Implementasi Kekeruhan Berbasis Sistem Komunikasi Can Bus pada Sensor Turbidity

Proyek Akhir

Oleh:

Maudy Arumdhyanita Ayu (3232101080)

Program Studi Teknik Instrumentasi
Jurusan Teknik Elektro
Politeknik Negeri Batam
2024

Pernyataan Keaslian Proyek Akhir

Saya yang bertandatangan di bawah ini menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya yang berjudul : "Implementasi kekeruhan berbasis Sistem komunikasi Can bus pada sensor turbidity." adalah **hasil karya sendiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan,** dan **bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.** Semua referensi yang dikutip atau dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan saya ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Batam, 24 Januari 2024

Maudy Arumdhyanita Ayu
NIM: 3232101080

i

Lembar Pengesahan

Tugas Akhir disusun untuk memenuhi salah satu syarat memperoleh gelar
Ahli Madya Teknik (AMd.T.)
Di
Politeknik Negeri Batam

oleh:
Maudy Arumdhyanita Ayu (3232101080)
Aldi Masqury (3232101012)
Ramadani Syafitri (3232101010)

Tanggal Sidang : Rabu, 10 Januari 2024

Disetujui oleh :

1. Hasnira,S.ST.,M.Tr.T.
NIK: 113112

1. Muhammad Jaka Wimbang Wicaksono.S.T.,M.T
NIK: 122272

2. Dessy Oktani, S.T., M.T.
NIK: 110075

2. Asrizal Deri Futra, S.Si, M.Si
NIK: 115133

Implementasi kekeruhan berbasis Sistem komunikasi Can Bus pada sensor turbidity

Abstrak

Proyek ini merupakan penelitian yang dilatarbelakangi belum diwajibkan kapal berukuran di bawah 100 GT memiliki *oily water saparator* (OWS*)* yang di dalam nya terdapat *oil in water* sebagai pengukur kekeruhan dalam air bilga (saluran buangan air, minyak dan pelumas hasil proses mesin kapal). Penelitian ini bertujuan untuk mengurangi pencemaran air yang diakibatkan oleh air bilga (saluran buangan air, minyak pelumas dari hasil proses mesin kapal) dan mendeteksi nilai kekeruhan pada air bilga. Metode penelitian yang dilakukan adalah metode kuantitatif yang difokuskan melakukan suatu percobaan mendeteksi kekeruhan dalam air menggunakan sensor *turbidity* yang dapat mengukur nilai kekeruhan dari 0 – 3.000 NTU. Pada pengujian data sensor *turbidity* , digunakan alat pembanding yaitu *turbidity meter* TU-2016 berdasarkan spesifikasi datasheet tersebut mampu mengukur nilai kekeruhan sebesar 0 – 1.000 NTU dan resolusi nya adalah 0,01 NTU. Kemudian data dari sensor *turbidity* akan dikomunikasikan menggunakan komunikasi RS-485, CAN Bus, dan TCP/IP. Alat ini juga dirancang untuk menampilkan nilai kekeruhan pada sensor *turbidity* dapat tampil pada LCD dan website. Hasil dari penelitian ini diharapkan alat yang kami buat dapat berguna untuk mengetahui kadar kekeruhan pada air dan dapat dikomunikasikan menggunakan RS-485, CAN Bus dan TCP/IP.

Kata kunci: Sensor *turbidity*, RS-485, CAN Bus dan TCP/IP

Implementation of turbidity based on Rs-485 Can bus and TCP/IP communication system on turbidity sensor

Abstract

This project is a research that is motivated by the background that ships under 100 GT are not required to have an oily water saparator (OWS) in which there is oil in water as a turbidity gauge in bilga water (water exhaust, oil and lubricant from ship engine processes). This study aims to reduce water pollution caused by bilga water (sewerage, lubricating oil from ship engine processes) and detect turbidity values in bilga water. The research method carried out is a quantitative method focused on conducting an experiment to detect turbidity in water using turbidity sensors that can measure turbidity values from 0 – 3,000 NTU. In testing turbidity sensor data, a comparison tool is used, namely the turbidity meter TU-2016 based on the datasheet specifications is able to measure turbidity values of 0-1,000 NTU and the resolution is 0.01 NTU. Then the data from the sensor will be communicated using RS-485, CAN Bus, and TCP/IP communication. This tool is also designed to display the turbidity value on the turbidity sensor can be displayed on LCD and websites. The results of this research are expected that the tools we make can be useful to determine the level of turbidity in water and can be communicated using RS-485, CAN Bus and TCP / IP.

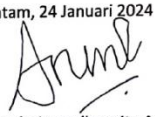
Keywords: Turbidity sensor, RS-485, CAN Bus and TCP/IP

Kata Pengantar

Puji syukur kami ucapkan kepada Tuhan yang Maha Esa atas berkat dan rahmat-Nya kami dapat menyelesaikan laporan Proyek Akhir yang berjudul "Implementasi kekeruhan berbasis Sistem komunikasi Rs-485 Can Bus dan TCP/IP pada sensor turbidity". Penulis menyelesaikan laporan Proyek Akhir untuk memenuhi salah satu persyaratan guna memperoleh gelar Ahli Madya Teknik (Amd.T) kelulusan di Politeknik Negeri Batam Jurusan Teknik Elektro Program Studi Teknik Instrumentasi. Penulisan Laporan Proyek Akhir ini disusun dan diselesaikan dengan baik berkat dukungan dan bantuan para dosen, rekan mahasiswa dan dukungan dari banyak pihat yang ikut dalam membimbing penulis untuk menyelesaikan Laporan Proyek Akhir ini. Pada kesempatan ini, penulis mengucapkan terimakasih kepada:

1. Tuhan Yang Maha Esa yang telah melimpahkan karunia dan Rahmat-Nya serta meridai jalannya kegiatan proyek akhir, sehingga penulis dapat menyelesaikan proyek akhir dengan tepat waktu
2. Kedua orangtua dan keluarga atas Doa, bimbingan dan nasehat.
3. Bapak Uuf Brajawidagda, S.T., M.T., Ph.D., selaku Direktur Politeknik Negeri Batam.
4. Bapak Dr. Budi Sugandi, S.T., M.Eng., selaku Ketua Jurusan Teknik Elektro.
5. Bapak Ir.Kamarudin, S.T.M.T.,IPM selaku Ketua Program Studi Jurusan Teknik Instrumentasi, Dosen Wali dan Dosen Pembimbing
6. Bapak Asrizal Deri Futra, S.Si, M.Si selaku Dosen Manajer Proyek .
7. Seluruh Bapak/Ibu Dosen Teknik Elektro dan teman-teman Teknik Instrumentasi Politeknik Negeri Batam yang ikut serta membantu.

Batam, 24 Januari 2024



Maudy Arumdhyanita Ayu

v

Daftar Isi

Daftar Gambar

Daftar Tabel

Bab 1. Pendahuluan

1.1. Latar Belakang

Indonesia merupakam salah satu negara kepulauan yang telah ditetapkan oleh Konvensi UNCLOS 1982, terdiri dari 17.504 pulau dan 2/3 wilayah Indoneisa adalah lautan. Lautan merupakan tempat tinggal bagi berbagai macam biota, baik biota air yang dibudidayakan maupun yang tidak dibudidayakan. Indonesia memiliki kekayan laut yang sangat melimpah. Lautan Indonesia mengalami penuruna kualitas air salah satu nya adalah pembuangan limbah kapal yang tidak sesuai peraturan Indonesia atau dunia. Oleh karena itu, sangat penting bagi kita untuk merawat serta menjaga lautan [1].

Banyaknya pencemaran di laut salah satu hasil tumpahan minyak hampir dari semua kapal yang berasal dari air bilga (saluran buangan air, minyak pelumas dari hasil proses mesin kapal). Salah satu data dampak dari tumpahan minyak yaitu pada tanggal 11 Januari 2019 dinas lingkungan hidup kota Parepare melaporkan bahwa perairan Cempae tercemar rembesan solar dan ikan mati [2]. Selain itu, bedasarkan Kementrian Kelautan dan Perikanan (2019) laut Indonesia sudah tercemar limbah minyak, mulai dari kasus bocornya sumur Pertamina sekitar 3.000 barel perhari [3]. Oleh karena itu, International Maritime Organization (IMO) yang merupakan organisasi untuk mencegah terjadinya pencemaran laut, membuat aturan MARPOL (Marine Police) yang berupa batasan pembuangan limbah bilga dari hasil proses mesin kapal ke laut. Selain itu, berdasarakan buku petunjuk dari gas cleaning systems nilai kekeruhan maksimum dalam air buangan tidak boleh lebih besar dari 25 NTU (Unit Turbiditas Nefometrik) atau unit setara, di atas kekeruhan air masuk [4].

Pencemaran air laut yang disebabkan berbagai factor menyebabkan air laut menjadi keruh. Pencemaran air laut yang tinggi dapat mempengaruhi proses fotosintesis dan reproduksi plankton, serta kehidupan biota laut lainnya [5]. Oleh demikian, Proyek ini dibuat untuk menciptakan sebuah produk sederhana yang mampu mengetahui nilai kekeruhan dan nilai kekeruhan tersebut akan di komunikasikan.

Dengan adanya inovasi ini diharapkan dengan mudah menampilkan nilai kekeruhan pada larutan limbah yang sebagai salah satu upaya agar nantinya proses hasil mesin kapal yang akan dibuang ke laut tidak banyak mengandung limbah yang telah ditentukan oleh pemerintah, dan juga dapat menggunakan tiga jenis sistem komunikasi yang berbeda. Beberapa contoh komunikasi yang dapat digunakan antara lain adalah RS-485, CAN Bus dan TCP/IP.

1.2. Rumusan Masalah

Adapun rumusan masalah yang dapat diangkat dari latar belakang yang telah diketahui yaitu sebagai berikut :

1. Bagaimana membuat sistem sederhana untuk mendeteksi nilai kekeruhan pada larutan menggunakan protokol komunikasi RS-485, CAN Bus dan TCP/IP?
2. Bagaimana keuntungan dan kekurangan masing-masing metode komunikasi RS-485, CAN Bus, dan TCP/IP?
3. Bagaimana implementasi komunikasi RS-485, CAN BUS, dan TCP/IP dalam mendeteksi kekeruhan menggunakan sensor turbidity
4. Bagaimana pengujian komunikasi RS-485,CAN Bus, dan TCP/IP

1.3. Tujuan

Tujuan-tujuan yang mendasari pembuatan proyek akhir ini adalah:

1. Membangun sistem sederhana Implementasi kekeruhan berbasis Sistem komunikasi Rs-485, Can Bus dan TCP/IP pada sensor turbidity
2. Menganalisis keuntungan dan kekurangan masing-masing metode komunikasi RS-485,CAN Bus dan TCP/IP
3. Mengimplementasikan komunikasi RS-485, CAN Bus, dan TCP/IP dalam mendeteksi kekeruhan menggunakan sensor turbidity
4. Melakukan pengujian komunikasi RS-485,CAN Bus, dan TCP/IP

1.4. Manfaat

Secara Teoritis memberikan sumbangsih keilmuan, terkhusus pada ilmu teknik mengenai produk “Implementasi kekeruhan berbasis Sistem komunikasi Rs-485 Can Bus, dan TCP/IP pada sensor turbidity” yang dapat mendeteksi kekeruhan pada larutan dengan pengembangan sistem komunikasi. Sebagai bahan informasi untuk penelitian lain dalam hal pengembangan serta mendalami sensor tubidity yang dapat menganalisis kekeruhan yang disebabkan oleh pencemaran dengan basis komunikasi yang digunakan yaitu: RS-485, CAN Bus dan TCP/IP.

1.5. Batasan

Dalam pembuatan tugas akhir ini ada beberapa hal yang menjadi Batasan. Adapun Batasan yang di tetapkan sebagai berikut:

1. Pengujian dilakukan dengan sampel asap rokok 25 – 50 NTU.
2. Mikrokontroler menggunakan Arduino Uno
3. Sistem komunikasi menggunakan RS-485 *Shield,* sistem komunikasi TCP/IP menggunakan *Ethernet Shield*, dan sistem komunikasi CAN

Bus *Shield*

4. Sensor *turbidity* memiliki kapasitas untuk mengukur nilai kekeruhan larutan yaitu 0 - 3.000 NTU.
5. Penggunaan sensor *turbidity* harus di tempat yang gelap, karena cahaya sangat mempengaruhi nilai kekeruhan.

1.6. *Work Breakdown Structure*

Tabel 1. Work Break Structure

No	Nama	Tugas dan Tanggung Jawab dalam Tim
1.	Ramadani Syafitri	**Implementasi kekeruhan berbasis Sistem komunikasi TCP/IP pada sensor turbidity** Koordinator Tim - Pemograman komunikasi TCP/IP, - Pengujian komunikasi TCP/IP menggunakan sensor *turbidity* SEN0189-DFRobot. - Perancangan mekanikal
2.	Aldi Masqury	**Implementasi kekeruhan berbasis Sistem komunikasi RS-485 pada sensor turbidity** - Akuisis data sensor kekeruhan yaitu sensor *turbidity* SEN0189-DFRobot. - Pemograman komunikasi RS-485. - Pengujian komunikasi RS-485 menggunakan sensor *turbidity* SEN0189-DFRobot. - Perancangan elektrikal.
3.	Maudy Arumdhyanita Ayu	**Implementasi kekeruhan berbasis Sistem komunikasi CAN Bus pada sensor turbidity** - Pemograman komunikasi CAN Bus. - Pengujian komunikasi CAN Bus menggunakan sensor *turbidity* SEN0189-DFRobot.

Bab 2. Tinjauan Pustaka

2.1. Kondisi Lingkungan

Pencemaran air laut berdasarkan UU No 32 Tahun 2009 yang dimaksud pencemaran laut adalah masuknya atau dimasukannya mahkluk hidup, zat, energi, dan atau komponen lain kedalam laut oleh kegiatan manusia atau oleh alam sehingga kualitas air laut menurun sampai ketingkat tidak berfungsi lagi sebagai peruntukannya [5]. Unsur – unsur dari pencemaran laut diantaranya adalah:

1. Kegiatan pelayaran, aktivitas bongkar muat tengah laut dan balasting, ketika kapal berlabuh yang memungkinkan jangkar kapal mengenai pipa bawah laut dan terjadi kebocoran pipa.
2. Kegiatan pengeboran, sedikit banyak menghasilkan residu yang dapat mencemari lingkungan laut terlebih terjadi kebocoran pipa.
3. Kegiatan penyulingan, sisa dari penyulingan yang menjadi sumber utama pencemaran air laut terlebih jika penyulingan dilakukan di tengah laut.
4. *Oil spill*, tumpahan minyak yang mengakibatkan pencemaran akibat dari hasil operasional kapal tangker, perbaikan, atau perawatan kapal, proses bongkar muat tengah laut dan bocornya pipa minyak bawah laut serta kecelakaan kapal.

