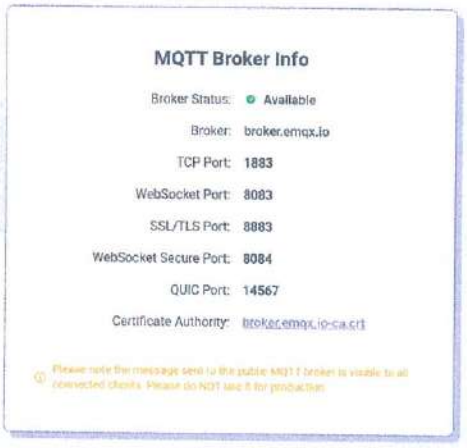
Gambar 4. 19. Kalibrasi *Voltage Converter*

Gambar 4. 20. Kalibrasi Dengan Pembacaan *ADC*

4.1.6 Konfigurasi MQTT Server (IoT)

IoT Server atau *MQTT server* yang digunakan kali adalah broker dari *EMQX*, dapat digunakan secara gratis namun data yang dikirim dapat dilihat oleh semua pihak, ada beberapa pilihan selain gratis kita dapat menggunakan akun *MQTT server* sendiri dengan mendaftar ke *EMQX* sendiri dan diberikan gratis untuk menampung data selama masa uji coba, karena kita menggunakan *EMQX* broker maka dapat menggunakan alamat

broker.emqx.io dan *port default*-nya adalah 1883 dengan *topic* yang sudah kita buat di rancangan sebelumnya, Ketika kita membuat koneksi ke *broker* maka secara otomatis *topic* akan dibuat dan dapat kita gunakan tanpa settingan lanjutan.



Gambar 4. 21. *Broker EMQX*

4.2 Hasil Analisis Penelitian

4.2.1 Hasil Pengujian Sistem

Setelah alat dan program selesai dibuat maka tahapan selanjutnya adalah pengujian pada tiap komponen yang telah dibuat dan akan dievaluasi juga hasil dari pengujian dilapangan atas hasil kinerja alat dan program yang sudah dibuat .

4.2.2 Pengujian Tiap Komponen Dalam Sistem

A. Pengujian Pembacaan dan Kalibrasi Sensor

Pengujian pertama menguji seberapa presisi sensor membaca umpan balik dari *transmitter* kemudian mengirim hasilnya ke *Android* maupun *SCADA* dengan memperhatikan bagaimana sinyal yang diolah

apakah sudah benar atau masih perlu diperbaiki dan melakukan kalibrasi pada sensor untuk dapat membaca dengan baik. Dalam pengujian pertama ini didapatkan hasil pembacaan yang kurang akurat, namun untuk penerimaan data dari aplikasi *android* dan *SCADA Haiwell* sudah dapat tertampil dengan baik dan jeda waktu pengiriman sekitar 2 detik dalam waktu kirim ke *MQTT*, maka perlu dilakukan proses kalibrasi pada program untuk mengejar akurasi pembacaan dengan cara yang sudah dijabarkan diatas.

Tabel 4. 1. Kalibrasi Sensor

No	Gambar	Keterangan
1		Hasil pembacaan pada transmitter menunjukan hasil 5.33 *NTU*
2		Hasil pembacaan pada terminal *Arduino IDE* adalah 12 *NTU*
3		Kalibrasi program pada alat karena akurasi pembacaan masih kurang

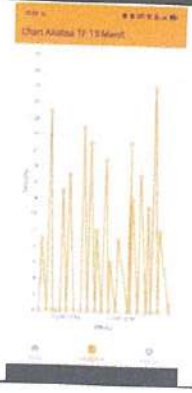
4		Hasil masih menunjukan terbaca 12 *NTU* hasil akurasi masih kurang.
5		Pembacaan sensor berubah menjadi 5.33 maka dilakukan kalibrasi kedua karena tahapan pertama tidak akurat
6		Hasil pembacaan menunjukan nilai 6,84 dan masih kurang, perlu dilakukan kalibrasi ulang
7		Pembacaan berubah ke 7.18 NTU pada transmitter
8		Hasil pembacaan pada terminal Arduino ide menunjukan hasil yang sama 7.18 akurasi sudah baik.

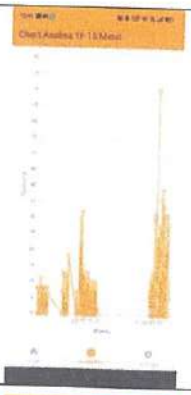
Setelah melalui 3 kali kalibrasi pada sensor dan program ESP32 hasil pembacaan menunjukan keakuratan yang baik maka selanjutnya pengambilan data sampel dapat dilakukan.

B. Hasil Pengujian Pembacaan Pada Aplikasi Android

Tahap pengujian ini akan melihat bagaiaman aplikasi android dapat menampikan pesan hasil pembacaan dari server *MQTT* dan seberapa cepat respon aplikasi dalam menampilkan pesan tersebut. Selain itu juga pengujian terhadap fungsi-fungsi yang sudah dibuat mulai dari suhu cuaca hari ini sampai dengan fungsi koneksi ke server *MQTT*. Dari hasil pengujian disimpulkan bahwa aplikasi berhasil terhubung dengan server MQTT dengan delay pembacaan paling lama 1 menit dan untuk fungsi koneksi dengan MQTT server juga telah berhasil beserta notifikasi yang muncul apabila koneksi sukses namun koneksi akan terputus bila keluar dari aplikasi dan harus koneksi ulang.

Tabel 4. 2. Hasil Pengujian Aplikasi *Android*

No	Gambar Pengujian	Keterangan
1		Tampilan aplikasi android berfungsi dengan baik dari fungsi menampilkan pembacaan dan kondisi cuaca hari ini
2		Fungsi chart pada aplikasi android berjalan dengan baik dan dapat menampilkan trend 15 menit kebelakang.

3		Tampilan chart yang berlalu selama 10 menit masih berjalan dengan baik.
4		Fungsi untuk membuat koneksi ke *MQTT* server berjalan dengan baik, dan muncul notifikasi saat koneksi sukses dibuat, Serta konfigurasi server tersimpan baik.
5		Tampilan *About* juga berfungsi dengan baik, menampilkan logo serta tahapan pengembangan yang sudah dilalui.

C. Hasil Pengujian Pembacaan Pada SCADA Haiwell

SCADA Haiwell dipilih karena selain mudah dalam penggunaan juga karena lisesnsi software yang digunakan tidak berbayar sehingga dalam proses pembuatan mudah dalam mengembangkannya, hasilnya integrasi antara *SCADA* dan *EPS32* dapat dilakukan dengan mudah karena dalam Haiwell sendiri fitur untuk berinteraksi dengan server

MQTT tersedia dan mudah dalam penggunaanya. Berikut hasil percobaan yang telah dilakukan:

1. Hasil pembacaan terintegrasi dengan baik.
2. Pembacaan yang masuk sesuai dengan yang terkirim dari ESP32
3. Delay pembacaan menggunakan *Haiwell SCADA* dan *MQTT* hanya 1 detik.
4. Respon pembacaan dengan perubahan di *Haiwell SCADA* terjadi secara *real-time*

Gambar 4. 22. Tampilan *SCADA Haiwell*

D. Hasil Pengujian Delay Pengiriman

Delay adalah jeda waktu yang terjadi antara saat data (*value*) dikirim dari perangkat (*client*) hingga data tersebut diterima dan diproses oleh *MQTT server* (*broker*). *Delay* ini bisa disebabkan oleh berbagai faktor, baik dari sisi perangkat pengirim, jaringan, maupun

server itu sendiri.*Delay* yang terjadi pada pengiriman value ke *MQTT server* dapat berdampak pada *real-time* monitoring, pengujian delay akan melihat bagaimana respon *MQTT server* dalam menangani informasi yang terkirim kemudia mengirimkan ke perangkat client agar informasi dapat tersampaikan, berikut hasil pengujian :

Tabel 4. 3. Hasil Pengujian *Delay*

Waktu	Hasil Pembacaan (JSON)
2025-05-25 22:46:53:425	{"ntu1":"593.00"}
2025-05-25 22:46:56:427	{"ntu1":"504.00"}
2025-05-25 22:46:59:459	{"ntu1":"550.00"}
2025-05-25 22:47:05:459	{"ntu1":"525.00"}
2025-05-25 22:47:08:460	{"ntu1":"513.00"}
2025-05-25 22:47:11:459	{"ntu1":"583.00"}
2025-05-25 22:47:14:478	{"ntu1":"578.00"}
2025-05-25 22:47:17:458	{"ntu1":"569.00"}
2025-05-25 22:47:20:459	{"ntu1":"594.00"}
2025-05-25 22:47:23:467	{"ntu1":"588.00"}
2025-05-25 22:47:26:459	{"ntu1":"576.00"}
2025-05-25 22:47:28:206	{"ntu1":"0"}

Dari data diatas bisa dilihat bahwa server meneriman data dalam 1 detik menerima 12 data yang berbeda dengan jeda waktu hanya 30 *milisecond* yang menandakan bahwa MQTT server sudah

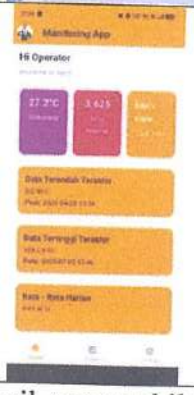
bekerja dengan baik dalam menangani data yang di terima dan terkirim ke perangkat, dalam uji diatas data akan dikonversi menjadi 2 digit di aplikasi.

Gambar 4. 23 Pengujian *Delay MQTT*

E. Hasil Pembacaan Sensor Dan Pengambilan Data

Pembacaan sensor yang sudah terkalibrasi akan diuji dan diambil data pada tahap ini untuk melihat bagaimana hasil pembacaan dibandingkan pembacaan sensor yang tertera pada perangkat, kemudian melihat juga bagaimana respon kecepatan pada perubahan yang terjadi pada air. Pengambilan data akan dilakukan sebanyak 2 kali dihari berbeda selama 30 menit untuk menguji bagaimana kinerja sensor dan ESP32 yang sudah terkalibrasi.

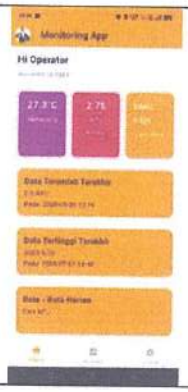
Tabel 4. 4. Pengambilan Data 1

No	Hasil Data 1	Keterangan
1		Pengambilan data pertama menunjukan nilai NTU 3.62 pada transmitter
2		Pada terminal IDE menampilkan data yang terbaca adalah 3.64 karena jeda waktu pengambilan foto berbeda.
3		Pada tampilan SCADA Haiwell menunjukan nilai 3.67
4		Pada tampilan aplikasi android menunjukan nilai 3.625

Hasil pengambilan data pada hari pertama menunjukan bahwa akurasi yang dimiliki oleh sensor sudah baik dan menunjukan angka yang berdekatan dengan hasil pembacaan oleh *transmitter*.