Minyak mentah mengandung hydrocarbon sekitar 50% sampai 98% dan sisanya adalah sulfur, nitrogen, oksigen, dan beberapa logam berat. Pencemaran minyak diakibatkan aktifitas pelayaran sekaligus karena aktifitas pengeboran seperti kebocoran pipa bawah laut, kebocoran kapal dan oil spill [6]. Hal tersebut telah menimbulkan bahaya bagi lingkungan laut baik secara lokal maupun secara luas yang berimbas pada rusaknya ekosistem biota laut dengan terjadinya kekeruhan oleh limbah tersebut. Oleh demikian, sistem Implementasi kekeruhan akan dapat mendeteksi kekeruhan pada air yang disebabkan oleh limbah yang terjadi akibat sisa pembuangan air bilga kapal yang dibuang kelaut, sesuai dengan peraturan international yang telah dibuat oleh MARPOL (*Marine Police*) dan *Guidelines For Exhaust Gas Cleaning Systems.*

Dari beberapa penelitian yang telah dikaji, didapatkan ide-ide untuk pengembangan sistem Implementasi kekeruhan berbasis Sistem komunikasi Rs-485, Can bus dan TCP/IP pada sensor turbidity. Dalam pengembangan ini digunakan alat dan bahan yang diperlukan untuk menjalankan proyek ini dengan baik. Sensor *turbidity* digunakan untuk mendeteksi kekeruhan air yang disebabkan oleh limbah yang dimana proyek ini menggunakan sampel limbah

yang terlarut. Sensor turbidity akan mendeteksi kekeruhan larutan dan data akan dikirimkan ke mikrokontroler Arduino UNO , dan selanjutnya menggunakan komunikasi, sistem komunikasi yang digunakan seperti RS-485,CAN Bus dan TCP/IP. Data dari kekeruhan tersebut akan ditampilkan melalui sistem antarmuka, baik itu Liquid Crystal Display (LCD) maupun Personal Computer (PC).

2.2. Gambaran Teknologi Alat yang Digunakan

2.2.1. Sensor

Sensor merupakan salah satu kompenen bagian dari transduser dan bertugas untuk mendeteksi atau merasakan dan menangkap adanya perubahan energi eksternal dari tranduser, kapasitas energi yang didapatkan akan dikirimkan ke *converter transduser* diubah menjadih energi listrik. Sensor bertugas menangkap perubahan kapasitas energi yang terdeteksi dan segera mengirimkannya ke bagian konverter transduser untuk diubah menjadi energi listrik [7]

Gambar 1. Sensor Turbidity

Sensor yang digunakan dalam projek Implementasi kekeruhan adalah sensor turbidity SEN0189-DFRobot. Sensor *turbidity* digunakan untuk mengukur tingkat kekeruhan dalam sebuah cairan, yang merupakan indikator jumlah partikel padat atau zat tersuspensi yang ada dalam cairan tersebut. Prinsip kerja sensor *turbidity* didasarkan pada interaksi cahaya dengan partikel-partikel padat atau zat tersuspensi dalam cairan. Semakin banyak partikel dalam air menunjukan tingkat kekeruhan air juga tinggi. Semakin tinggi tingkat kekeruhan air akan diikuti oleh perubahan dari tegangan output sensor [8]

2.2.2. Mikrokontroller

MIkrokontroler merupakan *chip* yang di dalamnya terdapat prosesor, memori dan *input* / *output* dimana memiliki fungsi yaitu sebagai pengontrol rangkaian elektronik dan juga otak dari sebuah alat yang yang dapat di program oleh pengguna atau pembuat program [9]

Berdasarkan namanya, "Uno" diambil dari bahasa Italia yang berarti "satu". Hal ini menunjukkan bahwa Arduino Uno adalah versi 1.0 dari produk tersebut

dan akan terus berkembang pada versi-versi selanjutnya, serta menggambarkan bahwa Arduino UNO sesuai dengan desain yang diinginkan.

Papan mikrokontroler Arduino Uno didasarkan pada ATmega328. Arduino Uno memiliki 14 pin *input/output digital*, di mana 6 pin *input* tersebut dapat digunakan sebagai *output* PWM, dan 6 pin *input analog*. Arduino UNO juga dilengkapi dengan osilator kristal 16 MHz, koneksi USB, *jack power*, *header* ICSP, dan tombol *reset*. Untuk mengoperasikan mikrokontroler, kita hanya perlu menghubungkan Board Arduino UNO ke komputer menggunakan kabel USB atau ke sumber listrik dengan adaptor AC-*to*-DC atau baterai [10].

Setiap pin digital pada Arduino UNO dapat digunakan sebagai *input* atau *output* dengan menggunakan fungsi pin *Mode, digitalWrite,* dan *digitalRead*. Fungsi-fungsi tersebut beroperasi pada tegangan 5 *volt.* Setiap pin dapat memberikan atau menerima arus maksimum 40 mA dan memiliki resistor *pull-up* sebesar 20-50 kOhm yang terputus secara *default* [11]

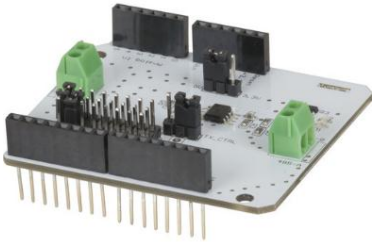
Gambar 2. Arduino UNO

Arduino UNO merupakan sistem berbiaya rendah dan berdaya rendah pada rangkaian chip mikrokontroler berbasis ATmega328. Arduino UNO diprogram khusus untuk mengolah hasil pembacaan sensor dan juga melakukan komunikasi data. Pada Implementasi kekeruhan mikrokontroler yang digunakan adalah Arduino UNO yang berjumah tiga buah. Arduino UNO akan diletakkan di dalam kotak yang digunakan untuk membaca atau memproses data dari sensor *turbidity* yang akan dikirimkan ke komunikasi RS-485, CAN Bus dan TCP/IP.

2.2.3. Sistem Komunikasi

Sistem komunikasi adalah berupa perangkat keras atau perangkat lunak yang dirancang untuk mengomunikasikan informasi dari suatu objek ke objek lainnya. RS-485 merupakan salah satu antarmuka yang komunikasi datanya dilakukan secara serial yang dapat berfungsi hingga jarak 1.200 meter. RS-485 salah satu komunikasi *multipoint* yang dijalankan oleh *master* dan *slave*. *Master* mampu

terhubung dengan 32 *slave* sekaligus diwaktu yang sama hanya menggunakan dua buah kabel yaitu A dan B. RS-485 mengeluarakan sinyal A lebih dari 200 mV dari sinyal b maka bernilai *high* dan jika sinyal B lebih tinggi maka nilai sinyal B bernilai *high* dan jika keduanya bernilai di bawah 200 mV maka nilai tersebut tidak terdefinisi. Di mana nilai sensor akan terbaca oleh komunikasi menggunakan RS 485 [12]



Gambar 3. RS-485 *Shield*

Penelitian ini memilih penggunaan RS-485 karena memiliki beberapa kelebihan, seperti biaya yang rendah untuk membuatnya, dapat menghubungkan banyak perangkat sekaligus, tahan terhadap interferensi, dan memiliki jangkauan jarak yang luas. Ini berarti bahwa setiap *transmitter* atau penerima terhubung ke pengirim atau penerima melalui satu jalur yang saling berhubungan.

Komunikasi TCP/IP merupakan komunikasi *wireless* yang dimana dapat mengirim data melalui jaringan internet. TCP/IP (*Transmission Control Protocol / Internet Protocol*) adalah sebuah standar komunikasi data yang digunakan oleh komunitas internet untuk pertukaran data antara komputer-komputer di dalam jaringan Internet. Protokol ini menggunakan alamat IP6 (IP *address*) sebagai skema pengalamatan yang sederhana, memungkinkan ratusan juta komputer dapat saling terhubung di Internet. Fungsi utama dari TCP/IP adalah protokol standar yang bertanggung jawab saat proses tukar menukar data pada suatu jaringan berlangsung [13]

Gambar 4. *Ethernet Shield*

Dalam komunikasi menggunakan TCP/IP, digunakan Arduino *Ethernet Shield* sebagai modul yang menghubungkan Arduino dengan internet. Arduino *Ethernet Shield* berperan penting dalam memungkinkan koneksi Arduino ke jaringan internet modul ini secara efektif mendukung protokol TCP/IP.

Komunikasi CAN (*Control Area Network*) adalah sebuah protokol komunikasi yang menggabungkan beberapa jalur ke dalam satu jalur BUS. Fungsinya adalah untuk mengatur dan mengendalikan aliran data yang masuk dan keluar. MCP2515 adalah sebuah perangkat *Controller Area Network* (CAN) yang digunakan untuk mempermudah koneksi antara Arduino dengan fisik CAN Bus. MCP2515 terdiri dari modul CAN, logika kontrol, dan protokol SPI (*Serial Peripheral Interface*) .

Gambar 5. *MCP2515 CAN BUS Shield For Arduino*

Berfungsi agar Arduino dapat berkomunikasi menggunakan CAN Bus dengan sensor oksigen sebagai inputan. CAN Bus merupakan sebuah jaringan *multi-master* yang menggunakan pesan dengan prioritas tinggi. Jaringan ini menggunakan protokol serial untuk mengirim dan menerima data dalam format bilangan heksadesimal dengan panjang maksimum 8 bit. Tujuan dari jaringan ini adalah untuk menghubungkan sensor-sensor dan mikrokontroler dengan aktuator-aktuator lainnya. Komunikasi CAN Bus merupakan komunikasi yang

bergantung pada jangkauan kabel yang dimana data dapat terkirim maksimal 500 m .

2.2.4. Tampilan

LCD adalah papan informasi (*display elektronik*) yang menampilkan data, seperti karakter, huruf, atau grafik. LCD dibuat dengan teknologi CMOS *logic* dan memantulkan cahaya di sekelilingnya ke depan dan mentransmisikan data. Cara kerja LCD yaitu, apabila elektroda diaktifkan dengan medan listrik (tegangan), molekul organik yang panjang dan silindris menyesuaikan diri dengan elektroda dari segmen. Lapisan *sandwich* memiliki polarizer cahaya vertikal depan dan horisontal belakang, dan lapisan reflektor kemudian mengikutinya. Cahaya yang dipantulkan tidak dapat melewati molekul-molekul yang telah menyesuaikan diri, sehingga segmen yang diaktifkan menjadi gelap dan membentuk karakter data yang ingin ditampilkan [14]

Gambar 6. LCD I2C 20x4

Web Server adalah *software* yang memberikan layanan data yang mempunyai fungsi untuk menerima permintaan HTTP (*Hyper Text Transfer Protocol*) dan tampilan perangkat lunak yang memberi layanan data pada Web atau *browser* melalui Ip *address* yang disediakan oleh *Ethernet shield*.

Gambar 7. Tampilan Web Server

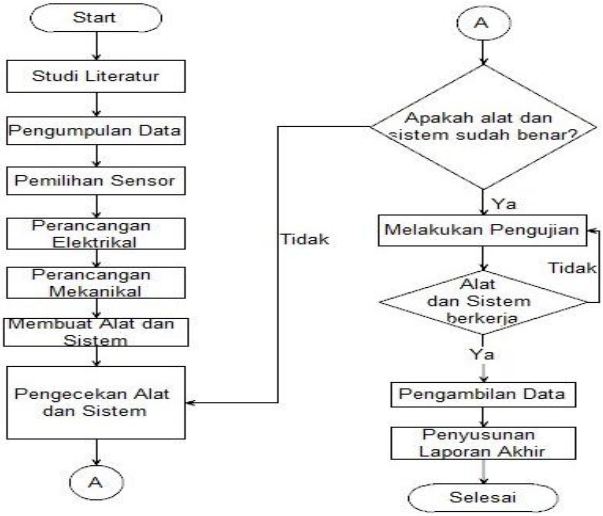
2.2.5. Sofware Arduino IDE

IDE merupakan singkatan dari *Integrated Development Environment*, yang pada dasarnya adalah sebuah lingkungan terintegrasi yang digunakan untuk melakukan pengembangan. Istilah "lingkungan" digunakan karena melalui perangkat lunak ini, pemrogram dapat memprogram Arduino untuk menjalankan berbagai fungsi yang telah tertanam menggunakan sintaks pemrograman. IDE (*Ingrated Development Environment*) yang diperuntukan untuk membuat perintah atau *source code*, melakukan pengecekan kesalahan,kompilasi,*upload* program, dan menguji hasil kerja Arduino melalui *serial monitor*. Arduino menggunakan bahasa pemrograman yang mirip dengan bahasa C [15].

Gambar 8. Arduino IDE

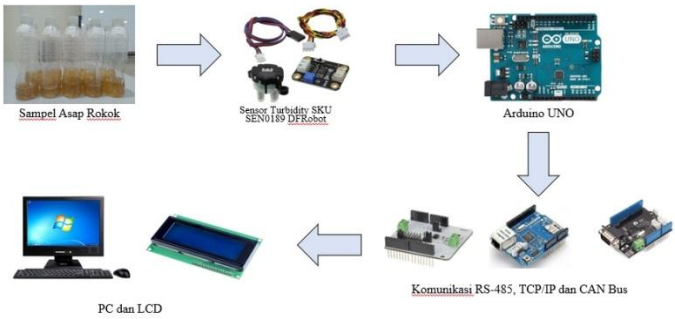
Bab 3. Metode Pelaksanaan

3.1. Perancangan Umum



Gambar 9. Diagram Alir Kegiatan

Perancangan sistem pada penelitian ini terdiri dari beberapa bagian. Diantaranya dimulai dari mempelajari studi literatur, pengumpulan data, pemilihan sensor, perancangan elektrikal, perancangan mekanikal, membuat alat dan sistem, melakukan *troubleshooting* jika terjadinya masalah pada sistem saat pengecekan, melakukan pengujian, pengambilan data, dan penyusunan buku. Dapat dilihat pada gambar diagram alir kegiatan.

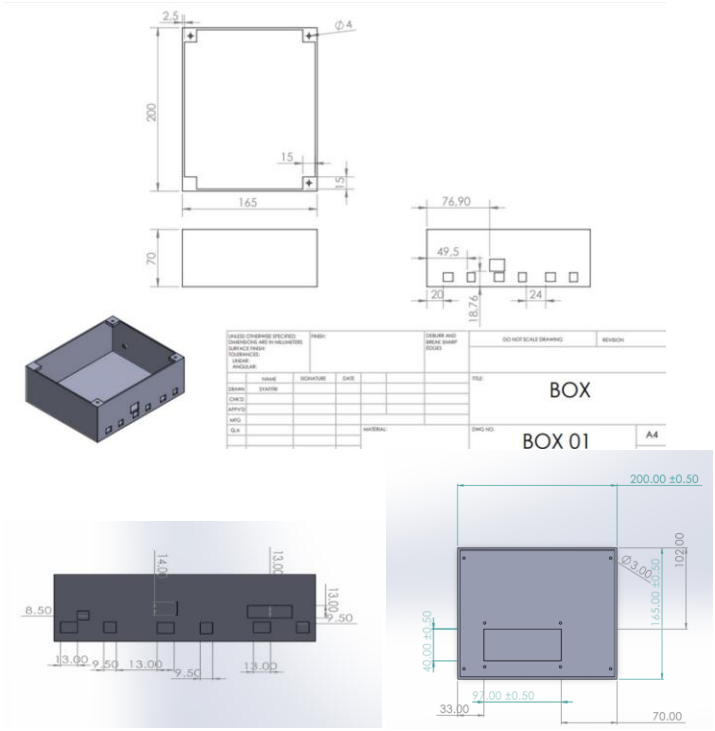


Gambar 10. Diagram Kerja Sistem

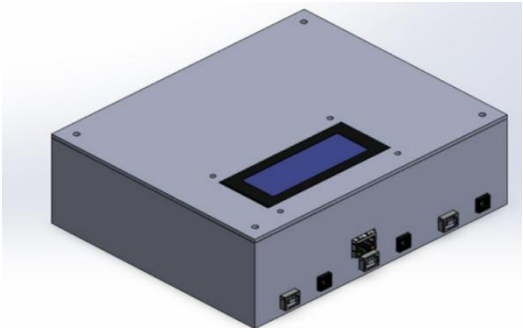
Pada diagram kerja sistem yang telah dibuat, dimulai dari mempersiapkan sampel asap rokok, selanjutnya dilakukan pengujian sensor *turbidity* mendeteksi kekeruhan pada sampel, kemudian data pada sampel dikelola oleh mikrokontroler Arduino UNO, lalu mikrokontroler mengirimkan data ke masing - masing komunikasi RS-485,CAN BUS dan TCP/IP dan data ditampilkan melalui LCD dan PC.

3.1.1. Perancangan Desain Mekanikal

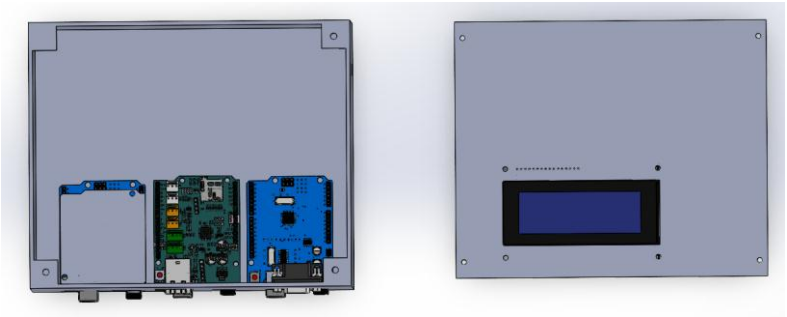
Desain mekanikal dibuat menggunakan perangkat *Solid Works*, dengan produk yang terdiri dari box berbahan filamen dengan pembuatan box menggunakan 3D printing dan terdapat dudukan komponen. Produk ini memiliki dimensi sebagai berikut: Box berukuran 20 cm (Panjang) x 16,5 cm (Lebar) x 7 cm (Tinggi). Seluruh komponen yang akan digunakan telah terpasang di dalam box tersebut. Fungsi dari box ini adalah untuk melindungi komponen yang ada didalamnya.



Gambar 11. Ukuran Box

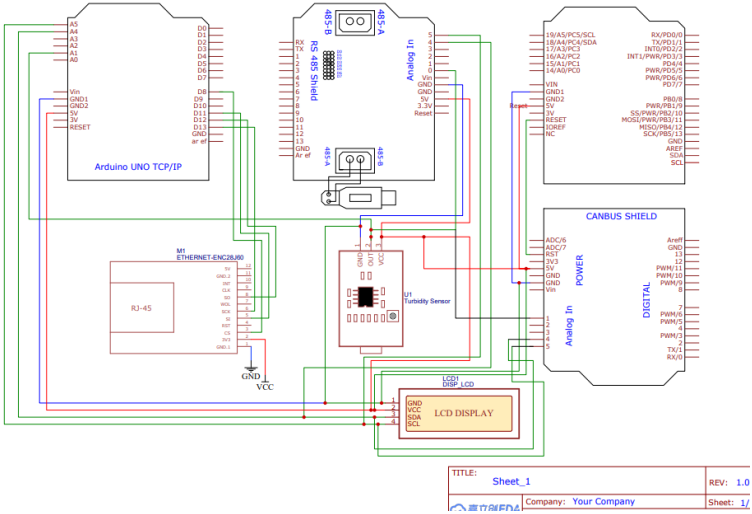


Gambar 12. Tampak Bagian Luar



Gambar 13. Tampak Bagian Dalam

3.1.2. Perancangan Desain Elektrikal



Gambar 14. Desain Elektrikal

Pada perancangan elektrikal Implementasi kekeruhan berbasis Sistem komunikasi Rs-485 Can bus dan TCP/IP pada sensor turbidity ini menggunakan aplikasi *Easyda* untuk membuat skema perancangan elektrikal.

Dalam perancangan elektrikal ini, terdapat 3 buah Arduino yang berfungsi sebagai mikrokontroler. Setiap Arduino terhubung melalui RS-485 *Shield*,CAN BUS *Shield* dan *Ethernet Shield*. Sensor mengirim data ke Arduino dengan inputan A0. Data Arduino terkoneksi ke masing masing komunikasi, selanjutnya komunikasi RS-485 *Shield* dan CAN BUS *Shield* mengirim data sensor yang ditampilkan ke LCD. *Ethernet Shield* digunakan untuk melakukan komunikasi TCP/IP melalui kabel LAN dengan perangkat lain. Tujuannya adalah untuk membuka sebuah *website* yang akan menampilkan nilai sensor.

Dengan demikian, ketiga Arduino dalam perancangan ini berperan penting dalam pengiriman dan penerimaan data melalui berbagai modul dan *shield* yang terkoneksi secara kabel dan serial untuk berkomunikasi dengan berbagai perangkat eksternal.

3.2. Alat dan Bahan

Dalam pelaksanaan penelitian ini dibutuhkan alat dan bahan. Berikut alat dan bahan yang dibutuhkan dapat dilihat pada tabel di bawah ini.

Tabel 2. Alat dan Bahan

No.	Alat dan Bahan	Fungsi
1	*Software Arduino IDE*	Berfungsi untuk membuat program perintah ke mikrokontroler Arduino UNO.
2	*Software Solidworks 2020*	Berfungsi untuk mendesain jenis desain mekanikal 3D.
3	*Software Easyda*	Berfungsi sebagai sarana untuk mendesain jenis desain elektrikal.
5	Module RS485 *Shield V2.1*	Berfungsi sebagai komunikasi serial yang mampu mengirimkan nilai sensor ke LCD atau *device* lain dengan rentang jarak yang jauh.
6	USB RS-485	Berfungsi sebagai adaptor Konverter USB ke RS485
6	CAN BUS *Shield* MCP2515	Berfungsi sebagai penghubung agar Arduino dapat berkomunikasi melalui CAN-BUS dan mengirimkan nilai sensor.
7	*Ethernet Shield*	Berfungsi sebagai komunikasi yang mampu mengirimkan nilai sensor kedalam website atau suatu laman internet.
8	Sensor Turbidity	Berfungsi mendeteksi kekeruhan pada larutan
9	Arduino UNO	Fungsinya adalah sebagai inti atau pusat dari sistem

		kendali dan sistem elektrik dalam rangkaian.
10	LCD (*Liquid Circuit Display*)	Berfungsi sebagai sarana menampilkan data yang dihasilkan.

3.3. Pengujian

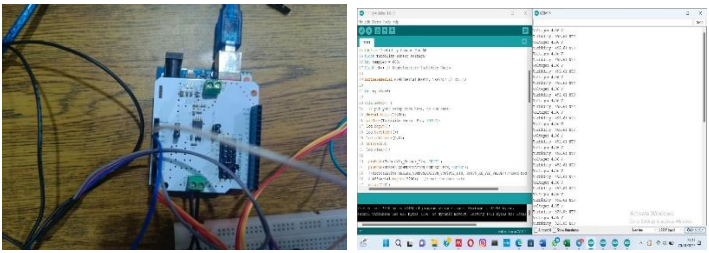
3.3.1. Pengujian Akuisisi Data Sensor Turbidity Menggunakan Arduino

Pengujian pada Metode pengujian sensor *turbidity* menggunakan sampel kekeruhan limbah. Tujuan pengujian sensor adalah untuk memperoleh persamaan linear melalui akuisisi data. Sensor ini mendeteksi partikel tersuspensi dalam air dengan cara mengukur transmitansi dan hamburan cahaya yang berbanding lurus dengan kadar *Total Suspended Solids* (TTS). Semakin tinggi kadar TTS, maka semakin tinggi pula tingkat kekeruhan air tersebut. Nilai dari sampel yang dihasilkan melalui sensor tersebut kemudian direkam oleh mikrokontroler.

Pengujian akan difokuskan pada masing-masing metode komunikasi yang digunakan dalam sistem tersebut. Tujuan dari pengujian ini adalah untuk mengevaluasi kelebihan dan kekurangan dari setiap jenis komunikasi yang digunakan dalam sistem Implementasi kekeruhan. Pengujian akan berbeda-beda untuk setiap jenis komunikasi, yaitu RS485,CAN Bus dan TCP/IP.

3.3.2. Pengujian Komunikasi RS-485 *Shield*

Pada pengujian Implementasi kekeruhan, terdapat komunikasi data menggunakan RS-485 *shield* V2.1 dan USB RS-485. Untuk cara kerjanya adalah Arduino UNO akan dihubungkan ke RS-485 *shield* V2.1. Kemudian sesuaikan elektrikalnya dan dihubungkan pada sensor *turbidity*. Pada komunikasi RS-485 *shield* V2.1 pada laptop pertama adalah sebagai master dan laptop kedua adalah slave. Hal yang pertama dilakukan adalah melakukan aturan serial monitor pada PC pertama dengan pastikan program sampai dengan *done uploading* dan kemudian atur port yang sesuai dengan port Arduino UNO yang dihubungkan ke laptop. Kemudian data tersebut akan dikirimkan ke PC lainnya. Data yang telah di kirimkan akan di terima oleh USB RS-485 dan akan ditampilkan pada serial monitor pada laptop kedua. USB RS-485 dihubungkan ke laptop lainnya dan kemudian diatur port sesuai dengan USB RS-485 yang dihubungkan di laptop lainnya. Pengujian ini dilakukan menggunakan data yang sama atau sampel yang sama pada sensor turbidity sensor.



Gambar 15. Proses Pengujian Komunikasi RS-485

3.3.3. Pengujian Komunikasi TCP/IP

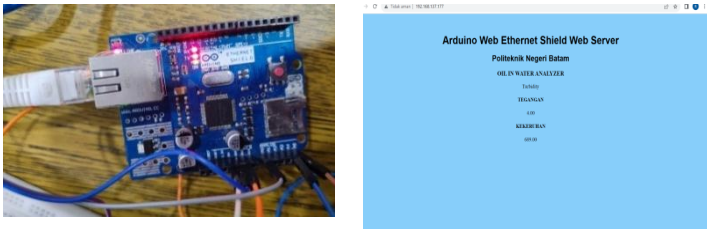
Pada sistem komunikasi TCP/IP ini menggunakan protokol web server, yang dimana server akan mentransfer data lalu web server akan menyediakan respon data dalam bentuk halaman web. Dengan menggunakan modul *Ethernet Shield* yang langsung terhubung ke Arduino. Sambungkan Arduino ke pin A0 dan GND pada sensor *turbidity*. Selanjutnya, hubungkan kabel LAN *Ethernet Shield* ke port kabel LAN pada laptop, dan hubungkan juga kabel transfer Arduino ke laptop.

Setelah koneksi fisik terbentuk, lakukan langkah-langkah berikut:

1. Pastikan Arduino dan laptop sudah terhubung. Buka control panel dan pilih "*Network and Internet*" kemudian pilih "*Network and Sharing Center*."
2. Pilih opsi "*Ethernet*" lalu klik *properties*. Pilih "*Internet Protocol Version TCP/IP 4*" dan pilih "*use the following IP address*." Atur pengaturan IP sesuai dengan ketentuan yang dibutuhkan, lalu klik "OK."

Pada Arduino:

1. Pilih program yang ingin dijalankan pada Arduino.
2. Pilih "*Tools*" dan atur port yang terhubung ke Arduino, kemudian klik "*Upload*" untuk mengunggah program ke Arduino.

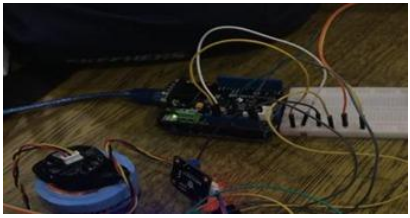


Gambar 16. Proses Pengujian Komunikasi TCP/IP

3.3.4. Pengujian Komunikasi CAN Bus

Pada alat Implementasi kekeruhan juga menggunakan komunikasi canbus, canbus yang digunakan merupakan canbus *shield* MCP2515 V1.0. Untuk sistem kerja alat akan dihubungkan pada sensor turbidity. Pada komunikasi canbus *shield* V1.0 hal pertama yang dilakukan adalah melakukan aturan *serial monitor* pada PCdan pastikan *port* sesuai dengan Arduino UNO lalu pada program dibuat perhitungan untuk diukur nilai tegangan dan kekeruhan, setelah itu upload program hingga berhasil. Lalu data akan ditampilkan pada serial monitor.

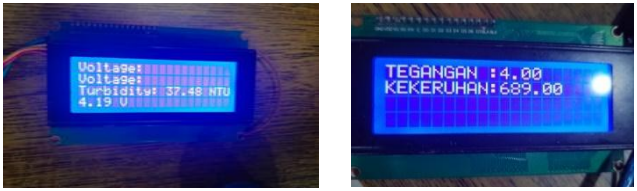
Pengujian ini dilakukan menggunakan 7 sampel dimana masing-masing sampel tersebut sudah diukur dengan menggunakan *turbidity meter*. Data yang akan keluar berupa tegangan dan kekeruhan.



Gambar 17. Proses Pengujian Komunikasi CAN Bus

3.3.5. Pengujian Data Tampil pada LCD

Pengujian data tampil pada LCD yaitu untuk menampilkan data sensor *turbidity* pada LCD. Sambungkan Arduino ke pin A0 dan GND pada sensor *turbidity* dan pada LCD pin SCL, SDA, VCC dan GND data sensor dan komunikasi akan tampil pada website dan serial monitor dari Arduino IDE dan juga terdapat tampilan pada layar LCD.