Kemudian dilanjutkan untuk pengambilan data dihari kedua untuk mengukur performa alat apakah masih sesuai atau tidak.

Tabel 4. 5. Pengambilan Data 2

No	Hasil Data 2	Keterangan
1		Data yang terbaca oleh aplikasi android adalah 2.75 NTU
2		Transmitter sensor menunjukan nilai 2.8 NTU
3		Dalam tampilan SCADA Haiwell Sendiri menunjukan angka 2.76 NTU

Dari hasil analisa diatas maka dapat disimpulkan bahwa setelah alat memalui serangkaian proses kalibrasi ESP32 dapat membaca keluaran dari sensor yang mendekati hasil serupa dengan pembacaan transmitter, walaupun adanya fluktuasi pembacaan namun *range* pembacaan masih mendekati dengan aktual sensor.*SCADA haiwel* maupun aplikasi *android* berfungsi dengan semestinya dalam

menyampaikan hasil pembacaan dari sensor.Dalam akurasi pembacaan sensor sendiri dalam pengambilan data di hari pertama dan kedua memiliki akurasi yang baik juga yaitu 98,32% sedangkan dihari kedua pengambilan data memiliki akurasi sebanyak 98,57%. Rumus perhitungan akurasi dari pengambilan data adalah sebagai berikut:

Akurasi = 1-(Nilai Transmitter – Nilai Terbaca/ Nilai Transmitter)x100% .....(1.1)

Dari hasil pengujian pertama dan kedua data dapat dijadikan tabel untuk melihat performa akurasi pembacaan dari alat menggunakan rumus diatas maka dapat dihitung akurasi dari kedua percobaan .

Tabel 4. 6. Hasil Pengambilan Data Pertama

No	Pembacaan Sensor Kekeruhan		Akurasi (persen)
	Aktual	Pembacaan *ESP32*	
1	3,62	3,4	93,92%
2	3,62	3,18	87,84%
3	3,62	3,64	99,50%
4	3,62	3,67	98,63%
5	3,62	3,73	97,05%
6	3,62	3,56	98,34%
7	3,62	3,11	85,91%
8	3,6	3,88	93,29%
9	3,6	3,71	97,03%
10	3,6	3,57	99,16%
Rata-Rata			95,07%

Dihari pertama pengambilan data didapatkan rata-rata akurasi tiap data adalah 95,07 %.

Tabel 4. 7. Hasil Pengambilan Data Kedua

No	Pembacaan Sensor Kekeruhan		Akurasi (persen)
	Aktual	Pembacaan *ESP32*	
1	2,8	2,7544	99,18%
2	2,8	2,8512	99,22%
3	2,8	2,8698	99,22%
4	2,8	2,8177	99,21%
5	2,8	2,674	99,16%
6	2,8	2,7246	99,17%
7	2,8	2,8363	99,21%
8	2,8	2,6621	99,15%
9	2,8	2,773	99,19%
10	2,8	2,6658	99,15%
11	2,8	2,7357	99,18%
12	2,8	2,8661	99,22%
13	2,8	2,7134	99,17%
14	2,8	2,6613	99,15%
15	2,8	2,6896	99,16%
16	2,8	2,732	99,18%
17	2,8	2,6605	99,15%
18	2,8	2,8326	99,21%
19	2,8	2,8363	99,21%
20	2,8	2,7693	99,19%
Rata-Rata			99,18%

Dari pengambilan data kedua dapat dilihat bahwa akurasi alat masih baik yaitu dengan 20 data dan rata-rata akurasi nya sebanyak 99,18% sedangkan untuk batas minimal keterimaan akurasi pembacaan di PT Air Semarang Barat sendiri adalah kurang lebih 5% *deviasi* pembacaan.

F. Analisa

Sistem monitoring kekeruhan air berbasis *SCADA* dan *IoT* dengan antarmuka *Android* menggunakan sensor *Endress and Hauser* akan mengirimkan hasil pembacaan yang diterima oleh ESP32 ke *MQTT server* dengan *delay* 30 *milisecond* sesuai dengan hasil pengujian sensor tidak akan mengirimkan hasil pembacaan apabila voltase yang dibaca dibawah *set point* yang sudah ditentukan untuk menjaga error dalam pembacaan, hasil dikirim ke *MQTT server* dengan topik "data/turbid526/ntu/4456" dan topik ini digunakan oleh *SCADA Haiwell* dan aplikasi *android* untuk menerima data kemudian data terserbut akan ditampilkan dalam antarmuka yang sudah dibangun sebelumnya. Hasil pembacaan antara aktual dengan ESP32 memiliki *deviasi* atau error dengan rata-rata diangka 95,07% pada pengujian pertama dan 99,18% pada pengujian kedua angka tersebut sudah memenuhi kategori standar *deviasi* pembacaan yang ada di PT Air Semarang Barat. Aplikasi *android* sendiri berjalan baik mulai dari fungsi koneksi dengan MQTT server sampai dengan menampilkan hasil data begitu pula dengan *Haiwell SCADA* yang berjalan tanpa kendala dalam menerima dan menampilkan data yang terbaca oleh sensor. Serta tahapan kalibrasi yang baik dapat menunjang akurasi yang baik pula pada pembacaan sensor.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan perancangan dan hasil pengujian dari judul Sistem monitoring kekeruhan air berbasis *SCADA* dan *IoT* dengan antarmuka *Android* menggunakan sensor *Endress And Hauser* dapat disimpulkan bahwa alat yang dirancang untuk melakukan pemantauan kekeruhan air secara *real-time* berhasil dirancang dengan bukti bahwa sensor dapat membaca hasil keluaran dari *transmitter Endress And Hauserr* dan menampilkan secara visual data yang terkirim kedalam aplikasi *android* maupun *SCADA Haiwell* dan sensor *turbidity Arduino* pun bekerja baik.

Alat yang sudah dilakukan kalibrasi untuk pembacaan sensor mampu membaca sensor dengan akurasi yang baik, dari pengujian pertama didapati bahwa rata-rata akurasi pembacaan 95,07% sedangkan pada pengujian kedua mendapatkan rata-rata akurasi sebanyak 99,18% serta *delay* dalam pengiriman pesan ke *MQTT server* sebanyak *30milisecond* dengan bukti pengujian *delay* yang sudah dilakukan.

5.2 Saran

Adapun beberapa saran yang dapat disampaikan selama penelitian agar dapat meningkatkan kinerja alat meliputi :

1. Agar penelitian selanjutnya disarankan dapat menggunakan modul komunikasi *ModBus* pada *transmitter* maupun *ESP32*.

2. Sensor Transmitter yang berada di Intake PT Air Semarang Barat agar dapat di modifikasi meggunakan keluaran sinyal berbentuk modbus.
3. Dipasang Wi-Fi yang jangkauan-nya lebih luas.

DAFTAR PUSTAKA

[1] S. W. Sari, S. D. Cahyani, and D. Sari, "Efektivitas Pengelolaan Air Bersih Menggunakan Metode Filtrasi Dengan Media Zeolit Dan Karbon Aktif Terhadap Derajat …," *J. Kesehat. Tambusai*, vol. 5, no. 3, pp. 9176–9186, 2024.

[2] V. Kallista and dan Govira Christiadora Asbanu, "Analisis Efektivitas Penggunaan Elektroklorinasi dan Gas Klor Pada Proses Disinfeksi Air Minum (Studi Kasus: PERUMDA Air Minum Tirta Khatulistiwa)," vol. 21, pp. 269–278, 2023, doi: 10.14710/jil.21.2.268-278.

[3] MENTERI NEGARA LINGKUNGAN HIDUP, "KEPUTUSANMENTERI NEGARA LINGKUNGAN HIDUPNOMOR : 115 TAHUN 2003TENTANGPEDOMAN PENENTUAN STATUS MUTU AIRMENTERI NEGARA LINGKUNGAN HIDUP." Accessed: Mar. 21, 2025. [Online]. Available: https://luk.staff.ugm.ac.id/atur/sda/KepmenLH115-2003StatusMutuAir.pdf

[4] A. Sinaga, I. Napitupulu, S. Sebayang,) T Hasballah,) Universitas, and D. Agung, "PERANCANGAN POMPA SENTRIFUGAL UNTUK WATER TREATMENT PLANT YANG DI GUNAKAN PADA PT. MULTIMAS NABATI ASAHAN."

[5] Y. Efendi, "INTERNET OF THINGS (IOT) SISTEM PENGENDALIAN LAMPU MENGGUNAKAN RASPBERRY PI BERBASIS MOBILE," *J. Ilm. Ilmu Komput.*, vol. 4, no. 1, 2018, [Online]. Available: http://ejournal.fikom-unasman.ac.id

[6] U. A. Pringsewu *et al.*, "Aisyah Journal of Informatics and Electrical Engineering IMPLEMENTASI SISTEM FILTRASI AIR BERBASIS INTERNET OF THINGS DAN FUZZY LOGIC DENGAN MEDIA FILTER LIMBAH SABUT KELAPA BERTENAGA MIKROHIDRO UNTUK KEBUTUHAN MANDI, CUCI DAN KAKUS", [Online]. Available: http://jti.aisyahuniversity.ac.id/index.php/AJIEE

[7] Muhammad Revaldi Frizky, Hendro Priyatman, and Seno D. Panjaitan, "RANCANG BANGUN SCADA DENGAN TEKNOLOGI INDUSTRIAL IoTDIIMPLEMENTASIKAN PADA SISTEM KENDALI BERBASIS PLC," *Kohesi J. Multidisiplin Saintek*, vol. Volume 7 No 3/ 2025, pp. 1–17, 2025, Accessed: Mar. 21, 2025. [Online]. Available: https://ejournal.warunayama.org/kohesi

[8] A. M. Putri and P. Kurnia, "IDENTIFIKASI KEBERADAAN BAKTERI COLIFORM DAN TOTAL MIKROBA DALAM ES DUNG-DUNG DI SEKITAR KAMPUS UNIVERSITAS MUHAMMADIYAH SURAKARTA," *Media Gizi Indones.*, vol. 13, no. 1, p. 41, Aug. 2018, doi: 10.20473/mgi.v13i1.41-48.

[9] S. M, H. B, M. Rizal H, and A. Ardillah, "RANCANG ALAT SISTEM KONTROL PERGANTIAN AIR KERUH DENGAN POMPA SP-12-00 DAN SENSOR TURBIDITY PADA AKUARIUM," *ILTEK J. Teknol.*, vol. 15, no. 01, pp. 21–24, Apr. 2020, doi: 10.47398/iltek.v15i01.7.

[10] S. A. Nugroho, S. Aprilia, F. Anindyahadi, and N. S. Budi, "PERANCANGAN ALAT UKUR KADAR pH DAN SUHU AIR BERBASIS ARDUINO UNO," *J. Inform. dan Tek. Elektro Terap.*, vol. 13, no. 2, 2025, doi: 10.23960/jitet.v13i2.6398.

[11] "Mengenal Android Studio untuk Pemula dan Cara Installnya untuk Memulai Project." Accessed: Jul. 13, 2025. [Online]. Available: https://www.binar.co.id/blog/mengenal-android-studio-untuk-pemula

[12] "Apa itu Refactoring? Arti, Fungsi, Contoh, FAQs 2025 | RevoU." Accessed: Jul. 13, 2025. [Online]. Available: https://www.revou.co/kosakata/refactoring

[13] "Apa itu User Interface? Arti, Fungsi, Contoh, FAQs 2025 | RevoU." Accessed: Jul. 13, 2025. [Online]. Available: https://www.revou.co/kosakata/user-interface

[14] "Drag–and–Drop: How to Design for Ease of Use - NN/g." Accessed: Jul. 13, 2025. [Online]. Available: https://www.nngroup.com/articles/drag-drop/

[15] "Apa itu Git dan Kenapa Penting bagi Programmer?" Accessed: Jul. 13, 2025. [Online]. Available: https://www.petanikode.com/git-untuk-pemula/

[16] "Kotlin vs Java: Lebih Unggul Mana untuk Aplikasi Android?" Accessed: Jul. 13, 2025. [Online]. Available: https://idwebhost.com/blog/kotlin-vs-java-aplikasi-android/

[17] "Framework vs. Library: Apa Bedanya dan Kapan Harus Digunakan? - CODEPOLITAN." Accessed: Jul. 13, 2025. [Online]. Available: https://www.codepolitan.com/blog/framework-vs-library-apa-bedanya-dan-kapan-harus-digunakan-wulpij/

[18] "Bahasa Pemrograman Flutter: Panduan Lengkap untuk Pemula - CODEPOLITAN." Accessed: Jul. 13, 2025. [Online]. Available: https://www.codepolitan.com/blog/bahasa-pemrograman-flutter-panduan-lengkap-untuk-pemula/

[19] "Memilih Framework Cross-Platform yang Tepat untuk Proyek Kamu." Accessed: Jul. 13, 2025. [Online]. Available: https://azuralabs.id/blog-programming/memilih-framework-cross-platform-yang-tepat-untuk-proyek-kamu

[20] W. T. Wahyudi, S. Karyanto, M. T. Si, M. Antosia, and S. Si, "Rancang Bangun Alat Resistivitas Berbasis Arduino Menggunakan Modul ACS712 dan ADS1115," *J. Tek.*, pp. 1–7, 2016.

LAMPIRAN

Lampiran 1. Kode ESP32

```
#include <WiFi.h>
#include <WebServer.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <Adafruit_ADS1X15.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
// Pin I2C Configuration
#define SDA_ADS 21       // Pin
SDA untuk ADS1115 (standar)
#define SCL_ADS 22       // Pin
SCL untuk ADS1115 (standar)
#define SDA_LCD 33       // Pin
SDA untuk LCD
#define SCL_LCD 32       // Pin
SCL untuk LCD
// WiFi dan MQTT Configuration
const char *ssid = "Tselhome-
HAP";
const char *password = "Em-
bul@8888";
const char *mqtt_broker = "bro-
ker.emqx.io";
const int mqtt_port = 1883;
const char *mqtt_username =
"emqx";
const char *mqtt_password =
"public";
// MQTT Topics
const char *topic_ntu = "da-
ta/turbid526/ntu/4456";
// Sensor Configuration
const int16_t ZERO_ADC_LOW =
1560;
// Nilai ADC saat 0 NTU (batas
bawah)
const int16_t ZERO_ADC_HIGH =
1580;
// Nilai ADC saat 0 NTU (batas
atas)
const float ZERO_VOLTAGE =
0.1975;       // Tegangan saat 0
NTU (dihitung dari ZE-
RO_ADC_HIGH)
const float MAX_VOLTAGE =
4.09// Tegangan maksimal sensor
const float maxNTU = 388.0
// Skala maksimal NTU
const float ADC_LSB = 0.125
// 4.096V / 32767 (untuk gain
GAIN_ONE)
const float CAL_VOLTAGE = 0.5;
const float CAL_NTU = 6.4;
// LCD Configuration
#define LCD_ADDRESS 0x27    //
Alamat I2C LCD (bisa 0x27 atau
0x3F)
#define LCD_COLS 16
#define LCD_ROWS 2
// Global Variables
float ntu1 = 0;
float ntu2 = 0;
int16_t currentADC0 = 0;
int16_t currentADC1 = 0;
float currentVoltage0 = 0;
float currentVoltage1 = 0;
unsigned long lastSensorRead =
0;
unsigned long lastPublish = 0;
unsigned long lastLcdUpdate = 0;
unsigned long lastScrollUpdate =
0;
bool wifiConnected = false;
bool mqttConnected = false;
// Objects
WiFiClient espClient;
PubSubClient
mqtt_client(espClient);
Adafruit_ADS1115 ads;  //
ADS1115 menggunakan bus I2C
terpisah
LiquidCrystal_I2C
lcd(LCD_ADDRESS, LCD_COLS,
LCD_ROWS);
WebServer server(80);

void setI2CPins(int sda, int
scl) {
    Wire.end(); // End previous
I2C session
    Wire.begin(sda, scl); //
Start I2C with new pins
}
```

```
void connectToWiFi() {
WiFi.begin(ssid, password);
Serial.print("Connecting to
WiFi");
// Update LCD saat connecting
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Connecting WiFi");
lcd.setCursor(0, 1);
lcd.print("Please wait...");
int attempts = 0;
while (WiFi.status() != WL_CON-
NECTED && attempts < 20) {
delay(1000);
Serial.print(".");
attempts++;
setI2CPins(SDA_LCD, SCL_LCD);
// Update progress di LCD
lcd.setCursor(attempts % 16, 1);
lcd.print(".");
}
if (WiFi.status() == WL_CON-
NECTED) {
wifiConnected = true;
Serial.println("\nConnected to
WiFi");
Serial.print("IP address: ");
Serial.println(WiFi.localIP());
// Update LCD dengan status ber-
hasil
setI2CPins(SDA_LCD, SCL_LCD);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("WiFi Connected");
lcd.setCursor(0, 1);
lcd.print(WiFi.localIP());
delay(2000);
}else {
wifiConnected = false;
Serial.println("\nFailed to con-
nect to WiFi");
// Update LCD dengan status ga-
gal
setI2CPins(SDA_LCD, SCL_LCD);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("WiFi Failed");
lcd.setCursor(0, 1);
lcd.print("Check config");
delay(2000);
}
}


void mqttCallback(char *topic,
byte *payload, unsigned int
```

```
LiquidCrystal_I2C
lcd(LCD_ADDRESS,
LCD_COLS, LCD_ROWS);
WebServer server(80);

void setI2CPins(int
sda, int scl) {
    Wire.end(); // End
previous I2C session
    Wire.begin(sda,
scl); // Start I2C
with new pins
}
```

```
void connectToMQTT() {
if (!wifiConnected)
return;
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Connecting
MQTT");
lcd.setCursor(0, 1);
lcd.print("bro-
ker.emqx.io");
while (!mqtt_cli-
ent.connected())
{String client_id =
"esp32-S2-client-" +
String(WiFi.mac-
Address());
Serial.printf("Con-
necting to MQTT Broker
as %s...\n", cli-
ent_id.c_str());
if (mqtt_client.con-
nect(cli-
ent_id.c_str(),
mqtt_username,
mqtt_password)) {Se-
rial.println("Con-
nected to MQTT bro-
ker");
mqtt_client.sub-
scribe(topic_ntu);
mqttConnected = true;
// Update LCD dengan
status berhasil
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("MQTT Con-
nected");
lcd.setCursor(0, 1);
```

```
length) {Serial.print("Message
received on topic [");
Serial.print(topic);
Serial.print("]: ");
String message = "";
for (int i = 0; i < length; i++)
{message += (char)payload[i];
}
Serial.println(message);
}

void publishSensorDataJSON(const
char *topic, float value) {if
(!mqttConnected) return;
StaticJsonDocument<200> doc;
doc["ntu1"] = String(value,
4);char jsonBuffer[256];
serializeJson(doc,
jsonBuffer);mqtt_client.pub-
lish(topic, jsonBuffer);
Serial.print("Published JSON to
");
Serial.print(topic);
Serial.print(": ");
Serial.println(jsonBuffer);
}

void publishSensorData(const
char *topic, float value) {
if (!mqttConnected) return;
char msg[10];
snprintf(msg, 10, "%.4f",
value);
mqtt_client.publish(topic, msg);
Serial.print("Published to ");
Serial.print(topic);
Serial.print(": ");
Serial.println(msg);
}
void readsensor() {
// Baca nilai ADC
setI2CPins(SDA_ADS, SCL_ADS); //
Set I2C pins for ADS1115
currentADC0 = ads.readADC_Sin-
gleEnded(0);
Serial.print("Raw ADC Value: ");
Serial.println(currentADC0)
// Hitung tegangan aktual
currentVoltage0 = currentADC0 *
ADC_LSB / 1000;  // Konversi

void updateLCD() {
// Ganti display mode setiap 3
detik
lcd.print("Ready to
send");
delay(2000);
}
else {
Serial.print("Failed
to connect to MQTT
broker, rc=");Se-
rial.println(mqtt_cli-
ent.state());
mqttConnected = false;
// Update LCD dengan
status gagal
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("MQTT
Failed");
lcd.setCursor(0, 1);
lcd.print("Retry in
5s");
delay(5000);
break; // Exit loop to
prevent blocking
}
}
}
ke volt
Serial.print("Voltage
1 : ");
Serial.print(cur-
rentVoltage1, 4);
Serial.println(" V");
// Kalibrasi nilai NTU
if (currentADC0 >=
ZERO_ADC_LOW && cur-
rentADC0 <=
ZERO_ADC_HIGH) {ntu1 =
0.0;  // Dianggap 0
NTU dalam range ini
}else if (currentADC0
< ZERO_ADC_LOW) {
// Jika nilai di bawah
range zero
ntu1 = 0.0;
Serial.println("Warn-
ing: ADC value below
zero point!");
}else if (currentVolt-
age0 <= CAL_VOLTAGE ){
```

```
if (millis() - lastLcdUpdate >= 3000) {displayMode = (displayMode + 1) % 4;
lastLcdUpdate = millis();
scrollPosition = 0;
}
// Update setiap 500ms untuk animasi scroll if (millis() - lastScrollUpdate >= 500) {
setI2CPins(SDA_LCD, SCL_LCD); // Set I2C pins for LCD
lcd.clear();
switch (displayMode) {
case 0: // Tampilan utama NTU
lcd.setCursor(0, 0);
lcd.print("Turbidity Level");
lcd.setCursor(0, 1);
lcd.print(ntu1, 2);
lcd.print(" NTU");
if (ntu1 > 200) {
lcd.setCursor(11, 1);
lcd.print("HIGH!");
}
break;
case 1: // Tampilan tegangan dan ADC
lcd.setCursor(0, 0);
lcd.print("ADC: ");
lcd.print(currentADC0);
lcd.setCursor(0, 1);
lcd.print("V: ");
lcd.print(currentVoltage0, 4);
lcd.print("V");
break;
case 2: // Status koneksi
lcd.setCursor(0, 0);
lcd.print("WiFi:");
lcd.print(wifiConnected ? "OK" : "NO");
lcd.print(" MQTT:");
lcd.print(mqttConnected ? "OK" : "NO");
lcd.setCursor(0, 1);
lcd.print("Uptime:");
lcd.print(millis() / 1000);
lcd.print("s");
break;
```

```
void setup() {
Serial.begin(115200);
pinMode(34, INPUT);
// Initialize LCD
```

```
ntu1 = (currentVoltage0 / CAL_VOLTAGE) * CAL_NTU;
}else {
// Hitung tegangan terkoreksi (dikurangi offset)
float correctedVoltage = currentVoltage0 - ZERO_VOLTAGE - CAL_VOLTAGE ;
float maxCorrectedVoltage = MAX_VOLTAGE - ZERO_VOLTAGE - CAL_VOLTAGE ;
// Hitung NTU (proporsional terhadap tegangan)
ntu1 = CAL_NTU + (correctedVoltage / maxCorrectedVoltage) * (maxNTU - CAL_NTU);
// Batasi nilai antara 0-maxNTU
ntu1 = constrain(ntu1, 0.0, maxNTU);
}
Serial.print("Calculated NTU 1: ");Serial.println(ntu1, 2);
setI2CPins(SDA_ADS, SCL_ADS); // Set I2C pins for ADS1115
currentADC1 = ads.readADC_SingleEnded(1);
Serial.print("Raw ADC Value: ");
Serial.println(currentADC1);
currentVoltage1 = currentADC1 * ADC_LSB / 1000;  // Konversi ke volt
Serial.print("Voltage 2: ");
Serial.print(currentVoltage1, 4);
Serial.println(" V");
float ntu2 = map(currentVoltage1, 1.1, 4.09, 50, 0);
Serial.print("Calculated NTU 2: ");Serial.println(ntu2, 2);
```

```
setI2CPins(SDA_LCD, SCL_LCD);
lcd.init();
lcd.backlight();
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Turbidity Monitor");
lcd.setCursor(0, 1);
lcd.print("Starting...");
Serial.println("LCD initialized
on pins SDA:" + String(SDA_LCD)
+ " SCL:" + String(SCL_LCD));
delay(2000);
// Initialize ADS1115
setI2CPins(SDA_ADS, SCL_ADS);
if (!ads.begin()) {Se-
rial.println("Failed to initial-
ize ADS.");
setI2CPins(SDA_LCD, SCL_LCD);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("ADS1115 Error");
lcd.setCursor(0, 1);
lcd.print("Check wiring");
while (1);
}ads.setGain(GAIN_ONE);
Serial.println("ADS1115 initial-
ized on pins SDA:" +
String(SDA_ADS) + " SCL:" +
String(SCL_ADS));
// Connect to WiFi
connectToWiFi();
 if (wifiConnected) {
// Setup MQTT
mqtt_client.setServer(mqtt_bro-
ker, mqtt_port);
mqtt_client.setCallback(mqtt-
Callback);
connectToMQTT();
// Setup Web Server
server.on("/", handleRoot);
server.on("/data", handleData);
server.onNotFound(handleNot-
Found);
server.begin();
Serial.println("Web server
started");Serial.print("Access
the dashboard at:
http://");Seral.println(WiFi.lo-
calIP());}
```

```
}case 3: // Informasi
jaringan (scroll)
lcd.setCursor(0, 0);
lcd.print("Network
Info");
lcd.setCursor(0, 1);
String ipString = "IP:
" + WiFi.lo-
calIP().toString();
if (ipString.length()
> 16) {
// Scroll teks panjang
int startPos = scroll-
Position % (ip-
String.length() - 15);
lcd.print(ip-
String.sub-
string(startPos,
startPos + 16));
scrollPosition++;
} else {
lcd.print(ipString);
}
break;
}
lastScrollUpdate =
millis();
}
}
// Web Server Handlers
void handleRoot() {
server.send(200,
"text/html",
htmlPage);
}
void handleData() {
StaticJsonDocu-
ment<300> doc;
doc["ntu"] = ntu1;
doc["adc"] = curren-
tADC0;
doc["voltage"] = cur-
rentVoltage0;
doc["wifi_connected"]
= wifiConnected;
doc["mqtt_connected"]
= mqttConnected;
doc["ip_address"] =
WiFi.lo-
calIP().toString();
doc["uptime"] = mil-
lis();
String jsonString;
serializeJson(doc,
jsonString);
```

```
server.send(200, "applica-
tion/json", jsonString);
}
void handleNotFound() {
server.send(404, "text/plain",
"Not Found");
}
// Tampilkan pesan ready di LCD
setI2CPins(SDA_LCD, SCL_LCD);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("System Ready");
lcd.setCursor(0, 1);
lcd.print("Monitoring...");
delay(2000);
}
void loop() {
// Handle web server
if (wifiConnected)
{server.handleClient();
}// Read sensor every 500ms
if (millis() - lastSensorRead >=
500) {readsensor();
lastSensorRead = millis();
}// Update LCD display
updateLCD();
// Publish to MQTT every 500ms
if (millis() - lastPublish >=
500) {
if (mqttConnected) {
publishSensorDataJSON(topic_ntu,
ntu1);publishSensor-
Data(topic_ntu,
ntu1);}lastPublish = millis();}
// Check WiFi connection
if (WiFi.status() != WL_CON-
NECTED) {wifiConnected =
false;mqttConnected = false;
} else {wifiConnected
= true;
// Try to reconnect
MQTT if disconnectedif
(!mqtt_client.con-
nected()) {
mqttConnected = false;
static unsigned long
lastMqttAttempt = 0;
if (millis() -
lastMqttAttempt >
10000) { // Try every
10 seconds
connectToMQTT();
lastMqttAttempt = mil-
lis();
}} else {
mqttConnected =
true;}}
// Handle MQTT
if (mqttConnected) {
mqtt_client.loop();
}delay(10); // Small
delay for stability
} constTextStyle(font-
Size: 16))) .toList(),
), const
```