Gambar 18. Pengujian Data Tampil pada LCD

Bab 4. Hasil dan Pembahasan

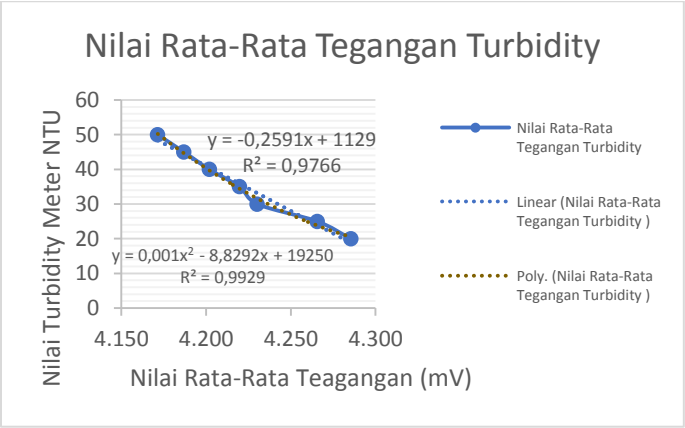
4.1. Hasil dan Pembahasan Akuisisi Data Sensor Turbidity Menggunakan Arduino

Pada Implementasi kekeruhan berbasis Sistem komunikasi Rs-485 Can bus dan TCP/IP pada sensor turbidity sudah melakukan beberapa tahapan pengembangan. Tahapan pengembangan yang sudah dilakukan desain elektrikal dan mekanikal. Sensor turbidity sudah mendapatkan nilai tegangan regresi secara linear dan juga secara polynomial pada sampel air rokok. Sampel air rokok tersebut dibuat secara mandiri dengan menggunakan 24 batang rokok. Kemudian, 1000 mL air yang dibagi menjadi 2 wadah yang masing masingnya adalah 500 mL air. Pembuatan sampel asap rokok dengan memasukkan asap rokok tersebut ke dalam botol yang penutup botol sudah dilubangi dan di letakkan di penutup botol tersebut. Setelah rokok dinyalakan dan diletakkan di penutup botol, maka kita menekan bagian badan botol agar asap rokok masuk kedalam botol. Setelah pengumpulan asap rokok pada botol dan rokok pun sudah habis maka ganti tutup botol dengan tanpa lubang. Lalu gerakkan botol tersebut sampai dengan asap rokok tidak ada lagi di dalam botol dan sudah tercampur dalam air mineral yang berisi 500 mL air. Berikut merupakan data hasil pengujian pengukuran nilai kekeruhan sensor *turbidity* dan nilai tegangan pada sensor yang diuji coba melalui Arduino UNO.

Tabel 3. Akuisisi Data Nilai Kekeruhan Turbidity Meter Terhadap Nilai Tegangan

No	Nama sampel	Nilai Turbidity Meter	Nilai Rata-Rata Tegangan Turbidity (mV)
1	Asap Rokok	50	4.171
2		45	4.187
3		40	4.202
4		35	4.220
5		30	4.230
6		25	4.266
7		20	4.285

Pada tabel 3 terdapat nilai kekeruhan pada alat ukur pembanding yaitu *turbidity meter* TU-2016 sebesar 20-50 NTU dan terdapat nilai tegangan sensor turbidity. Berikut adalah grafik Tegangan Sensor dan Nilai *Turbidity Meter* TU-2016



Gambar 19. Grafik Nilai Tegangan Sensor dan Nilai *Turbidity Meter* TU-2016

Gambar 19 merupakan grafik yang menunjukkan dimana pada nilai kekeruhan (*Turbidity Meter* TU-2016) yaitu 20-50 NTU semakin rendah kekeruhan pada larutan asap rokok maka semakin tinggi pula nilai tegangan yang dihasilkan pada sensor *turbidity*.

Tabel 4. Akuisisi Data Nilai Kekeruhan Turbidity Sensor Terhadap Nilai Tegangan

No	Nilai Turbidity Meter (NTU)	Rata-Rata Nilai Tegangan (mV)	Rata-Rata Nilai Sensor Turbidity (NTU)
1	50	3.414,4445	244,318
2	45	3.611,446	193,275
3	40	3.679,8385	175,554
4	35	3.798,284	144,8645
5	30	3.861,2295	128,5555
6	25	3.886,7505	121,9405
7	20	3.907,072	116,678

Pada tabel 4 didapatkan nilai rata-rata tegangan pada sensor *turbidity* setiap sampel yang di uji. Pada data rata-rata nilai tegangan sensor didapatkan semakin tinggi nilai tegangan tersebut maka semakin rendah nilai turbidity sensor dimana larutan pada sampel semakin jernih.

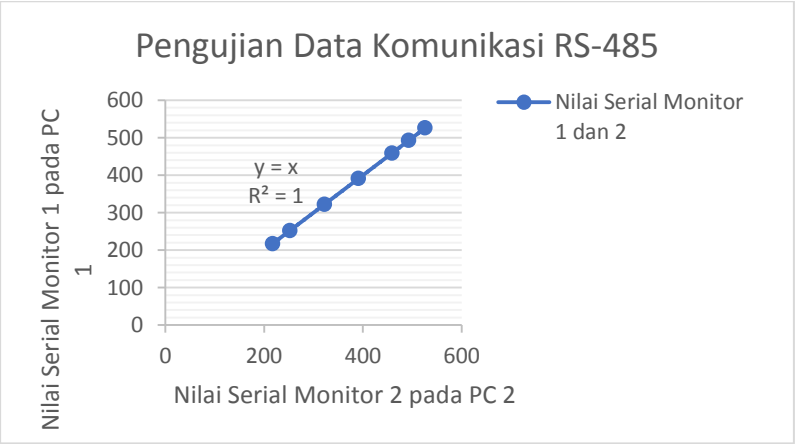
4.2. Hasil dan Pembahasan Komunikasi RS-485 *Shield* V2.I

Tabel 4 merupakan data dari hasil pengujian komunikasi RS-485 *Shield* menggunakan sensor *turbidity*.

Tabel 5. Pengujian Komunikasi RS-485 *Shield*

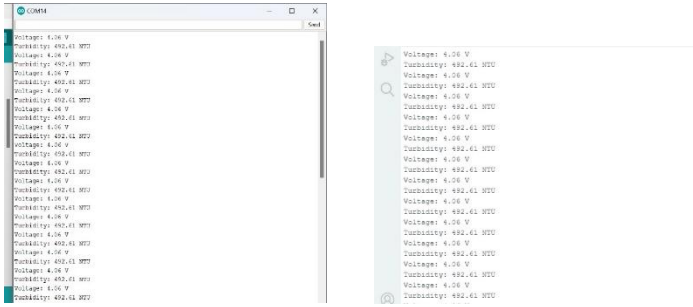
No	Serial Monitor 1 (NTU)	Serial Monitor 2 (NTU)	Tampilan LCD (NTU)
1	217,01	217,01	217,01
2	252,25	252,25	252,25
3	322,04	322,04	322,04
4	390,94	390,94	390,94
5	458,95	458,95	458,95
6	492,61	492,61	492,61
7	526,05	526,05	526,05

Berdasarkan pengujian komunikasi RS-485 *Shield* menggunakan sensor *turbidity*. Pada sampel 1 yang nilai 217,01 tampilan pada *serial monitor 1* sebagai master yang akan dikirimkan melalui USB RS-485 ke *serial monitor* 2 didapatkan hasil pada *serial monitor* 2 adalah 217,01. Tampilan pada LCD sama dengan 217,01 NTU. Pada *serial monitor* 1 sebagai *master* mengirimkan data dan akan diterima oleh *slave* dan data tersebut dengan keluaran yang sama yaitu nilai kekeruhan. Berdasarkan hasil pengujian komunikasi RS-485 *serial monitor 1* , *serial monitor 2* dan LCD keluaran data nya sama. Gambar 23 Merupakan grafik perbandingan komunikasi RS-485.



Gambar 20. Grafik Nilai Pengujian Data Komunikasi RS-485

Pada grafik nilai pengujian data komunikasi RS-485 didapatkan persamaan y = x dan R^2 = 1. Berdasarkan persamaan yang telah di dapatkan maka pengujian RS-485 sesuai dengan dimana fungsinya komunikasi ketika pengirim nilai keluaran sensor adalah 217,01 NTU maka sebagai penerima keluaran sensornya adalah 217,01 NTU.



Gambar 21. Tampilan Komunikasi RS-485 pada PC 1 dan 2

Pada gambar 21 merupakan tampilan pengujian pada PC 1 dan 2. Pada PC 1 didapatkan nilai kekeruhan sebesar 492,61 NTU dan nilai tegangan sebesar 4,06 volt begitu pula pada PC 2 yang data yang didapatkan adalah nilai kekeruhan sebesar 492,61 NTU dan nilai tegangan sebesar 4,06 volt.

4.3. Hasil dan Pembahasan Komunikasi CAN Bus

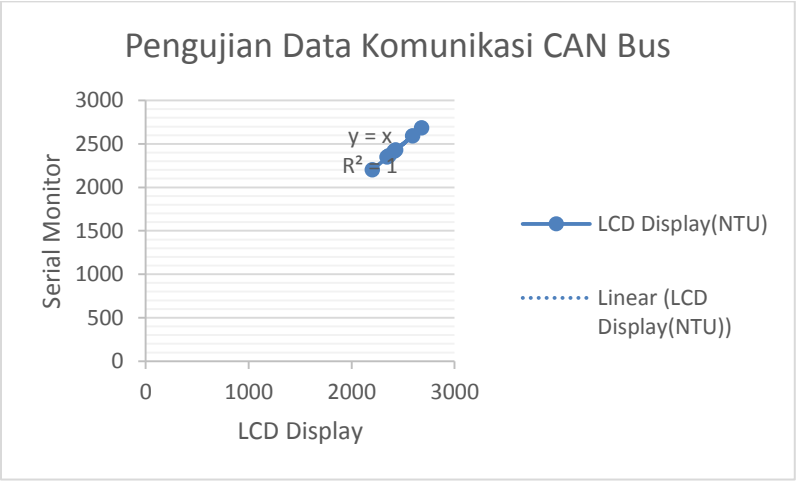
Tabel 6 merupakan data dari hasil pengujian komunikasi CAN BUS menggunakan sensor *turbidity.*

Tabel 6. Pengujian Komunikasi Canbus *Shield*

Sampel	Serial Monitor(NTU)	LCD Display(NTU)
1	2200,22	2200,22
2	2344,95	2344,95
3	2362,04	2362,04
4	2411,94	2411,94
5	2591,4	2591,4
6	2428,13	2428,13
7	2681,13	2681,13

No	Nilai Rata-Rata Tegangan		Nilai Rata-Rata Kekeruhan	
	CAN Bus	Arduino	CAN Bus	Arduino
1	3,404411765	3,425588235	2210,7706	2170,317941
2	3,979298221	3,193529412	3,3258824	2558,802647
3	3,317352941	3,348235294	2362,04	2313,110882
4	3,255151515	3,196969697	2466,7179	2553,853939
5	3,173939394	3,16	2585,9952	2604,9
6	3,313636364	3,466363636	2372,4012	2089,612727
7	3,099393939	3,140606061	2681,903	2661,822727

Berdasarkan Pengujian komunikasi Can bus *Shield* menggunakan sensor turbidity. Pada sampel 1 yang nilai 2200,22 tampilan pada Serial monitor, Kemudian pada tampilan LCD sama dengan 2200,22 NTU. Pada serial monitor akan mengirimkan data dan akan diterima oleh LCD dengan keluaran yang sama yaitu nilai kekeruhan dan tegangan. Berdasarkan hasil pengujian komunikasi Can bus *serial monitor* dan LCD keluaran data nya sama.



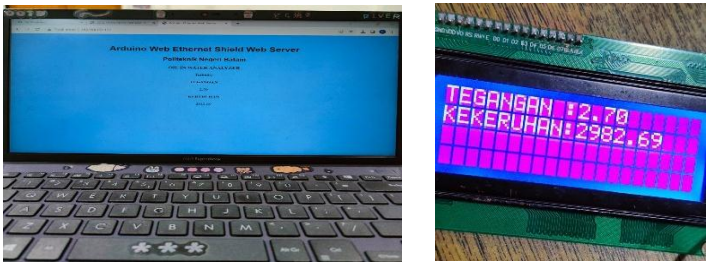
Gambar 22. Grafik Nilai Pengujian Data Komunikasi CAN Bus

Pada grafik nilai pengujian data komunikasi Canbus didapatkan persamaan y = x dan R^2 = 1. Berdasarkan persamaan yang telah di dapatkan maka pengujian Canbus sesuai dengan dimana fungsinya komunikasi ketika pengirim nilai

keluaran sensor adalah 2200,22 NTU maka sebagai penerima keluaran sensornya adalah 2200,22 NTU.

4.4. Hasil dan Pembahasan Komunikasi TCP/IP

Pada gambar 23 merupakan gambar dari tampilan web server dan LCD komunikasi TCP/IP.



Gambar 23. Tampilan Komunikasi TCP/IP

Ketika program Arduino UNO IDE di upload dan pada serial monitor akan mengeluarkan *IP address* yang akan kita bawa pada web, dan data sensor akan tampil melalui web server dan LCD secara bersamaan. Berikut tabel data keluaran dari web server dan LCD.

Tabel 5. Data keluaran Web Server dan LCD

No	Web Server		LCD	
	Tegangan (Volt)	Kekeruhan (NTU)	Tegangan (Volt)	Kekeruhan (NTU)
1	2.9	2876,3	2.9	2876,3
2	2.8	2940,7	2.8	2940,7
3	2.9	2876,3	2.9	2876,3
4	2.7	2982,69	2.7	2982,69
5	2.8	2940,7	2.8	2940,7
6	2.1	3000	2.1	3000
7	2.8	2940,7	2.8	2940,7

Bab 5. Kesimpulan dan Saran

5.1. Kesimpulan

Kesimpulan dari hasil pengembangan Implementasi kekeruhan berbasis Sistem komunikasi Rs-485 Can bus dan TCP/IP pada sensor turbidity pada sistem komunikasi RS485, CAN BUS dan TCP/IP adalah sebagai berikut:

1. Hasil data dari sensor turbidity masi jauh dari hasil dari alat pembanding sensor
2. TCP/IP menggunakan *Ethernet shield* mampu menghubungkan perangkat secara mudah dan error tidak terlalu tinggi ,hanya saja pada TCP/IP *real time* yang terjadi pada tampilan web terdapat selisih delay dengan tampilan real time pada LCD. Namun, di sisi lain, komunikasi TCP/IP memiliki kekurangan yang perlu diperhatikan, yaitu memerlukan bandwidth yang cukup besar, yang dapat menyebabkan proses pengiriman data menjadi lambat.
3. Sistem Komunikasi RS-485 menunjukkan persamaan y = x . Dimana Y adalah sebagai pengirim dan x adalah penerima dari hasil data yang dikelola oleh Arduino UNO sebagai mikrokontroler dan sensor turbidity sebagai sensor pengolah nilai kekeruhan pada suatu larutan. Pada proses pengiriman data baik karena pada saat pengiriman kecepatan mencapai 115200bps. Selain itu, RS-485 digunakan melaui panjang kabel sesuai dengan kapasitas daya *transfer* kepada penerima.
4. Hasil data dari sensor turbidity masih jauh dari hasil alat ukur pembanding sensor yaitu *turbidity meter* TU-2016. *Turbidity meter* TU-2016 memiliki kemampuan untuk mengukur nilai kekeruhan sebesar 0 – 1.000 NTU.
5. Melalui komunikasi RS-485 dan TCP/IP sudah dapat menampilkan data sensor yang tampil pada LCD

5.2. Saran

Penelitian ini tidak luput dari kekurangan penulis salah satunya adalah diharapkan untuk bisa mengkalibrasi sensor turbidity, walapun sensor tersebut sangat sensitive terhadap perubahan eksternal yaitu cahaya dan pembacaan dari komunikasi TCP/IP tergantung pada koneksi yang stabil antara pengirim dan penerima. Jika koneksi terputus, proses pengiriman data bisa terganggu, oleh karena itu membutuhkan teknologi yang canggih dan protokol keamanan tambahan. Untuk sistem komunikasi TCP/IP ini, penting untuk terus mengoptimalkan penggunaan *bandwidth* agar proses pengiriman data menjadi lebih cepat tanpa mengorbankan efisiensi. Selain itu, RS-485 juga harus diperhatikan kabel pengirim dan penerima RS-485 Shield terhadp pc sebagai penerima data sensor. Untuk sensor sendiri, harus diperhatikan apakah sensor

tersebut terkena Cahaya. Jika sensor terkena cahaya, memengaruhi nilai data yang dihasilkan.