Lampiran 2. Kode Android Studio

About.dart

```
import ' package : flutter/
material. dart '; class
AboutPage extends Stateless-
Widget { AboutPage ({ super.
key });final List < String >
team Members = [" Septyawan
Bagus W.", "24014012 "];final
List < String > changelog = ["
v2 .0.0 - Bug fixes and new UI
Improvement"," v1 .3.0 - Fix-
ing MQTT Function on UI & Add
local storage MQTT Client","
v1 .2.0 - UI enhancements and
MQTT Client Features", " v1
.1.0 - Added new UI monitoring
feature ",];@ overrideWidget
build ( Build Context context)
{ return Scaffold (app Bar:
App Bar(title : const Text ("
About "), background Color :
Colors. orange ,),body : Cen-
ter( child : Column (main Ax-
isAlignment : Main AxisAlign-
ment . center , children :
[Clip RRect(borderRadius: Bor-
derRadius. circular (20) ,
child :          Image .
asset( ' asset/ images/
logo_bulet. png ',width : 250
,height: 250 ,fit: Box Fit.
cover ,),),const Sized Box (
height: 20), const Text(" De-
veloped by :",style :
TextStyle ( fontSize : 18 ,
fontWeight:  FontWeight.
bold),),const Sized Box ( Col-
umn (children : team Members.
map (( name ) => Text( name ,
style : const TextStyle (
fontSize : 16))). to List
(),),const Sized Box ( height:
20), const Text(" Changelog
:",style : TextStyle ( font-
Size : 18 ,  fontWeight:
FontWeight. bold ),),const
Sized Box ( height: 10), Col-
umn (children : changelog. map
(( log ) => Padding (padding :
const Edge Insets. symmetric(
vertical: 4), child : Text(log
,textAlign :  TextAlign . cen-
ter ,style : const TextStyle (
fontSize : 16),),)). to List
(),),const Sized Box ( height:
30), const Text(" Dibuat Guna
Mendukung Pengembangan Ap-
likasi Tugas Akhir", style :
TextStyle ( fontSize : 16 ,
fontWeight: FontWeight. bold
,color:  Colors. grey
),),],),),);}}
```

main.dart

```
import 'package:flutter/mate-
rial.dart'; import 'pack-
age:shared_prefer-
ences/shared_prefer-
ences.dart'; import 'pack-
age:get_time_ago/get_time_ago.
dart'; import 'package:syncfu-
sion_flut-
ter_charts/charts.dart'; im-
port 'package:flutter_na-
tive_splash/flutter_na-
tive_splash.dart'; import
'package:intl/intl.dart'; im-
port 'dart:async'; import
'dart:convert'; import
'notf_setting.dart'; import
'mqtt_service.dart'; import
'weather.dart'; import
"about.dart"; final
MqttService mqttService =
MqttService(); void main() {
WidgetsBinding widgetsBinding
= WidgetsFlutterBinding.en-
sureInitialized(); FlutterNa-
tiveSplash.preserve(widgets-
Binding: widgetsBinding); run-
App(MyApp()); FlutterNa-
tiveSplash.remove(); } class
ChartData { final DateTime
timestamp; final double pay-
load;
ChartData(this.timestamp,
this.payload); } class MyApp
```

```
extends StatefulWidget { const
MyApp({super.key}); @override
MyAppState createState() =>
MyAppState(); } class My-
AppState extends State<MyApp>
{ int currentIndex = 0; final
List<Widget> children = [
HomeScreen(), Analyt-
icsScreen(), SettingsScreen(),
]; void onTabTapped(int index)
{ setState(() {

currentIndex = index; }); }
@override Widget build(Build-
Context context) { return Ma-
terialApp( debug-
ShowCheckedModeBanner: false,
// Menghilangkan tulisan debug
title: 'Monitoring Sensor',
theme: ThemeData( prima-
rySwatch: Colors.orange,
fontFamily: 'Roboto', ), home:
Scaffold( body: children[cur-
rentIndex], bottomNavigation-
Bar: BottomNavigationBar(
items: const <BottomNaviga-
tionBarItem>[ BottomNaviga-
tionBarItem( icon:
Icon(Icons.home), label:
'Home', ), BottomNavigation-
BarItem( icon: Icon(Icons.ana-
lytics), label: 'Analytics',
), BottomNavigationBarItem(
icon: Icon(Icons.settings),
label: 'Settings', ), ], cur-
rentIndex: currentIndex, se-
lectedItemColor: Colors.or-
ange, unselectedItemColor:
Colors.grey, onTap: onTab-
Tapped, ), ), ); } } class
HomeScreen extends Stateful-
Widget { @override HomeScreen-
State createState() =>
HomeScreenState(); const
HomeScreen({super.key}); }

class HomeScreenState extends
State<HomeScreen> { final
WeatherService weatherService
= WeatherService(); String
topic = ''; String uptime =
'No data'; String dataTime =
''; String turbid = '0';
String temperature = 'Load-
ing...'; String city = 'Sema-
rang'; String turbidvalue =
'''; String lowestTurbid =
'0.0'; String lowestTurbid-
Timestamp = ''; String high-
estTurbid = '0.0'; String
highestTurbidTimestamp = '';
String averageTurbid = '0.0';
Timer? timer; @override void
initState() { super.init-
State(); getTop-
icFromSharedPreferences();
readTopic();
loadWeatherData(); timeAgo();
GetTimeAgo.setDefault-
Locale('id'); startTimer(); }
@override void dispose() {
timer?.cancel(); // Hentikan
timer saat widget dihapus su-
per.dispose(); } void start-
Timer() { timer = Timer.peri-
odic(Duration(seconds: 10),
(Timer timer) { // Refresh se-
tiap 10 detik setState(() {
readTopic(); // Ambil data
terbaru timeAgo(); }); }); }
Future<void> getTop-
icFromSharedPreferences()
async { // Mengambil topik
dari SharedPreferences final
prefs = await SharedPrefer-
ences.getInstance(); topic =
prefs.getString('Topic') ??
'default_topic'; }

Future<void> readTopic() async
{ await getTop-
icFromSharedPreferences(); fi-
nal prefs = await SharedPref-
erences.getInstance(); String
key = 'mqtt_history_$topic';
List<String>? history =
prefs.getStringList(key); if
(history == null) { history =
[]; }

double lowestValue = dou-
ble.infinity; String
lowestTimestamp = ''; String
highestTimestamp = ''; double
highestValue = -double.infin-
ity; double totalValue = 0.0;
// Untuk menyimpan total nilai
int dataCount = his-
tory.length; // Jumlah data
final DateTime now =
DateTime.now(); // Waktu saat
ini final DateTime last24Hours
```

```
= now.subtract(Duration(hours:
24)); // Waktu 24 jam yang
lalu for (String item in his-
tory) { final Map<String, dy-
namic> mqttData =
jsonDecode(item); // Ambil
payload dan timestamp final
double count = double.try-
Parse(mqttData['payload'] ??
'0.0') ?? 0.0; final String
gettime = mqtt-
Data['timestamp'] ?? ''; final
timed = DateTime.parse(get-
time); final String timestamp
= DateFormat('yyyy-MM-dd
HH:mm').for-
mat(timed).toString(); if
(timed.isAfter(last24Hours)) {
// fungsi menghitung rata2 da-
lam 24 jam totalValue +=
count; // Count Total untuk
Average dataCount++; double
averageValue = dataCount > 0 ?
totalValue / dataCount : 0.0;
// Rata-rata setState(() { av-
erageTurbid = aver-
ageValue.toStringAsFixed(2);
// Simpan rata-rata dengan 2
digit desimal }); } if (count
< lowestValue) { lowestValue =
count; lowestTimestamp =
timestamp; }else if (count >
highestValue) { highestValue =
count; highestTimestamp =
timestamp; } setState(() {
turbid = count.toString();

if (count > 100) { Notifica-
tionService.showNotification(
title: 'Warning Turbidity
Meningkat', body: 'Trubidity
Naik $turbid NTU - Warning
🟤' ); } lowestTurbid =
lowestValue.toString();
lowestTurbidTimestamp =
lowestTimestamp; highestTurbid
= highestValue.toString();
highestTurbidTimestamp = high-
estTimestamp; }); } } Fu-
ture<void> loadWeatherData()
async { try { Map<String, dy-
namic> weatherData = await
weatherService.getWeather();
double temp =
weatherData['main']['temp'];
setState(() { temperature =
'${temp.toStrin-
gAsFixed(1)}°C'; }); } catch
(e) { setState(() { tempera-
ture = 'Error'; }); } } Fu-
ture<void> timeAgo() async {
await getTopicFromSharedPref-
erences(); final prefs = await
SharedPreferences.get-
Instance(); String key =
'mqtt_history_$topic';
List<String>? history =
prefs.getStringList(key); if
(history != null && his-
tory.isNotEmpty) { for (String
item in history) { final
Map<String, dynamic> mqttData
= jsonDecode(item); // Ambil
timestamp final String
timestamp = mqtt-
Data['timestamp']; DateTime
dataTime =
DateTime.parse(timestamp);
setState(() {//Set State untuk
mengupdate UI var live = Get-
TimeAgo.parse(dataTime); //
Parsing ke Waktu uptime =
live.toString(); //Pass data
ke string agar tampil ke UI
}); }} }

@override Widget build(Build-
Context context) { return
Scaffold( appBar: AppBar(
backgroundColor: Colors.or-
ange, elevation: 0, leading:
Padding( padding: const Edge-
Insets.all(8.0), child: Cir-
cleAvatar( backgroundImage:
AssetImage('asset/images/Per-
son.jpg'), ), ), title:
Text('Monitoring App',
textAlign: TextAlign.center,
style: TextStyle(fontSize: 24,
fontWeight: FontWeight.bold,
fontFamily: 'Roboto'), ), ),
body: SingleChildScrollView(
padding: const EdgeIn-
sets.all(16.0), child: Column(
crossAxisAlignment: CrossAxis-
Alignment.start, children: [
Text( 'Hi Operator', style:
TextStyle(fontSize: 24, font-
Weight: FontWeight.bold), ),
SizedBox(height: 8), Text(
'Welcome to Apps', style:
TextStyle(fontSize: 16, color:
```

```
Colors.grey[600]), ),
SizedBox(height: 28), //
Menambahkan kartu informasi
IntrinsicHeight( child: Row(
mainAxisAlignment: MainAxisA-
lignment.spaceBetween, chil-
dren: [ Expanded( child: Info-
Card( color: Colors.purple,
title: temperature, subtitle:
city,

status: '',

), ), SizedBox(width: 8), //
Menambahkan jarak antara kartu
Expanded( child: InfoCard(
color: Colors.pink, title:
turbid, subtitle: 'NTU', sta-
tus: 'Normal', ), ),
SizedBox(width: 8), // Menam-
bahkan jarak antara kartu Ex-
panded( child: InfoCard(
color: Colors.orange, title:
uptime, subtitle: 'Last
Alive', status: '', ), ), ], )
), SizedBox(height: 24), //
Kartu biasa dengan warna
oranye Row( mainAxisAlignment:
MainAxisAlignment.spaceBe-
tween, children: [ Expanded(
child: SimpleCard( title:
'Data Terendah Terakhir', fam-
ilyMembers: '$lowestTurbid
NTU', devices: 'Pada:
$lowestTurbidTimestamp' ), ),
], ), SizedBox(height: 24), //
Kartu biasa dengan warna
oranye Row( mainAxisAlignment:
MainAxisAlignment.spaceBe-
tween, children: [ Expanded(

child: SimpleCard( title:
'Data Tertinggi Terakhir',
familyMembers: '$highestTurbid
NTU', devices: 'Pada: $high-
estTurbidTimestamp' ), ), ],
), SizedBox(height: 24), //
Kartu biasa dengan warna
oranye Row( mainAxisAlignment:
MainAxisAlignment.spaceBe-
tween, children: [ Expanded(
child: SimpleCard( title:
'Rata - Rata Harian', fami-
lyMembers: '$averageTurbid
NTU', devices: '', ), ), ], ),
], ), ), ); } } class
AnalyticsScreen extends State-
fulWidget { const Analyt-
icsScreen({super.key}); @over-
ride AnalyticsScreenState cre-
ateState() => AnalyticsScreen-
State(); } class Analyt-
icsScreenState extends
State<AnalyticsScreen> { Fu-
ture<List<ChartData>>?
chartData; String topic = '';
Timer? timer; Future<void>
getTopicFromSharedPrefer-
ences() async { // Mengambil
topik dari SharedPreferences
final prefs = await
SharedPreferences.get-
Instance(); topic =
prefs.getString('Topic') ??
'default_topic'; } Fu-
ture<List<ChartData>>
getChartData() async {

final prefs = await
SharedPreferences.get-
Instance(); String key =
'mqtt_history_$topic'; //
Ganti dengan key yang sesuai
List<String>? history =
prefs.getStringList(key); if
(history == null || his-
tory.isEmpty) { return []; }
List<ChartData> chartData =
[]; final DateTime now =
DateTime.now(); // Waktu saat
ini final DateTime
last15Minutes = now.sub-
tract(Duration(minutes: 15));
// Waktu 15 menit yang lalu
for (String item in history) {
final Map<String, dynamic>
mqttData = jsonDecode(item);
final double payload = dou-
ble.tryParse(mqttData['pay-
load'] ?? '0.0') ?? 0.0; final
String timestamp = mqtt-
Data['timestamp'] ?? ''; //
Konversi timestamp ke DateTime
final DateTime dateTime =
DateTime.parse(timestamp); //
Cek apakah data masih dalam 15
menit terakhir if
(dateTime.isAfter(last15Minute
s)) { // Tambahkan ke list
chartData
chartData.add(ChartData(dateTi
me, payload)); } } return
```

```
chartData; } void startTimer()
{ timer = Timer.periodic(Dura-
tion(seconds: 7), (Timer
timer) { // Refresh setiap 5
detik setState(() { chartData
= getChartData(); // Ambil
data terbaru }); }); } @over-