Daftar Pustaka

[1] Y. Listiyono, L. Yudho Prakoso, D. Sianturi, S. P. Laut, S. Pertahanan, And U. Pertahanan, “Membangun Kekuatan Laut Indonesia Dipandang Dari Pengawal Laut Dan Detterence Effect Indonesia Building Indonesian Sea Power Based On The Indonesian Sea Guard And Detterent Effect.”

[2] Jadda, Asram AT, Sadriyah Mansur, and Kaswin Hartono Hamzah. "Peran Dinas Lingkungan dalam Pengendalian Pencemaran Akibat Tumpahan Minyak oleh Pertamina di Kota Parepare." Madani Legal Review 6.1 (2022): 1-20.

[3] M. D. Firmansyah, A. Ismanto, S. Y. Wulandari, R. Widiaratih, A. Rifai, And W. Atmodjo, “Pemodelan Sebaran Tumpahan Minyak Di Perairan Karawang, Jawa Barat,” *Buletin Oseanografi Marina*, Vol. 10, No. 2, Pp. 200–212, May 2021, Doi: 10.14710/Buloma.V10i2.31736.

[4] “2021 Guidelines For Exhaust Gas Cleaning Systems”.

[5] Y. H. Fernandez, L. Nauli, L. Toruan, And L. C. Soewarlan, “Tingkat Pencemaran Perairan Laut Di Pesisir Teluk Kupang, Nusa Tenggara Timur, Indonesia Pollution Level Of Sea Water On The Coast Of Kupang Bay, East Nusa Tenggara, Indonesia.” [Online]. Available: Https://Poluseajurnal.Ub.Ac.Id

[6] Widodo, BL Hentri, and Eni Tri Wahyuni. "Manajemen penanggulangan tumpahan minyak di laut akibat dari pengoperasian kapal." Majalah Ilmiah Gema Maritim 22.1 (2020): 60-66.

[7] M. Gilang Ganesha, M. Ikhsan Sani, And M. Lisda Meisaroh, “‘(Iot Gas Leakage Detector Based On Blynk).’”

[8] N. Kn, M. Alfaiz, And R. A. Prabowo, “Rancang Bangun System Recycle Limbah Air Berbasis Iot Dan Analisa Traffic Jaringan Menggunakan Wireshark”, Doi: 10.37817/Ikraith-Teknologi.V8i1.

[9] H. Badri, Z. Tharo, S. Aryza, P. Wibowo, And S. Anisah, “Rancangan Alat Pengaman Instalasi Listrik Menggunakan Sistem Proteksi Relay Terhadap Beban Lebih Dan Hubung Singkat Berbasis Mikrokontroler,” *Agustus*, Vol. 6, No. 3, 2022.

[10] D. Auliya Saputra, “Rancang Bangun Alat Pemberi Pakan Ikan Menggunakan Mikrokontroler,” 2020. [Online]. Available: Http://Jim.Teknokrat.Ac.Id/Index.Php/Teknikelektro/Index

[11] S. Samsugi, D. Gunawan, A. Thyo, And A. T. Prastowo, “Penerapan Penjadwalan Pakan Ikan Hias Molly Menggunakan Mikrokontroler Arduino Uno Dan Sensor Rtc Ds3231.”

[12] Muchamad Chadiq Zakaria, Edy Kurniawan, And Jawwad Sulthon H, “Sistem Monitoring Instrument Air Compressor (Iac) Berbasis Scada

Dengan Komunikasi Modbus Rtu Rs-485," *J-Eltrik*, Vol. 2, No. 2, P. 117, Nov. 2021, Doi: 10.30649/J-Eltrik.V2i2.117.

[13] S. Wardoyo, T. Ryadi, And R. Fahrizal, "06 Jurnal Nasional Teknik Elektro Analisis Performa File Transport Protocol Pada Perbandingan Metode Ipv4 Murni, Ipv6 Murni Dan Tunneling 6to4 Berbasis Router Mikrotik," 2014.

[14] M. Akbar, Prihadi Murdiyat, And M. A. Putra, "Rancang Bangun Peringatan Banjir Di Jalan Cipto Mangun Kusumo Berbasis Arduino," *Poligrid*, Vol. 3, No. 2, P. 41, Dec. 2022, Doi: 10.46964/Poligrid.V3i2.1710.

[15] G. Hergika, "Perancangan Internet Of Things (Iot) Sebagai Kontrol Infrastuktur Dan Peralatan Toll Pada Pt. Astra Infratoll Road," Vol. 8, No. 2, 2021, [Online]. Available: Https://Www.Esp8266.Com/Viewtopic.Php?P=68657

[1] Biodata

Nama : Aldi Masqury
TTL : Palembang, 24 Januari 2002
Agama : Islam
Alamat : Seraya Atas Blok C No 23 RT 002 RW 005 Kel. Kampung Pelita Kec. Lubuk Baja
Email
Riwayat Pendidikan SMA/SMK : SMAN 20 Batam
SMP : SMPN 6 Batam

Nama : Ramadani Syafitri
TTL : Batam, 05 Desember 2002
Agama : Islam
Alamat : Batam Center
Email :ramadanisyafitri20@gmail.com
Riwayat Pendidikan SMA/SMK : SMAN 2 Pariaman
SMP : SMPN 2 Pariaman

Nama : Maudy Arumdhyanita Ayu
TTL : Batam, 24 Februari 2003
Agama : Islam
Alamat : Perum. Cipta Sarana Indah 2 Blok B3 No 11
Email : maudyarum50@gmail.com
Riwayat Pendidikan SMA/SMK : SMAN 18 Batam
SMP : SMPN 11 Batam

Lampiran

Program Komunikasi RS-485

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 20, 4);

#include "Arduino.h"
#include <SoftwareSerial.h>
#define RXPin       3  // Serial Receive pin
#define TXPin       2  // Serial Transmit pin

//RS485 control
#define SERIAL_COMMUNICATION_CONTROL_PIN 5 // Transmission set pin
#define RS485_TX_PIN_VALUE HIGH
#define RS485_RX_PIN_VALUE LOW

#define Turbidity_Sensor_Pin A5
float Turbidity_Sensor_Voltage;
int samples = 600;
float ntu; // Nephelometric Turbidity Units

SoftwareSerial RS485Serial(RXPin, TXPin); // RX, TX

int byteSend;

void setup() {
  // put your setup code here, to run once:
 Serial.begin(19200);
 pinMode(Turbidity_Sensor_Pin, INPUT);
 lcd.begin();
 lcd.backlight();
 lcd.setCursor(0,0);
 delay(100);
 lcd.clear();

  pinMode(Turbidity_Sensor_Pin, INPUT);
  pinMode(SERIAL_COMMUNICATION_CONTROL_PIN, OUTPUT);
  //digitalWrite(SERIAL_COMMUNICATION_CONTROL_PIN,
RS485_RX_PIN_VALUE);//Read mode
  RS485Serial.begin(19200);   // set the data rate
```

```
      delay(100);

      digitalWrite(SERIAL_COMMUNICATION_CONTROL_PIN,
RS485_TX_PIN_VALUE); // trasmit mode
      Serial.println("Send Turbidity Data!");
      RS485Serial.print("Ready to Send Data"); // Send message
    }
    void loop() {
     // put your main code here, to run repeatedly:
       Turbidity_Sensor_Voltage = 0;
        for(int i=0; i<samples; i++)
       {
          Turbidity_Sensor_Voltage                                                    +=
((float)analogRead(Turbidity_Sensor_Pin)/1023)*5;
       }
       Turbidity_Sensor_Voltage = Turbidity_Sensor_Voltage/samples;
          Turbidity_Sensor_Voltage = round_to_dp(Turbidity_Sensor_Voltage,2);
       if(Turbidity_Sensor_Voltage < 2.5){
        ntu = 3000;
       }else{
        ntu = -1120.4*square(VHasil)+5742.3*VHasil-4353.8:;
       }
       //lcd.setCursor(0,0);
       //lcd.print("Temperature:");
       //lcd.print(temperature,1);
       //lcd.print("^C pH:");
       lcd.print("Voltage: ");
       lcd.setCursor(0,3);
       lcd.print(Turbidity_Sensor_Voltage,2);
       lcd.print(" V");
       lcd.setCursor(0,2);
       lcd.print("Turbidity: ");
       lcd.print(ntu);
       lcd.print(" NTU");
       delay(100);

       //Serial.print
       Serial.print("Voltage: ");
       Serial.print(Turbidity_Sensor_Voltage);
       Serial.println(" V");
       Serial.print("Turbidity: ");
       Serial.print(ntu);
```

```
  Serial.println(" NTU");
  delay(100);

  //RS485 Send Value
  RS485Serial.print("Voltage: ");
  RS485Serial.print(Turbidity_Sensor_Voltage);
  RS485Serial.println(" V");
  RS485Serial.print("Turbidity: ");
  RS485Serial.print(ntu);
  RS485Serial.println(" NTU");
  delay(500);
  }
 float round_to_dp( float in_value, int decimal_place )
{
 float multiplier = powf( 10.0f, decimal_place );
 in_value = roundf( in_value * multiplier ) / multiplier;
 return in_value;
}
```

Program Komunikasi TCP/IP

```
#include <SPI.h>
#include <Ethernet.h>
int pinTurb = A0;
float V;
float kekeruhan;
float VRata2;
float VHasil;

#include <LiquidCrystal_I2C.h>
//LiquidCrystal_I2C mylcd(0x20, 20, 4);
LiquidCrystal_I2C mylcd(0x27, 20, 4);

#define Turbidity_Sensor_Pin_A0
float Turbidity_Sensor_Voltage;
int samples = 600;
float ntu; //Nephelometric Turbidity Units

//untuk kalibrasi
//float PH4 = 3.031;
//float PH7 = 2.513;
```

```
// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};
IPAddress ip(192, 168, 137, 177);

// Initialize the Ethernet server library
// with the IP address and port you want to use
// (port 80 is default for HTTP):
EthernetServer server(80);

void setup() {
   mylcd.begin();
 mylcd.backlight();
 mylcd.setCursor(0, 0);
 delay(100);
 mylcd.clear();
 delay(500);
  // You can use Ethernet.init(pin) to configure the CS pin
  //Ethernet.init(10);  // Most Arduino shields
  //Ethernet.init(5);   // MKR ETH Shield
  //Ethernet.init(0);   // Teensy 2.0
  //Ethernet.init(20);  // Teensy++ 2.0
  //Ethernet.init(15);  // ESP8266 with Adafruit FeatherWing Ethernet
  //Ethernet.init(33);  // ESP32 with Adafruit FeatherWing Ethernet

  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }
  Serial.println("Ethernet WebServer Example");

  // start the Ethernet connection and the server:
  Ethernet.begin(mac, ip);

  // Check for Ethernet hardware present
  if (Ethernet.hardwareStatus() == EthernetNoHardware) {
    Serial.println("Ethernet shield was not found.  Sorry, can't run without
hardware. :(");
    while (true) {
```

```
    delay(100); // do nothing, no point running without Ethernet hardware
   }
  }
  if (Ethernet.linkStatus() == LinkOFF) {
   Serial.println("Ethernet cable is not connected.");
  }

  // start the server
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}


void loop() {
  // listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
   Serial.println("new client");
   // an HTTP request ends with a blank line
   bool currentLineIsBlank = true;
   while (client.connected()) {
    if (client.available()) {
     char c = client.read();
     Serial.write(c);
     // if you've gotten to the end of the line (received a newline
     // character) and the line is blank, the HTTP request has ended,
     // so you can send a reply
     if (c == '\n' && currentLineIsBlank) {
      // send a standard HTTP response header
      client.println("HTTP/1.1 200 OK");
      client.println("Content-Type: text/html");
      client.println("Connection: close");  // the connection will be closed after
completion of the response
      client.println("Refresh: 3");  // refresh the page automatically every 3  sec
      client.println();
      client.println("<!DOCTYPE HTML>");
      client.println("<html>");
      client.println("<header>");
      client.println("<meta  name='apple-mobile-web-app-capable'  content='yes'
/>");
```

```
        client.println("<meta name='apple-mobile-web-app-status-bar-style'
content='black-translucent' >");
        client.println("<link rel='stylesheet' type='text/css'
href='https://randomnerdtutorials.com/ethernetcss.css'/>");
        client.println("<title>Arduino Ethernet Web Server</title></head>");
        client.println("<body style=\"background-color:#87CEFA\">\n");
        client.println("<h1>Arduino Web Ethernet Shield Web Server</h1>");
        client.println("<h2>Politeknik Negeri Batam </h2>");
        client.println("<h3>OIL IN WATER ANALYZER</a> </h3")
        // output the value of each analog input pin


 ;V = 0;
 for(int i=0; i<800; i++)
 {
   V += ((float)analogRead(pinTurb)/1023)*5;
 }

 VRata2 = V/800;
 VHasil = roundf(VRata2*10.0f)/10.0f;

 if(VHasil < 2.5)
 {
   kekeruhan = 3000;
 }
 else
 {
   kekeruhan = -1120.4*square(VHasil)+5742.3*VHasil-4353.8;
 }

 {


        client.println("<h2>Turbidity</h2>");
        client.println("<h4>TEGANGAN</h4>");
        client.println(VHasil);
        client.println("<h4>KEKERUHAN</h4>");
        client.println(kekeruhan);

         mylcd.setCursor(0,0);
      //lcd.print("Temperature:");
      //lcd.print(temperature,1);
```

```
      //lcd.print("^C pH:");
      mylcd.print("TEGANGAN :");
      mylcd.print(VHasil);
      mylcd.setCursor(0,1);
      mylcd.print("KEKERUHAN:");
      mylcd.print(kekeruhan);
      delay(500);