ride void initState() { su-
per.initState(); getTop-
icFromSharedPreferences();
chartData = getChartData(); //
Ambil data saat inisialisasi
startTimer(); } @override void
dispose() { timer?.cancel();
// Hentikan timer saat widget
dihapus super.dispose(); }

@override Widget build(Build-
Context context) { return
Scaffold( appBar: AppBar(
backgroundColor: Colors.or-
ange, elevation: 0, title:
Text('Chart Analisa TF 15 Men-
it'), ), body: Future-
Builder<List<ChartData>>( fu-
ture: chartData, builder:
(context, snapshot) { if
(snapshot.connectionState ==
ConnectionState.waiting) { re-
turn Center(child: Circu-
larProgressIndicator()); }
else if (snapshot.hasError) {
return Center(child: Text('Er-
ror: ${snapshot.error}')); }
else if (!snapshot.hasData ||
snapshot.data!.isEmpty) { re-
turn Center(child: Text('No
data available')); } else {
return Padding( padding: const
EdgeInsets.all(16.0), child:
SfCartesianChart( prima-
ryXAxis: DateTimeAxis( title:
AxisTitle(text: 'Waktu'),
dateFormat: DateFormat('dd/MM
HH:mm'), // Format tanggal dan
waktu ), primaryYAxis: Numeri-
cAxis( title: AxisTitle(text:
'Turbidity'), ), series: <Car-
tesianSeries<ChartData,
DateTime>>[
LineSeries<ChartData,
DateTime>( // Ganti StepLi-
neSeries dengan LineSeries
dataSource: snapshot.data!,
xValueMapper: (ChartData data,
_) => data.timestamp,
yValueMapper: (ChartData data,
_) => data.payload, color:
Colors.orange, width: 2, mark-
erSettings: MarkerSettings(is-
Visible: true), // Tampilkan
marker ), ], ), ); } ), )

); } } class SettingsScreen
extends StatelessWidget {
const SettingsScreen({su-
per.key}); @override Widget
build(BuildContext context) {
return Scaffold( appBar: Ap-
pBar( backgroundColor: Col-
ors.orange, elevation: 0, ti-
tle: Text('Settings'), ),
body: ListView( children: [
ListTile( leading:
Icon(Icons.settings), title:
Text('MQTT Settings'), onTap:
() { Navigator.push( context,
MaterialPageRoute(builder:
(context) => MqttSet-
tingsScreen()), ); }, ), List-
Tile( leading:
Icon(Icons.info), title:
Text('About'), onTap: () {
Navigator.push( context, Mate-
rialPageRoute(builder: (con-
text) => AboutPage()), ); },
), ], ), ); } } class InfoCard
extends StatelessWidget { fi-
nal Color color; final String
title;

final String subtitle; final
String status; const Info-
Card({required this.color, re-
quired this.title, required
this.subtitle,required
this.status, super.key});
@override Widget build(Build-
Context context) { return
Card( color: color, child:
Padding( padding: const Edge-
Insets.all(16.0), child: Col-
umn( children: [ Text( title,
style: TextStyle(fontSize: 24,
fontWeight: FontWeight.bold,
color: Colors.white),
textAlign: TextAlign.justify,
), SizedBox(height: 8), Text(
subtitle, style:
TextStyle(fontSize: 16, color:
Colors.white), textAlign:
TextAlign.center, ),
```

```
SizedBox(height: 8), Text(sta-
tus, style: TextStyle(font-
Size: 16, color: status ==
'Warning' ? Colors.red : Col-
ors.white), textAlign:
TextAlign.justify), ], ), ),
); } } class SimpleCard ex-
tends StatelessWidget { final
String title; final String
familyMembers; final String
devices; const SimpleCard({re-
quired this.title, required
this.familyMembers, required
this.devices, super.key});
@override Widget build(Build-
Context context) { return Ges-
tureDetector( onTap: () {

// Navigasi ke halaman detail
atau pengaturan Naviga-
tor.push( context, Materi-
alPageRoute(builder: (context)
=> MqttPage()), ); }, child:
Card( color: Colors.orange,
child: Padding( padding: const
EdgeInsets.all(16.0), child:
Column( crossAxisAlignment:
CrossAxisAlignment.start,
children: [ Text( title,
style: TextStyle(fontSize: 20,
fontWeight: FontWeight.bold),
), SizedBox(height: 4),
Text(familyMembers, style:
TextStyle(fontSize: 16)),
SizedBox(height: 4), Text(de-
vices, style: TextStyle(font-
Size: 16, color: devices ==
'Warning' ? Colors.red : Col-
ors.black)), ], ), ), ), ); }
} class MqttPage extends
StatelessWidget { const
MqttPage({super.key}); @over-
ride Widget build(BuildContext
context) { return Scaffold(
appBar: AppBar( title:
Text('MQTT Page'), ), body:
Center( child: Text('MQTT Page
Content'), ), ); } }

class MqttSettingsScreen ex-
tends StatefulWidget { const
MqttSettingsScreen({su-
per.key}); @override MqttSet-
tingsScreenState createState()
=> MqttSettingsScreenState();
} class
```

```
MqttSettingsScreenState ex-
tends State<MqttSet-
tingsScreen> { final _formKey
= GlobalKey<FormState>();
String _brokerAddress = '';
String _port = ''; String
_clientId = ''; String _topic
= ''; bool _isSaved = false;
bool _isConnected = false;
bool _isSubscribed = false;
@override void initState() {
super.initState(); _loadSet-
tings(); _checkMqttConnec-
tion(); Notification-
Service.initialize(); if (_is-
Connected) { _isConnected =
mqttService.isConnected; } }
Future<void> _loadSettings()
async { SharedPreferences
prefs = await SharedPrefer-
ences.getInstance();
setState(() { _brokerAddress =
prefs.getString('broker-
Address') ?? ''; _port =
prefs.getString('port') ?? '';
_clientId =
prefs.getString('clientId') ??
''; _topic =
prefs.getString('Topic') ??
''; _isSaved = _broker-
Address.isNotEmpty ||
_port.isNotEmpty || _clien-
tId.isNotEmpty; }); } void
_checkMqttConnection() {
setState(() { _isConnected =
mqttService.isConnected; }); }
Future<void> _saveSettings()
async { if (_formKey.current-
State!.validate()) {
_formKey.currentState!.save();

SharedPreferences prefs =
await SharedPreferences.get-
Instance(); await
prefs.setString('broker-
Address', _brokerAddress);
await prefs.setString('port',
_port); await
prefs.setString('clientId',
_clientId); await
prefs.setString('Topic',
_topic); setState(() { _is-
Saved = true; }); } } Fu-
ture<void> _toggleConnection()
async { int port = int.try-
Parse(_port) ?? 1883; if
```

```
(_isConnected) { await
mqttService.disconnect();
setState(() { _isConnected =
false; _isSubscribed = false;
}); NotificationService.showN-
otification( title: 'MQTT Dis-
connected', body: 'Discon-
nected from MQTT broker - Dis-
connected 🔴' ); } else {
bool connected = await
mqttService.connect( broker:
_brokerAddress, port: port,
clientId: _clientId, topic:
_topic, ); setState(() { _is-
Connected = connected; }); if
(connected) { Notification-
Service.showNotification( ti-
tle: 'MQTT Connected', body:
'Successfully connected to
MQTT broker - Connected ✨',
); } else { Notification-
Service.showNotification( ti-
tle: 'MQTT Connection Failed',
body: 'Failed to connect to
MQTT broker - Failed 🔴', );
}

} } Future<void> _toggleSub-
scribe() async { if (_isSub-
scribed) { // Unsubscribe dari
semua topic bool unsubscribed
= await mqttService.unsub-
scribeAllTopics(); setState(()
{ _isSubscribed = !unsub-
scribed; }); Notification-
Service.showNotification( ti-
tle: 'MQTT Successfully Unsub-
scribed', body: 'Successfully
Not Subscribed ❌', ); } else
{ // Subscribe ke semua topic
bool subscribed = await
mqttService.subscribeToAllTop-
ics(); setState(() { _isSub-
scribed = subscribed; }); No-
tificationService.showNotifi-
cation( title: 'MQTT Success-
fully subscribed', body: 'Suc-
cessfully All Topics Sub-
scribed ☑️',

} } Widget _buildText-
Field(String label, String
value, Function(String) on-
Saved) { return Padding( pad-
ding: const
EdgeInsets.only(bottom: 10),
child: TextFormField( ini-
tialValue: value, decoration:
InputDecoration( labelText:
label, border: OutlineIn-
putBorder(), ), validator:
(value) => value == null ||
value.isEmpty ? 'Please enter
$label' : null, onSaved:
(value) => onSaved(value!), ),
); } Widget _build-
LastSavedSettings() { return
Column( crossAxisAlignment:
CrossAxisAlignment.start,

children: [ Text( 'Last Saved
Settings:', style:
TextStyle(fontWeight: Font-
Weight.bold, fontSize: 16), ),
SizedBox(height: 5),
Text('Broker Address: $_bro-
kerAddress'), Text('Port:
$_port'), Text('Client ID:
$_clientId'), Text('Topic :
$_topic'), ], ); } @override
Widget build(BuildContext con-
text) { return Scaffold( ap-
pBar: AppBar( title:
Text('MQTT Settings'), back-
groundColor: Colors.orange, ),
body: SingleChildScrollView(
child: Padding( padding: const
EdgeInsets.all(16.0), child:
Column( children: [ Form( key:
_formKey, child: Column( chil-
dren: [ _buildTextField('Bro-
ker Address', _brokerAddress,
(value) => _brokerAddress =
value), _buildText-
Field('Port', _port, (value)
=> _port = value), _buildText-
Field( 'Client ID', _clientId,
(value) => _clientId = value),
_buildTextField('Topic',
_topic, (value) => _topic =
value), SizedBox(height: 20),
Row( mainAxisAlignment:
MainAxisAlignment.spaceBe-
tween, children: [ Expanded(
child: ElevatedButton( on-
Pressed: _saveSettings, style:
ElevatedButton.styleFrom(

backgroundColor: Colors.blue),
child: Text('Save Settings'),
), ), SizedBox(width: 10),
```

```
Expanded( child: ElevatedBut-
ton( onPressed: _toggleConnec-
tion, style: ElevatedBut-
ton.styleFrom( background-
Color: _isConnected ? Col-
ors.red : Colors.green, ),
child: Text(_isConnected ?
'Disconnect' : 'Connect'), ),
), SizedBox(width: 10), Ex-
panded( child: ElevatedButton(
onPressed: _isConnected ?
_toggleSubscribe : null,
style: ElevatedButton.style-
From( backgroundColor: _isSub-
scribed ? Colors.red : Col-
ors.green, ), child: Text(_is-
Subscribed ? 'Unsubscribe All'
: 'Subscribe All'), ), ), ],
), SizedBox(height: 20), _is-
Saved ? _buildLastSavedSet-
tings() : Text('No settings
saved yet.', style:
TextStyle(color: Col-
ors.grey)), SizedBox(height:
20), Text( _isConnected ?
'MQTT Status: Connected 🕷' :
'MQTT Status: Disconnected
🔘', style: TextStyle( font-
Weight: FontWeight.bold,

fontSize: 16, color: _isCon-
nected ? Colors.green : Col-
ors.red, ), ),
SizedBox(height: 10), Text(
_isSubscribed ? 'Subscription
Status: All Topics Subscribed
☑' : 'Subscription Status:
Not Subscribed ❌', style:
TextStyle( fontWeight: Font-
Weight.bold, fontSize: 16,
color: _isSubscribed ? Col-
ors.green : Colors.red, ), ),
], ), ), ], ), ), ), ); } }
```

weather.dart

```
import 'dart:convert'; import
'package:http/http.dart' as
http; class WeatherService {
final String apiKey =
'8ede7f8b710ac896e472a47f83700
ac9'; // Ganti dengan API key
Anda final String city = 'Se-
marang'; // Ganti dengan kota
yang diinginkan

Future<Map<String, dynamic>>
getWeather() async { final url
= Uri.parse(
'https://api.openweather-
map.org/data/2.5/weather?q=$ci
ty&appid=$apiKey&units=met-
ric', ); try { final response
= await http.get(url); if (re-
sponse.statusCode == 200) {
return jsonDecode(re-
sponse.body); } else

throw Exception('Failed to
load weather data'); } } catch
(e) { print('Error fetching
weather data: $e'); throw Ex-
ception('Failed to load
weather data'); } } }
```

notf_setting.dart

```
import 'package:flutter_lo-
cal_notifications/flutter_lo-
cal_notifications.dart'; class
NotificationService { static
final FlutterLocalNotifica-
tionsPlugin _flutterLocalNoti-
ficationsPlugin = FlutterLo-
calNotificationsPlugin();
static Future<void> initial-
ize() async { const AndroidI-
nitializationSettings initial-
izationSettingsAndroid = An-
droidInitializationSet-
tings('@mipmap/ic_launcher');
final InitializationSettings
initializationSettings = Ini-
tializationSettings( android:
initializationSettingsAndroid,
); await _flutterLocalNotifi-
cationsPlugin.initialize(ini-
tializationSettings); } static
Future<void> showNotification(
{required String title, re-
quired String body}) async {
const AndroidNotificationDe-
tails androidPlatformChannel-
Specifics = AndroidNotifica-
tionDetails( 'your_chan-
nel_id', 'your_channel_name',
channelDescription:
'your_channel_description',
importance: Importance.max,
priority: Priority.high, show-
When: false, ); const Notifi-
cationDetails
```

```
platformChannelSpecifics = No-
tificationDetails(android: an-
droidPlatformChannelSpecif-
ics); await _flutterLocalNoti-
ficationsPlugin.show( 0, ti-
tle, body, platformChannelSpe-
cifics, p

); } }
```

mqtt_service.dart

```
// mqtt_service.dart import
'dart:async'; import
'dart:convert'; import
'dart:io'; import 'pack-
age:flutter/services.dart';
import 'package:mqtt_cli-
ent/mqtt_client.dart'; import
'package:mqtt_cli-
ent/mqtt_server_client.dart';
import 'package:shared_prefer-
ences/shared_prefer-
ences.dart'; enum MqttConnec-
tionState { disconnected, con-
necting, connected, reconnect-
ing, error } class MqttService
{ static final MqttService
_instance = MqttService._in-
ternal(); factory
MqttService() => _instance;
MqttService._internal(); late
MqttServerClient client;
MqttConnectionState _connec-
tionState = MqttConnection-
State.disconnected; MqttCon-
nectionState get connection-
State => _connectionState;
bool get isConnected => _con-
nectionState == MqttConnec-
tionState.connected; String?
topic; // Connection parame-
ters String? _broker; int?
_port; String? _clientId;
String? _topic; String?
_username; String? _password;
// Auto reconnect properties
Timer? _reconnectTimer; Timer?
_keepAliveTimer; Timer? _con-
nectionTimeoutTimer; bool
_shouldAutoReconnect = true;
int _reconnectAttempts = 0;

int _maxReconnectAttempts =
10; Duration _reconnectDelay =
const Duration(seconds: 5);
Duration _connectionTimeout =
const Duration(seconds: 30);
// Stream controllers final
StreamController<String> _mes-
sageController = StreamCon-
troller<String>.broadcast();
final StreamControl-
ler<MqttConnectionState> _con-
nectionStateController =
StreamController<MqttConnec-
tionState>.broadcast();
Stream<String> get mes-
sageStream => _messageControl-
ler.stream; Stream<MqttConnec-
tionState> get connection-
StateStream => _connection-
StateController.stream; //
Connection methods Fu-
ture<bool> connect({ required
String broker, required int
port, required String clien-
tId, required String topic,
String? username, String?
password, bool autoReconnect =
true, int maxReconnectAttempts
= 10, Duration reconnectDelay
= const Duration(seconds: 5),
}) async { // Store connection
parameters _broker = broker;
_port = port; _clientId = cli-
entId; _topic = topic;
_username = username; _pass-
word = password; _shouldAuto-
Reconnect = autoReconnect;
_maxReconnectAttempts = maxRe-
connectAttempts; _reconnectDe-
lay = reconnectDelay; return
await _attemptConnection(); }
Future<bool> _attemptConnec-
tion() async { if (_broker ==
null || _port == null || _cli-
entId == null) { print('❌
Connection parameters not
set'); _updateConnection-
State(MqttConnectionState.er-
ror); return false; } if
(_connectionState == MqttCon-
nectionState.connecting ||
_connectionState == MqttCon-
nectionState.reconnecting) {

print('⏳ Connection already
in progress'); return false; }
try { _updateConnection-
State(_reconnectAttempts > 0 ?
```

```
MqttConnectionState.reconnecting : MqttConnectionState.connecting); print('🔄 ${_reconnectAttempts > 0 ? "Reconnecting" : "Connecting"} to MQTT broker: $_broker:$_port'); // Cleanup previous client await _cleanupClient(); // Create new client client = MqttServerClient(_broker!, _clientId!); client.port = _port!; client.logging(on: false); client.keepAlivePeriod = 20; client.connectTimeoutPeriod = 30000; // 30 seconds client.autoReconnect = false; // We handle reconnection manually // Set callbacks client.onDisconnected = _onDisconnected; client.onConnected = _onConnected; client.onSubscribed = _onSubscribed; client.onAutoReconnect = _onAutoReconnect; client.onAutoReconnected = _onAutoReconnected; // Setup connection message final connMessage = MqttConnectMessage() .withClientIdentifier(_clientId!) .withWillQos(MqttQos.atMostOnce) .startClean(); if (_username != null && _password != null) { connMessage.authenticateAs(_username!, _password!); } client.connectionMessage = connMessage; // Set connection timeout _startConnectionTimeout(); // Attempt connection await client.connect(); _cancelConnectionTimeout(); final isConnected = client.connectionStatus!.state == MqttConnectionState.connected; if (isConnected) { _updateConnectionState(MqttConnectionState.connected); _reconnectAttempts = 0; await _subscribeToTopics(); _startKeepAliveMonitoring();

print('☑ MQTT Connected successfully!'); } else { _updateConnectionState(MqttConnectionState.error);

print('❌ MQTT Connection failed'); } return isConnected; } catch (e) { _cancelConnectionTimeout(); _updateConnectionState(MqttConnectionState.error);

print('❌ Connection error: $e'); if (_shouldAutoReconnect && _reconnectAttempts < _maxReconnectAttempts) { _scheduleReconnect(); } return false; } } void _startConnectionTimeout() { _connectionTimeoutTimer?.cancel(); _connectionTimeoutTimer = Timer(_connectionTimeout, () { print('⏰ Connection timeout'); _updateConnectionState(MqttConnectionState.error); client.disconnect(); if (_shouldAutoReconnect && _reconnectAttempts < _maxReconnectAttempts) { _scheduleReconnect(); } }); } void _cancelConnectionTimeout() { _connectionTimeoutTimer?.cancel(); _connectionTimeoutTimer = null; } void _onConnected() { print('☑ MQTT Connected callback triggered'); _updateConnectionState(MqttConnectionState.connected); _reconnectAttempts = 0; } void _onDisconnected() { print('🔌 MQTT Disconnected callback triggered'); _updateConnectionState(MqttConnectionState.disconnected); _stopKeepAliveMonitoring(); if (_shouldAutoReconnect && _reconnectAttempts < _maxReconnectAttempts) { _scheduleReconnect(); } else if (_reconnectAttempts >= _maxReconnectAttempts) {

print('🔴 Max reconnection attempts ($_maxReconnectAttempts) reached'); _updateConnectionState(MqttConnectionState.error); } } void _onSubscribed(String topic) { print('📡 Subscribed to:
```

```
$topic'); } void _onAutoReconnect() { print('🔄 Auto reconnect triggered by mqtt_client'); } void _onAutoReconnected() { print('☑ Auto reconnected by mqtt_client'); _updateConnectionState(MqttConnectionState.connected); } void _scheduleReconnect() { if (!_shouldAutoReconnect) return; _reconnectTimer?.cancel(); _reconnectAttempts++; // Exponential backoff with jitter final backoffDelay = Duration( seconds: (_reconnectDelay.inSeconds * (_reconnectAttempts * 0.5)).round().clamp(5, 60) ); print('🔄 Scheduling reconnect in ${backoffDelay.inSeconds}s (Attempt $_reconnectAttempts/$_maxReconnectAttempts)'); _reconnectTimer = Timer(backoffDelay, () async { if (_shouldAutoReconnect && !isConnected) { await _attemptConnection(); } }); } void _startKeepAliveMonitoring() { _stopKeepAliveMonitoring(); _keepAliveTimer = Timer.periodic(const Duration(seconds: 30), (timer) { if (!isConnected) { timer.cancel(); return; } // Check if connection is still alive if (client.connectionStatus?.state != MqttConnectionState.connected) { print('💔 Connection lost detected by keep-alive monitor'); _onDisconnected(); } else { // Send ping to test connection

try { client.publishMessage('ping/test', MqttQos.atMostOnce, MqttClientPayloadBuilder().addString('ping').payload!); } catch (e) { print('💔 Keep-alive ping failed: $e'); _onDisconnected(); } }); } void _stopKeepAliveMonitoring() { _keepAliveTimer?.cancel(); _keepAliveTimer = null; } void _updateConnectionState(MqttConnectionState newState) { if (_connectionState != newState) { _connectionState = newState; _connectionStateController.add(newState); print('📊 Connection state changed to: ${newState.toString().split('.').last}'); } } // Subscription methods Future<bool> _subscribeToTopics() async { if (!isConnected) { print('❌ Cannot subscribe, MQTT is not connected'); return false; } try { final prefs = await SharedPreferences.getInstance(); topic = prefs.getString('Topic') ?? _topic; if (topic != null) { client.subscribe(topic!, MqttQos.exactlyOnce); _startMessageListener(); print('📡 Subscribed to: $topic'); return true; } else { print('⚠ No topic found to subscribe'); return false; } } catch (e) { print('❌ Subscription error: $e'); return false;

}

} void _startMessageListener() { client.updates?.listen( (List<MqttReceivedMessage<MqttMessage?>>? messages) { if (messages != null) { for (var message in messages) { final receivedTopic = message.topic; final payload = MqttPublishPayload.bytesToStringAsString( (message.payload as MqttPublishMessage).payload.message); _handleReceivedMessage(receivedTopic, payload); } } }, onError: (error) { print('❌ Error receiving messages: $error'); if (_shouldAutoReconnect) { _onDisconnected(); } }, cancelOnError: false, ); } void _handleReceivedMessage(String receivedTopic, String payload)
```

```
{ print('📨 Received: "$payload" from: "$receivedTopic"'); // Save message to history _saveMqttData(receivedTopic, payload); // Send to stream if (receivedTopic == topic) { if (!_messageController.isClosed) { _messageController.add(payload); } } } // Publishing methods Future<bool> publishMessage(String pubTopic, String message) async { if (!isConnected) { print('❌ Cannot publish, MQTT not connected. Attempting reconnect...'); if (await _attemptConnection()) { return await publishMessage(pubTopic, message); } return false; } try {

final builder = MqttClientPayloadBuilder(); builder.addString(message); client.publishMessage(pubTopic, MqttQos.atLeastOnce, builder.payload!); print('📤 Published: "$message" to "$pubTopic"'); return true; } catch (e) { print('❌ Publish error: $e'); return false; } } // Data persistence methods Future<void> _saveMqttData(String saveTopic, String payload) async { try { final prefs = await SharedPreferences.getInstance(); final historyKey = 'mqtt_history_$saveTopic'; List<String> history = prefs.getStringList(historyKey) ?? []; final mqttData = { 'topic': saveTopic, 'payload': payload, 'timestamp': DateTime.now().toIso8601String(), }; history.add(jsonEncode(mqttData)); // Keep only last 1000 messages to prevent storage bloat if (history.length > 1000) { history = history.sublist(history.length - 1000); } await prefs.setStringList(historyKey, history); print('💾 Data saved: $saveTopic'); } catch (e) { print('❌ Error saving data: $e'); } } Future<List<Map<String, dynamic>>> getSavedHistory(String historyTopic) async { try { final prefs = await SharedPreferences.getInstance(); final history = prefs.getStringList('mqtt_history_$historyTopic') ?? []; return history.map((item) => jsonDecode(item) as Map<String, dynamic>).toList(); } catch (e) { print('❌ Error getting history: $e'); return []; } } Future<void> clearHistory(String historyTopic) async {

try { final prefs = await SharedPreferences.getInstance(); await prefs.remove('mqtt_history_$historyTopic'); print('🗑️ History cleared for: $historyTopic'); } catch (e) { print('❌ Error clearing history: $e'); } } // Control methods Future<bool> reconnect() async { print('🔄 Manual reconnect requested'); _reconnectAttempts = 0; return await _attemptConnection(); } void setAutoReconnect(bool enabled) { _shouldAutoReconnect = enabled; print('🔧 Auto reconnect ${enabled ? "enabled" : "disabled"}'); if (!enabled) { _reconnectTimer?.cancel(); } } void setMaxReconnectAttempts(int attempts) { _maxReconnectAttempts = attempts; print('🔧 Max reconnect attempts set to: $attempts'); } void setReconnectDelay(Duration delay) { _reconnectDelay = delay; print('🔧 Reconnect
```

```
delay set to:
${delay.inSeconds}s'); }
Future<bool>
unsubscribeFromTopic(String
unsubTopic) async { if
(!isConnected) { print('❌
Cannot unsubscribe, MQTT not
connected'); return false; }
try {
client.unsubscribe(unsubTopic)
; print('🔕 Unsubscribed
from: $unsubTopic'); return
true; } catch (e) { print('❌
Unsubscribe error: $e');
return false; }

} // Cleanup methods Fu-
ture<void> _cleanupClient()
async { try { if (client.con-
nectionStatus?.state ==
MqttConnectionState.connected)
{ client.disconnect(); } }
catch (e) { print('⚠️ Error
during client cleanup: $e'); }
} Future<void> disconnect()
async { print('🔌 Disconnect-
ing MQTT...'); _shouldAutoRe-
connect = false; _recon-
nectTimer?.cancel();
_keepAliveTimer?.cancel();
_connectionTimeoutTimer?.can-
cel(); await _cleanupClient();
_updateConnection-
State(MqttConnectionState.dis-
connected); print('☑️ MQTT
Disconnected'); } void dis-
pose() { print('🗑️ Disposing
MQTT Service...'); _shoul-
dAutoReconnect = false; _re-
connectTimer?.cancel();
_keepAliveTimer?.cancel();
_connectionTimeoutTimer?.can-
cel(); if (isConnected) { cli-
ent.disconnect(); } if (!_mes-
sageController.isClosed) {
_messageController.close(); }
if (!_connectionStateControl-
ler.isClosed) { _connection-
StateController.close(); }
print('☑️ MQTT Service dis-
posed'); } // Utility methods
String getConnectionSta-
tusString() { switch
(_connectionState) { case
MqttConnectionState.connected:
r

qttConnectionState.connecting:
return 'Connecting...'; case
MqttConnectionState.reconnect-
ing: return 'Reconnecting...
($_reconnectAttempts/$_maxRe-
connectAttempts)'; case
MqttConnectionState.discon-
nected: return 'Disconnected';
case MqttConnectionState.er-
ror: return 'Connection Er-
ror'; } } Map<String, dynamic>
getConnectionInfo() { return {
'broker': _broker, 'port':
_port, 'clientId': _clientId,
'topic': topic, 'state':
getConnectionStatusString(),
'autoReconnect': _shouldAuto-
Reconnect, 'reconnec-
tAttempts': _reconnec-
tAttempts, 'maxReconnec-
tAttempts': _maxReconnec-
tAttempts, }; } }
```