       }
       client.println("</html>");
       break;
      }
      if (c == '\n') {
       // you're starting a new line
       currentLineIsBlank = true;
      } else if (c != '\r') {
       // you've gotten a character on the current line
       currentLineIsBlank = false;
      }
     }
    }
    // give the web browser time to receive the data
    delay(100);
    // close the connection:
    client.stop();
    Serial.println("client disconnected");
  }
```

Program Komunikasi Canbus

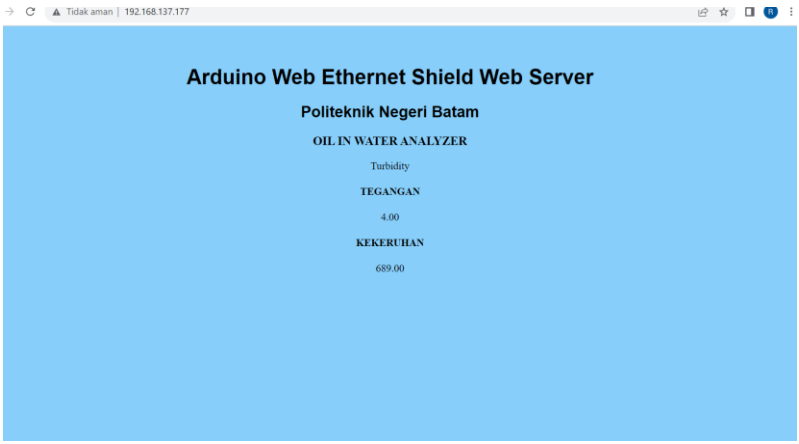
```
#include <mcp_can.h>
#include <SPI.h>

#define Turbidity_Sensor_Pin A5
float Turbidity_Sensor_Voltage;
int samples = 600;
float ntu;

const int SPI_CS_PIN = 10;
```

```
MCP_CAN can(SPI_CS_PIN);

void setup() {
  Serial.begin(19200);
  pinMode(Turbidity_Sensor_Pin, INPUT);

  if (can.begin(MCP_ANY, CAN_500KBPS, MCP_16MHZ) == CAN_OK) {
    Serial.println("MCP2515 Initialized Successfully!");
  } else {
    Serial.println("Error Initializing MCP2515!");
    while (1);
  }

  while (!Serial);

  Serial.println("Send Turbidity Data!");
}

void loop() {
  Turbidity_Sensor_Voltage = 0;

  for (int i = 0; i < samples; i++) {
    Turbidity_Sensor_Voltage += ((float)analogRead(Turbidity_Sensor_Pin) / 1023)
* 5;
  }

  Turbidity_Sensor_Voltage = Turbidity_Sensor_Voltage / samples;
  Turbidity_Sensor_Voltage = round_to_dp(Turbidity_Sensor_Voltage, 3);

  if (Turbidity_Sensor_Voltage < 2.5) {
    ntu = 3000;
  } else {
    ntu = -1120.4 * square(Turbidity_Sensor_Voltage) + 5742.3 *
Turbidity_Sensor_Voltage - 4353.8;
  }

  Serial.print("Voltage: ");
  Serial.print(Turbidity_Sensor_Voltage, 3);
  Serial.println(" V");
  Serial.print("Turbidity: ");
  Serial.print(ntu);
```

```
  Serial.println(" NTU");

  char message[50];
  sprintf(message, "Voltage: %.3f V, Turbidity: %.2f NTU",
Turbidity_Sensor_Voltage, ntu);
  can.sendMsgBuf(0x100, 0, strlen(message) + 1, (byte*)message);

  delay(1000);
}

float round_to_dp(float in_value, int decimal_place) {
  float multiplier = powf(10.0f, decimal_place);
  in_value = roundf(in_value * multiplier) / multiplier;
  return in_value;
}
```

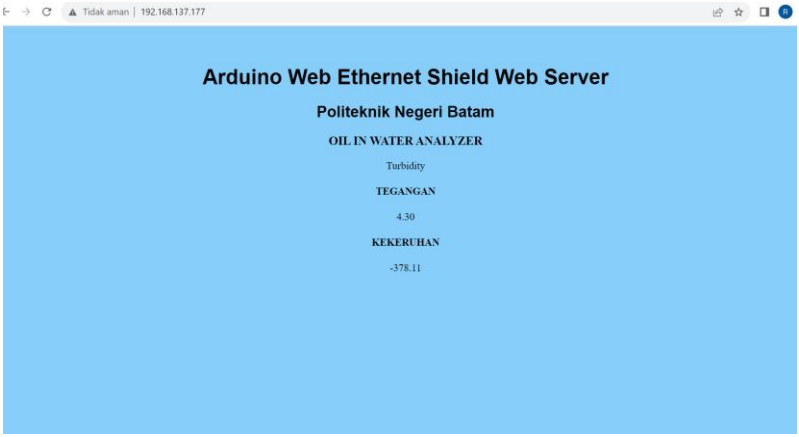
Tampilan Data di Web Server

Sampel 1

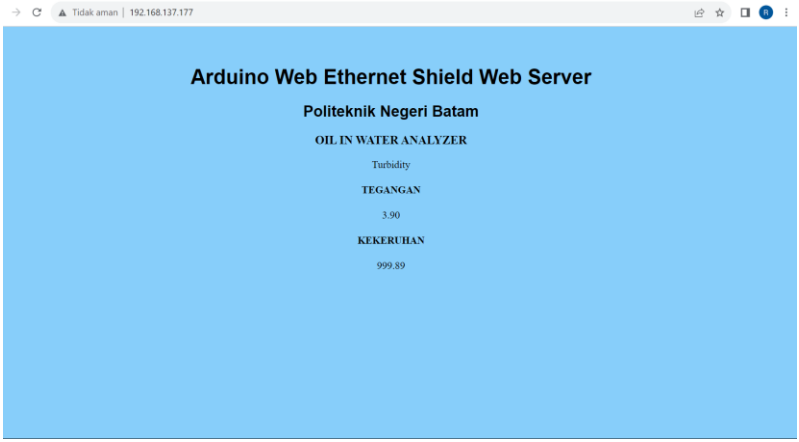


Sampel 2

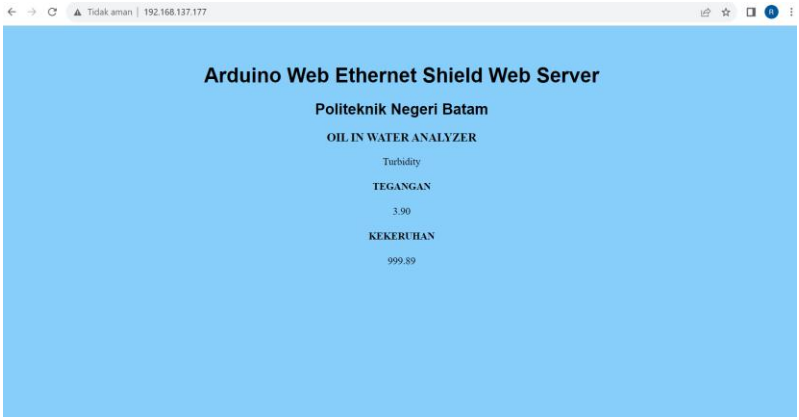
Sampel 3



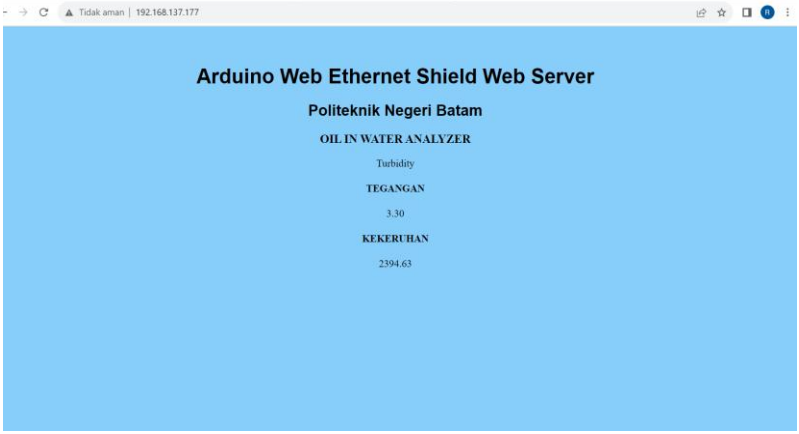
Sampel 4



Sampel 5



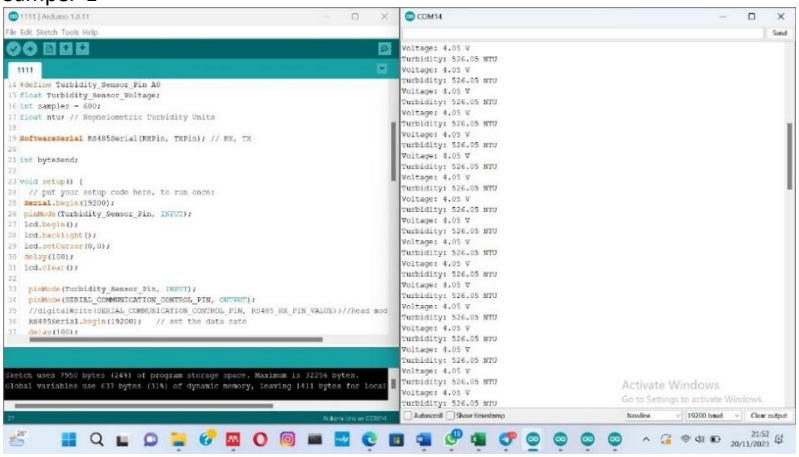
Sampel 6

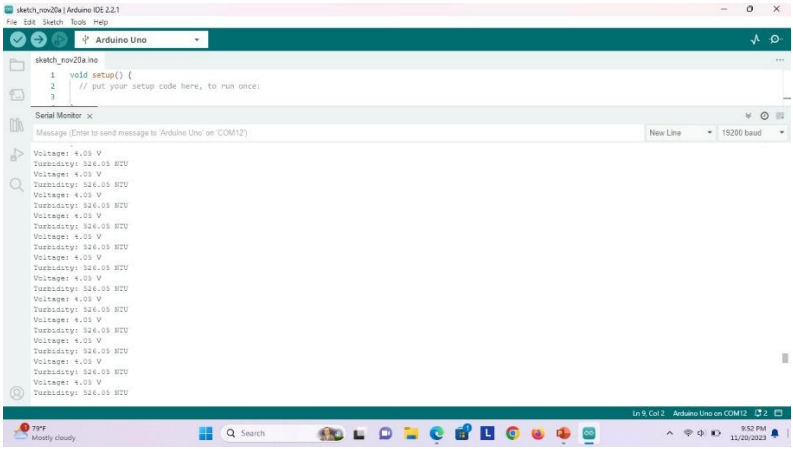


Sampel 7

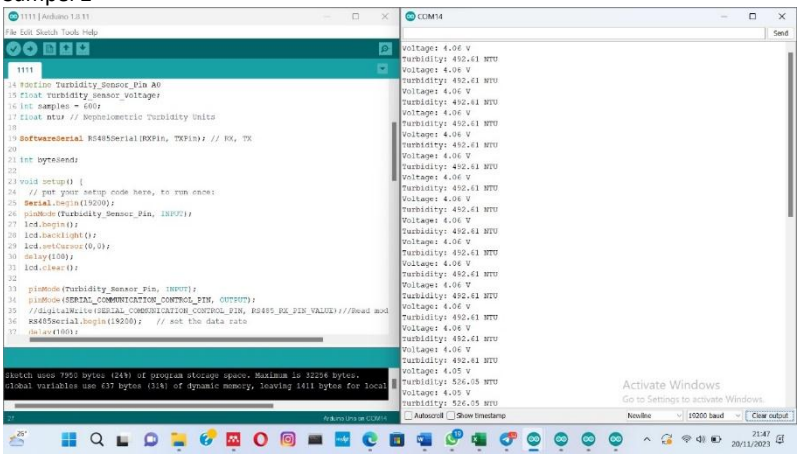
Data Komunikasi RS 485

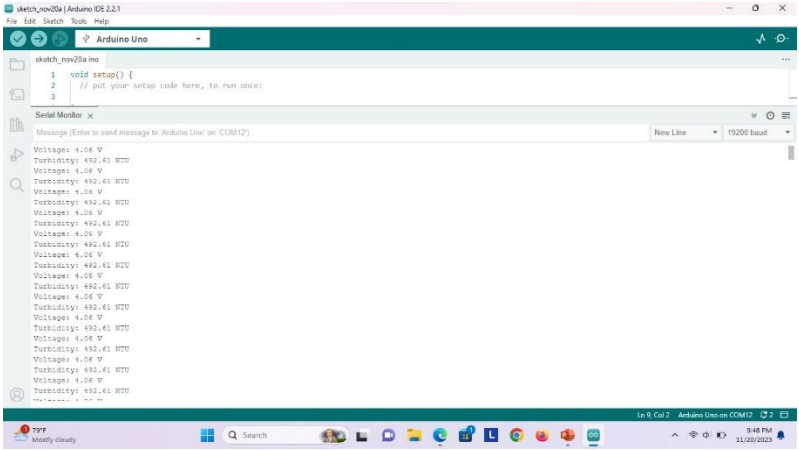
Sampel 1



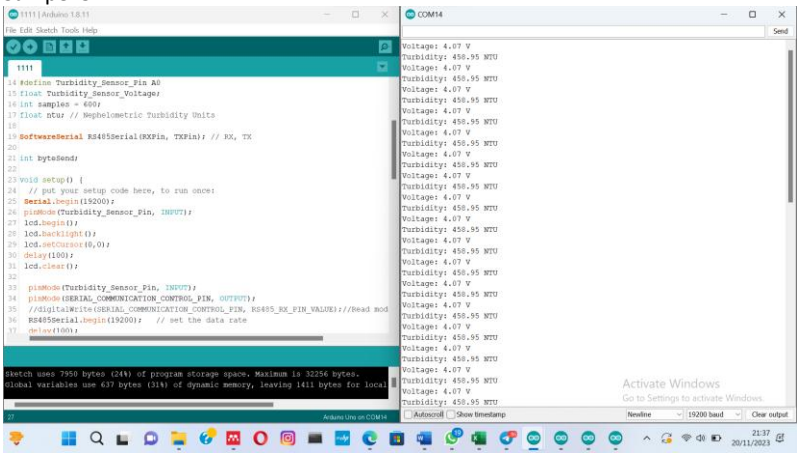

sketch_nov20a | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Arduino Uno
sketch_nov20a.ino
1 void setup() {
2 // put your setup code here, to run once:
3
Serial Monitor
Message (Enter to send message to 'Arduino Uno' on 'COM12')
New Line
19200 baud
Voltage: 4.05 V
Turbidity: 526.05 NTU
Ln 9, Col 2 Arduino Uno on COM12
79°F
Mostly cloudy
Search
9:52 PM
11/20/2023

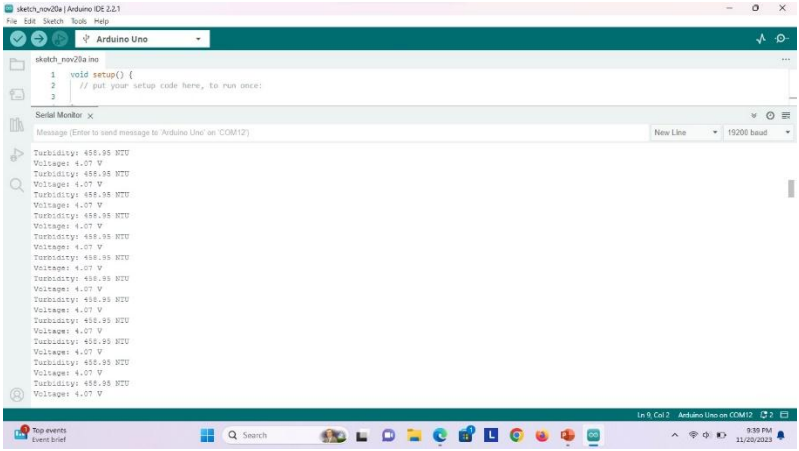
Sampel 2


1111 | Arduino 1.8.11
File Edit Sketch Tools Help
1111
#define Turbidity_Sensor_Pin A0
float turbidity_sensor_voltage;
int samples = 600;
float ntu; // Nephelometric Turbidity Units
SoftwareSerial RS485Serial(RXPin, TXPin); // RX, TX
int byteSend;
void setup() {
// put your setup code here, to run once:
Serial.begin(19200);
pinMode(Turbidity_Sensor_Pin, INPUT);
lcd.begin();
lcd.backlight();
lcd.setCursor(0,0);
delay(100);
lcd.clear();
pinMode(Turbidity_Sensor_Pin, INPUT);
pinMode(SERIAL_COMMUNICATION_CONTROL_PIN, OUTPUT);
//digitalWrite(SERIAL_COMMUNICATION_CONTROL_PIN, RS485_RX_PIN_VALUE);//Read mod
RS485Serial.begin(19200); // set the data rate
delay(100);
Sketch uses 7950 bytes (24%) of program storage space. Maximum is 32256 bytes.
Global variables use 457 bytes (31%) of dynamic memory, leaving 1411 bytes for local
Arduino Uno on COM14
COM14
Send
Voltage: 4.06 V
Turbidity: 492.61 NTU
Autoscroll
Show timestamp
Newline
19200 baud
Clear output
Activate Windows
Go to Settings to activate Windows.
21:47
20/11/2023

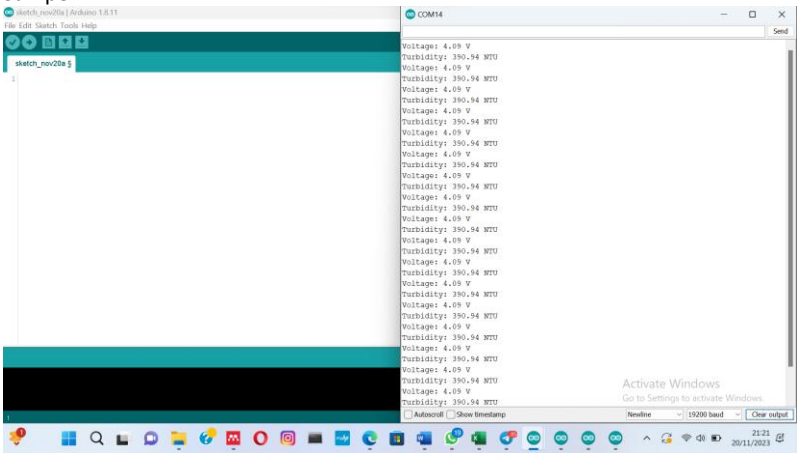

sketch_nov20a | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Arduino Uno
sketch_nov20a.ino
1 void setup() {
2 // put your setup code here, to run once:
3
Serial Monitor
Message (Enter to send message to 'Arduino Uno' on 'COM12')
New Line
19200 baud
Voltage: 4.06 V
Turbidity: 492.61 NTU
Ln 9, Col 2 Arduino Uno on COM12
79°F
Mostly cloudy
Search
9:48 PM
11/20/2023

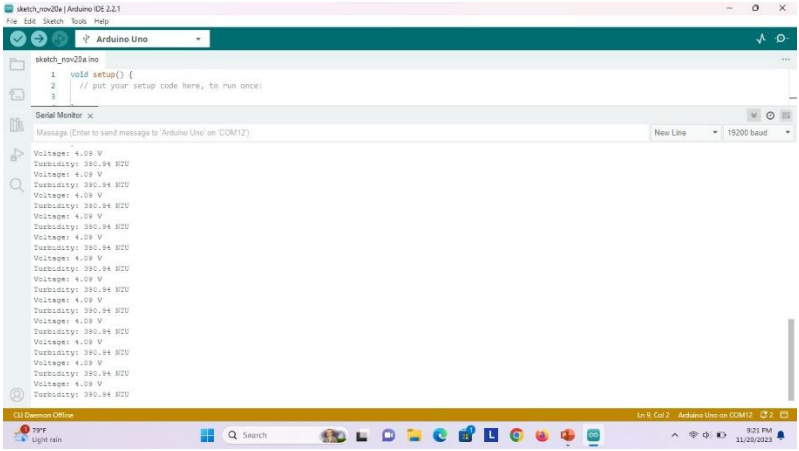
Sampel 3


1111 | Arduino 1.8.11
File Edit Sketch Tools Help
1111
#define Turbidity_Sensor_Pin A0
float Turbidity_Sensor_Voltage;
int samples = 600;
float ntu; // Nephelometric Turbidity Units
SoftwareSerial RS485Serial(RXPin, TXPin); // RX, TX
int bytesSend;
void setup() {
// put your setup code here, to run once:
Serial.begin(19200);
pinMode(Turbidity_Sensor_Pin, INPUT);
lcd.begin();
lcd.backlight();
lcd.setCursor(0,0);
delay(100);
lcd.clear();
pinMode(Turbidity_Sensor_Pin, INPUT);
pinMode(SERIAL_COMMUNICATION_CONTROL_PIN, OUTPUT);
//digitalWrite(SERIAL_COMMUNICATION_CONTROL_PIN, RS485_RX_PIN_VALUE);//Read mod
RS485Serial.begin(19200); // set the data rate
delay(100);
Sketch uses 7950 bytes (24%) of program storage space. Maximum is 32256 bytes.
Global variables use 637 bytes (31%) of dynamic memory, leaving 1411 bytes for local
Arduino Uno on COM14
COM14
Send
Voltage: 4.07 V
Turbidity: 458.95 NTU
Autoscroll
Show timestamp
Newline
19200 baud
Clear output
Activate Windows
Go to Settings to activate Windows.
21:37
20/11/2023

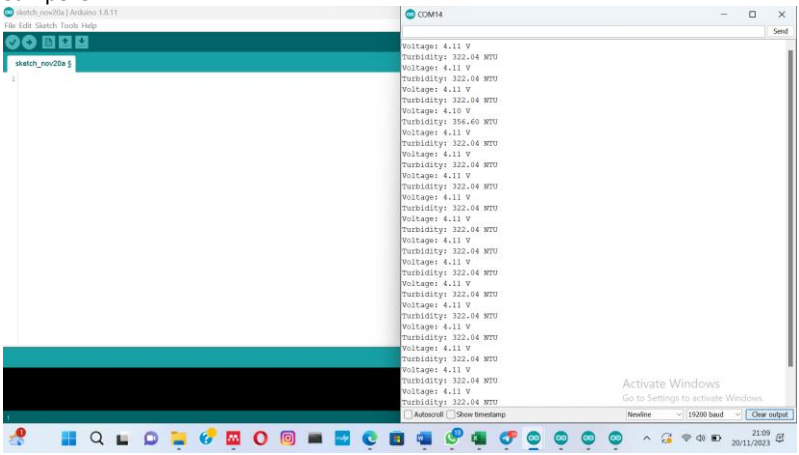

sketch_nov20a | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Arduino Uno
sketch_nov20a.ino
1 void setup() {
2 // put your setup code here, to run once:
3
Serial Monitor
Message (Enter to send message to 'Arduino Uno' on 'COM12')
New Line
19200 baud
Turbidity: 458.98 NTU
Voltage: 4.07 V
Ln 9, Col 2 Arduino Uno on COM12
Top events
Event brief
Search
9:39 PM
11/20/2023

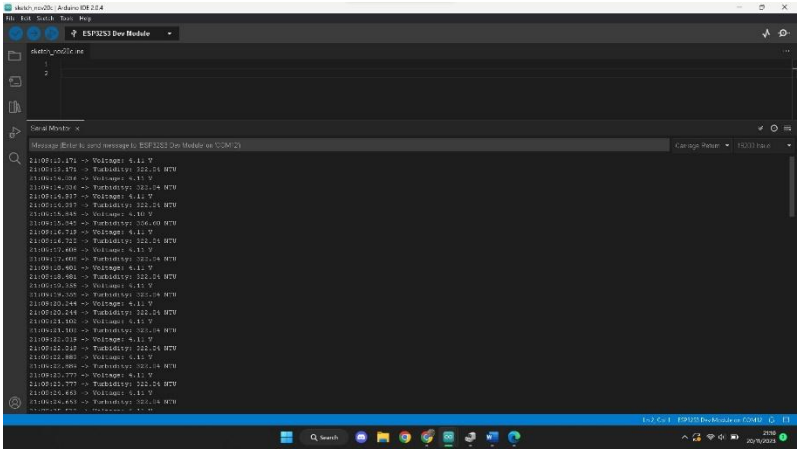
Sampel 4


sketch_nov20a | Arduino 1.8.11
File Edit Sketch Tools Help
sketch_nov20a §
COM14
Send
Voltage: 4.09 V
Turbidity: 390.94 NTU
Autoscroll
Show timestamp
Newline
19200 baud
Clear output
Activate Windows
Go to Settings to activate Windows.
21:21
20/11/2023

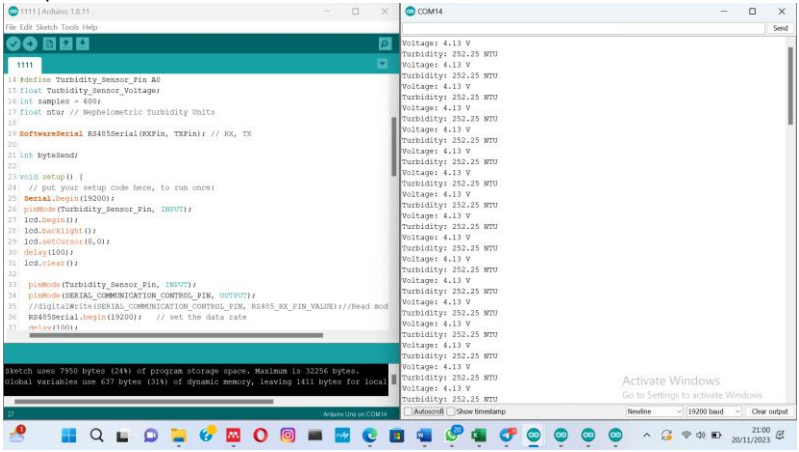

sketch_nov20a | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Arduino Uno
sketch_nov20a.ino
1 void setup() {
2 // put your setup code here, to run once:
3
Serial Monitor
Message (Enter to send message to 'Arduino Uno' on 'COM12')
New Line
19200 baud
Voltage: 4.09 V
Turbidity: 390.94 NTU
Voltage: 4.09 V
Turbidity: 390.94 NTU
Voltage: 4.09 V
Turbidity: 390.94 NTU
Voltage: 4.09 V
Turbidity: 390.94 NTU
Voltage: 4.09 V
Turbidity: 390.94 NTU
Voltage: 4.09 V
Turbidity: 390.94 NTU
Voltage: 4.09 V
Turbidity: 390.94 NTU
Voltage: 4.09 V
Turbidity: 390.94 NTU
Voltage: 4.09 V
Turbidity: 390.94 NTU
Voltage: 4.09 V
Turbidity: 390.94 NTU
Voltage: 4.09 V
Turbidity: 390.94 NTU
Voltage: 4.09 V
Turbidity: 390.94 NTU
Voltage: 4.09 V
Turbidity: 390.94 NTU
CLI Daemon Offline
Ln 9, Col 2 Arduino Uno on COM12
79°F
Light rain
Search
9:21 PM
11/20/2023

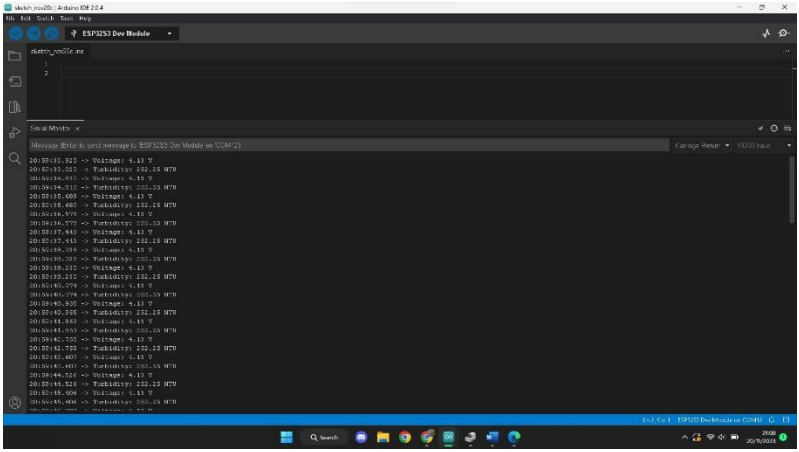
Sampel 5


sketch_nov20a | Arduino 1.8.11
File Edit Sketch Tools Help
sketch_nov20a §
COM14
Send
Voltage: 4.11 V
Turbidity: 322.04 NTU
Voltage: 4.11 V
Turbidity: 322.04 NTU
Voltage: 4.11 V
Turbidity: 322.04 NTU
Voltage: 4.10 V
Turbidity: 356.60 NTU
Voltage: 4.11 V
Turbidity: 322.04 NTU
Voltage: 4.11 V
Turbidity: 322.04 NTU
Voltage: 4.11 V
Turbidity: 322.04 NTU
Voltage: 4.11 V
Turbidity: 322.04 NTU
Voltage: 4.11 V
Turbidity: 322.04 NTU
Voltage: 4.11 V
Turbidity: 322.04 NTU
Voltage: 4.11 V
Turbidity: 322.04 NTU
Voltage: 4.11 V
Turbidity: 322.04 NTU
Voltage: 4.11 V
Turbidity: 322.04 NTU
Voltage: 4.11 V
Turbidity: 322.04 NTU
Voltage: 4.11 V
Turbidity: 322.04 NTU
Voltage: 4.11 V
Turbidity: 322.04 NTU
Voltage: 4.11 V
Turbidity: 322.04 NTU
Activate Windows
Go to Settings to activate Windows.
Autoscroll
Show timestamp
Newline
19200 baud
Clear output
21:09
20/11/2023

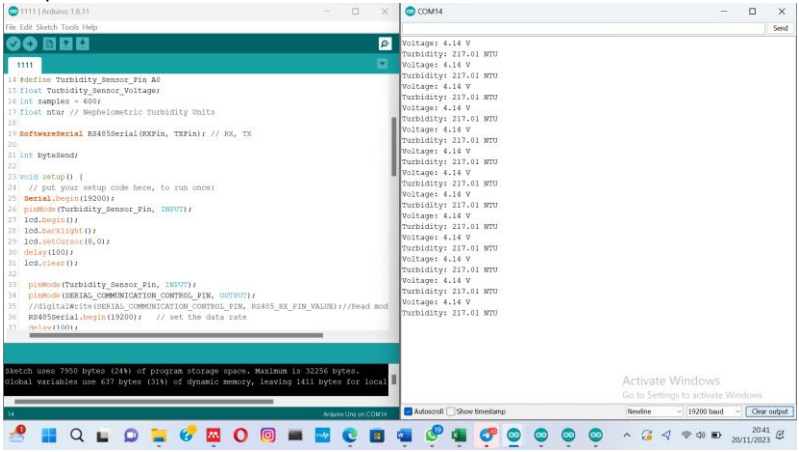


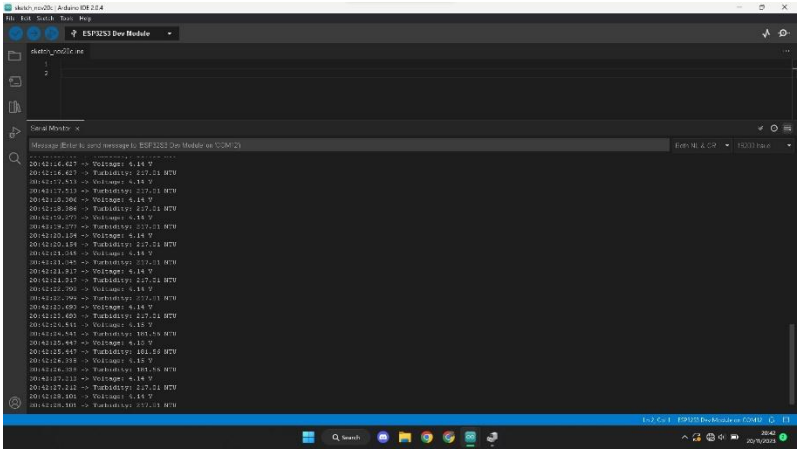
Sampel 6


1111 | Arduino 1.8.11
File Edit Sketch Tools Help
1111
#define Turbidity_Sensor_Pin A0
float Turbidity_Sensor_Voltage;
int samples = 600;
float ntu; // Nephelometric Turbidity Units
SoftwareSerial RS485Serial(RXPin, TXPin); // RX, TX
int bytesSend;
void setup() {
// put your setup code here, to run once:
Serial.begin(19200);
pinMode(Turbidity_Sensor_Pin, INPUT);
lcd.begin();
lcd.backlight();
lcd.setCursor(0,0);
delay(100);
lcd.clear();
pinMode(Turbidity_Sensor_Pin, INPUT);
pinMode(SERIAL_COMMUNICATION_CONTROL_PIN, OUTPUT);
//digitalWrite(SERIAL_COMMUNICATION_CONTROL_PIN, RS485_RX_PIN_VALUE);//Read mod
RS485Serial.begin(19200); // set the data rate
delay(100);