Lampiran 3.Surat Kesediaan Membimbing TA Pembimbing 1

SURAT KESEDIAAN MEMBIMBING TA

Yang bertanda tangan dibawah ini :

Nama : Bahrun Niam, M.T
NIPY : 09.015.277
Jabatan : Dosen Tetap

Dengan ini menyatakan bersedia untuk menjadi Pembimbing I pada Tugas Akhir Mahasiswa berikut :

Nama	: Septyawan Bagus Wibowo
NIM	: 24014012
Program Studi	: DIII Teknik Elektronika
Judul LaporanTugas Akhir	: Sistem Monitoring Kekeruhan Air Berbasis Scada dan IoT Dengan Antarmuka Android Menggunakan Sensor *Endress and Hauserr*

Demikian Pernyataan ini dibuat agar dapat dilaksanakan sebagaimana mestinya.

Semarang, 23 Juni 2025

Mengetahui,

Ketua Program Studi DIII Teknik Elektronika	Calon Dosen Pembimbing I,
Rony Darpono, M.T	Bahrun Niam, M.T
NIPY.09.015.282	NIPY. 09.015.277

Lampiran 4. Surat Kesediaan Membimbing TA Pembimbing 2

SURAT KESEDIAAN MEMBIMBING TA

Yang bertanda tangan dibawah ini :

Nama : Qirom, S.Pd, M.T
NIPY : 09.015.281
Jabatan : Dosen Tetap

Dengan ini menyatakan bersedia untuk menjadi Pembimbing I pada Tugas Akhir Mahasiswa berikut :

Nama	: Septyawan Bagus Wibowo
NIM	: 24014012
Program Studi	: DIII Teknik Elektronika
Judul Laporan Tugas Akhir	: Sistem Monitoring Kekeruhan Air Berbasis Scada dan IoT Dengan Antarmuka Android Menggunakan Sensor *Endress and Hauserr*

Demikian Pernyataan ini dibuat agar dapat dilaksanakan sebagaimana mestinya.

Semarang, 23 Juni 2025

Mengetahui,

Ketua Program Studi DIII Teknik Elektronika	Calon Dosen Pembimbing II,
Rony Darpono, M.T	Qirom, S.Pd, M.T
NIPY.09.015.282	NIPY.09.015.281

Lampiran 5. Form Bimbingan Pembimbing 1

FORM BIMBINGAN
TUGAS AKHIR

NAMA : SEPTYAWAN BAGUS WIBOWO

NIM : 24014012

JUDUL TA : SISTEM MONITORING KEKERUHAN AIR BERBASIS *SCADA* DAN *IOT* DENGAN ANTARMUKA *ANDROID* MENGGUNAKAN SENSOR *ENDRESS AND HAUSER*

Pembimbing I

No.	Hari/Tanggal	Uraian	Tanda Tangan
1	Senin, 10/4/2025	Revisi Judul TA	
2	Selasa, 11/4/2025	Bimbingan Laporan Tugas Akhir Ke-1	
3	Senin, 21/4/2025	Bimbingan Laporan Tugas Akhir Revisi Pada Logo dan Bab II	
4	Senin, 2/6/2025	Bimbingan Laporan Tugas Akhir dan Revisi Pada Bab III untuk pembuatan flowchart	
5	Selasa, 10/6/2025	Laporan TA ACC	

Lampiran 6. Form Bimbingan Pembimbing 2

FORM BIMBINGAN
TUGAS AKHIR

NAMA : SEPTYAWAN BAGUS WIBOWO

NIM : 24014012

JUDUL TA : SISTEM MONITORING KEKERUHAN AIR BERBASIS SCADA DAN IOT DENGAN ANTARMUKA ANDROID MENGGUNAKAN SENSOR *ENDRESS AND HAUSER*

Pembimbing II

No.	Hari/Tanggal	Uraian	Tanda Tangan
1	Selasa, 11/3/2025	Bimgingan Revisi Judul TA	
2	Jumat, 21/3/2025	Bimbingan Bab I & Bab II	
3	Jumat, 9/5/2025	Bimbingan Bab III	
4	Jumat, 23/5/2025	Bimbingan Bab IV dan cara pengambilan data	
5	Rabu, 28/5/2025	Bimbingan Bab IV dan V serta kunjungan ke ASB	

Lampiran 7. Form Revisi Laporan Tugas Akhir Ketua Penguji

FORM REVISI
UJIAN TUGAS AKHIR

NAMA : SEPTYAWAN BAGUS WIBOWO

NIM : 24014012

JUDUL TA : SISTEM MONITORING KEKERUHAN AIR BERBASIS SCADA DAN IOT DENGAN ANTARMUKA ANDROID MENGGUNAKAN SENSOR *ENDRESS AND HAUSER*

Ketua Penguji

No.	Hari/Tanggal	Uraian	Tanda Tangan
1	16 Juli 2025	Revisi perambahan Data uji	
2			
3		ACC	
4			
5			

Ketua Penguji

M. Sobri Suyhor.

Lampiran 8. Form Revisi Laporan Tugas Akhir Penguji 1

FORM REVISI
UJIAN TUGAS AKHIR

NAMA : SEPTYAWAN BAGUS WIBOWO

NIM : 24014012

JUDUL TA : SISTEM MONITORING KEKERUHAN AIR BERBASIS SCADA DAN IOT DENGAN ANTARMUKA ANDROID MENGGUNAKAN SENSOR *ENDRESS AND HAUSER*

Penguji II

NNo.	Hari/Tanggal	Uraian	Tanda Tangan
1	16 Juli 2025	Revisi gambar dan keterangannya serta urutan pembuatan aplikasi	
2			
3		Acc	
4			
5			

Penguji 1

.....

Rony Dapaw MT

Lampiran 9. Form Revisi Laporan Tugas Akhir Penguji 2

FORM REVISI
UJIAN TUGAS AKHIR

NAMA : SEPTYAWAN BAGUS WIBOWO

NIM : 24014012

JUDUL TA : SISTEM MONITORING KEKERUHAN AIR BERBASIS SCADA DAN IOT DENGAN ANTARMUKA ANDROID MENGGUNAKAN SENSOR *ENDRESS AND HAUSER*

Penguji II

No.	Hari/Tanggal	Uraian	Tanda Tangan
1	9 Juli 2025	Revisi Bahasa asing & Gambar serta penomoran Halaman	
2	10 Juli 2025	Revisi flow chart & penambahan teori untuk sensor turbid	
3	14 Juli 2025	Acc	
4			
5			

Penguji 2

Dony Kuncoro, M.T

09.015.278