Sketch uses 7950 bytes (24%) of program storage space. Maximum is 32256 bytes.
Global variables use 637 bytes (31%) of dynamic memory, leaving 1411 bytes for local
Arduino Uno on COM14
COM14
Send
Voltage: 4.13 V
Turbidity: 252.25 NTU
Autoscroll
Show timestamp
Newline
19200 baud
Clear output
Activate Windows
Go to Settings to activate Windows.



Sampel 7

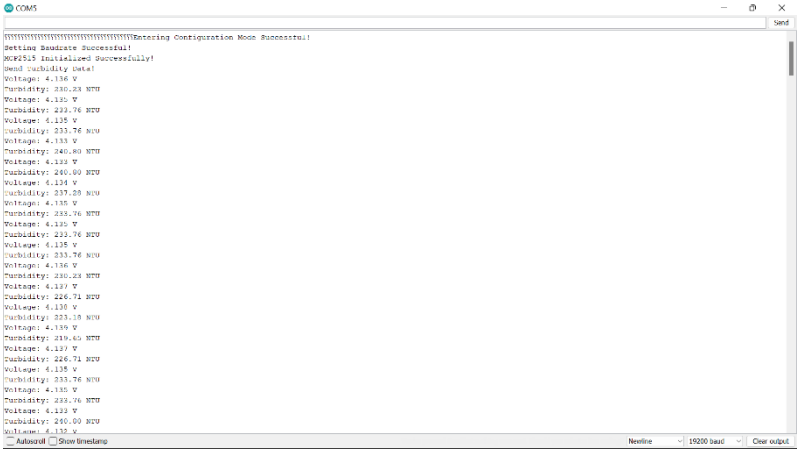

1111 | Arduino 1.8.11
File Edit Sketch Tools Help
1111
#define Turbidity_Sensor_Pin A0
float Turbidity_Sensor_Voltage;
int samples = 600;
float ntu; // Nephelometric Turbidity Units
SoftwareSerial RS485Serial(RXPin, TXPin); // RX, TX
int bytesSend;
void setup() {
// put your setup code here, to run once:
Serial.begin(19200);
pinMode(Turbidity_Sensor_Pin, INPUT);
lcd.begin();
lcd.backlight();
lcd.setCursor(0,0);
delay(100);
lcd.clear();
pinMode(Turbidity_Sensor_Pin, INPUT);
pinMode(SERIAL_COMMUNICATION_CONTROL_PIN, OUTPUT);
//digitalWrite(SERIAL_COMMUNICATION_CONTROL_PIN, RS485_RX_PIN_VALUE);//Read mod
RS485Serial.begin(19200); // set the data rate
delay(100);
Sketch uses 7950 bytes (24%) of program storage space. Maximum is 32256 bytes.
Global variables use 637 bytes (31%) of dynamic memory, leaving 1411 bytes for local
Arduino Uno on COM14
COM14
Send
Voltage: 4.16 V
Turbidity: 217.01 NTU
Autoscroll
Show timestamp
Newline
19200 baud
Clear output
Activate Windows
Go to Settings to activate Windows.



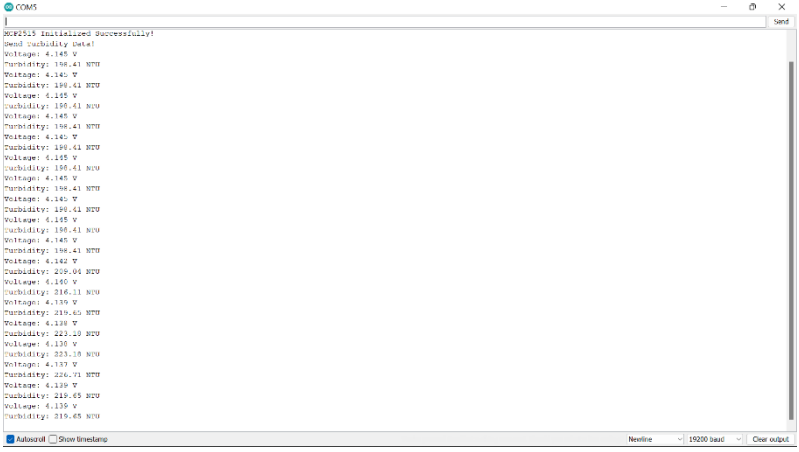
Data Komunikasi Canbus

Sampel 1

Sampel 2

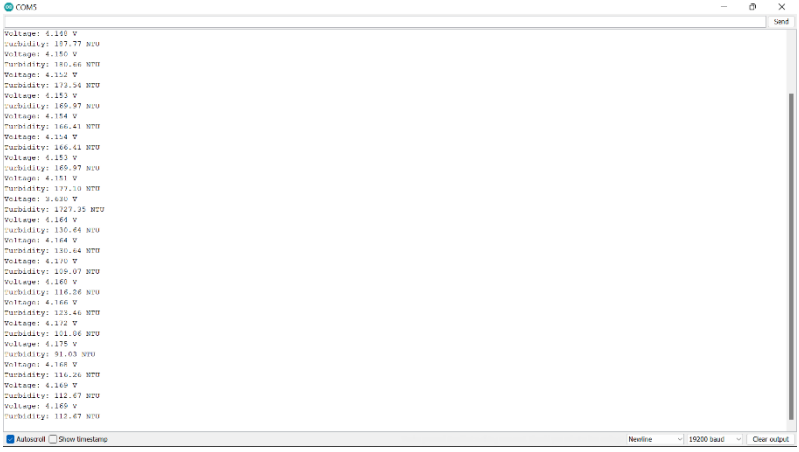


Sampel 3



Sampel 4

Sampel 5



Sampel 6

Sampel 7

Tabel Pengujian Sensor pada sampel asap rokok

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	50	4,183
2			4,178
3			4,183
4			4,183
5			4,183
6			4,183
7			4,188
8			4,188
9			4,183
10			4,188
11			4,183
12			4,183
13			4,188
14			4,183
15			4,183
16			4,183
17			4,188
18			4,188
19			4,183
20			4,183
Rata-rata Nilai Tegangan			4,184

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	50	4,163
2			4,163
3			4,163
4			4,163
5			4,163
6			4,163
7			4,158
8			4,158
9			4,158
10			4,158
11			4,153
12			4,148
13			4,148
14			4,148
15			4,148
16			4,153
17			4,153
18			4,158
19			4,158
20			4,158
Rata-rata Nilai Tegangan			4,157

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	50	4,158
2			4,158
3			4,153
4			4,153
5			4,158
6			4,158
7			4,153
8			4,153
9			4,153
10			4,158
11			4,158
12			4,153
13			4,158
14			4,153
15			4,153
16			4,158
17			4,158
18			4,163
19			4,163
20			4,168
Rata-rata Nilai Tegangan			4,157

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	50	4,182
2			4,178
3			4,184
4			4,157
5			4,157
Rata-rata Nilai Tegangan			4,171

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	45	4,183
2			4,183
3			4,188
4			4,198
5			4,203
6			4,198
7			4,198
8			4,203
9			4,203
10			4,198
11			4,203
12			4,198
13			4,198
14			4,193
15			4,193
16			4,193
17			4,193
18			4,193
19			4,188
20			4,188
Rata-rata Nilai Tegangan			4,195

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	45	4,198
2			4,198
3			4,198
4			4,193
5			4,193
6			4,188
7			4,188
8			4,193
9			4,183
10			4,183
11			4,193
12			4,178
13			4,183
14			4,173
15			4,178
16			4,188
17			4,188
18			4,188
19			4,183
20			4,188
Rata-rata Nilai Tegangan			4,188

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	45	4,193
2			4,188
3			4,188
4			4,188
5			4,188
6			4,188
7			4,183
8			4,183
9			4,183
10			4,188
11			4,178
12			4,183
13			4,188
14			4,188
15			4,188
16			4,188
17			4,188
18			4,188
19			4,188
20			4,183
Rata-rata Nilai Tegangan			4,187

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	45	4,198
2			4,193
3			4,193
4			4,188
5			4,183
6			4,183
7			4,183
8			4,188
9			4,188
10			4,188
11			4,188
12			4,188
13			4,183
14			4,193
15			4,198
16			4,198
17			4,198
18			4,198
19			4,188
20			4,188
Rata-rata Nilai Tegangan			4,190

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	45	4,193
2			4,188
3			4,188
4			4,188
5			4,188
6			4,188
7			4,183
8			4,183
9			4,183
10			4,188
11			4,178
12			4,183
13			4,188
14			4,188
15			4,188
16			4,188
17			4,188
18			4,188
19			4,188
20			4,183
Rata-rata Nilai Tegangan			4,187

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	45	4,198
2			4,193
3			4,193
4			4,188
5			4,183
6			4,183
7			4,183
8			4,188
9			4,188
10			4,188
11			4,188
12			4,188
13			4,183
14			4,193
15			4,198
16			4,198
17			4,198
18			4,198
19			4,188
20			4,188
Rata-rata Nilai Tegangan			4,190

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	45	4,193
2			4,183
3			4,183
4			4,178
5			4,178
6			4,173
7			4,173
8			4,173
9			4,178
10			4,173
11			4,173
12			4,168
13			4,173
14			4,173
15			4,173
16			4,168
17			4,173
18			4,173
19			4,178
20			4,173
Rata-rata Nilai Tegangan			4,176

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	45	4,195
2			4,188
3			4,187
4			4,190
5			4,176
Rata-rata Nilai Tegangan			4,187

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	40	4,193
2			4,198
3			4,198
4			4,203
5			4,198
6			4,203
7			4,198
8			4,183
9			4,193
10			4,188
11			4,183
12			4,193
13			4,198
14			4,208
15			4,208
16			4,213
17			4,208
18			4,208
19			4,203
20			4,198
Rata-rata Nilai Tegangan			4,199

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	40	4,203
2			4,203
3			4,203
4			4,198
5			4,198
6			4,198
7			4,198
8			4,198
9			4,198
10			4,198
11			4,198
12			4,198
13			4,203
14			4,203
15			4,203
16			4,203
17			4,203
18			4,208
19			4,203
20			4,203
Rata-rata Nilai Tegangan			4,201

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	40	4,198
2			4,203
3			4,203
4			4,208
5			4,203
6			4,208
7			4,203
8			4,203
9			4,203
10			4,203
11			4,208
12			4,203
13			4,198
14			4,203
15			4,198
16			4,198
17			4,198
18			4,198
19			4,203
20			4,203
Rata-rata Nilai Tegangan			4,202

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	40	4,208
2			4,208
3			4,203
4			4,203
5			4,203
6			4,198
7			4,198
8			4,198
9			4,198
10			4,198
11			4,198
12			4,203
13			4,198
14			4,203
15			4,203
16			4,203
17			4,198
18			4,198
19			4,198
20			4,198
Rata-rata Nilai Tegangan			4,201

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	40	4,208
2			4,208
3			4,208
4			4,208
5			4,208
6			4,213
7			4,208
8			4,203
9			4,203
10			4,208
11			4,208
12			4,208
13			4,203
14			4,208
15			4,208
16			4,203
17			4,208
18			4,208
19			4,208
20			4,203
Rata-rata Nilai Tegangan			4,207

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	40	4,199
2			4,201
3			4,202
4			4,201
5			4,207
Rata-rata Nilai Tegangan			4,202

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	35	4,227
2			4,227
3			4,223
4			4,223
5			4,223
6			4,227
7			4,223
8			4,223
9			4,227
10			4,227
11			4,227
12			4,232
13			4,232
14			4,232
15			4,232
16			4,227
17			4,227
18			4,227
19			4,227
20			4,227
Rata-rata Nilai Tegangan			4,227

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	35	4,218
2			4,223
3			4,223
4			4,223
5			4,223
6			4,223
7			4,223
8			4,218
9			4,213
10			4,213
11			4,218
12			4,218
13			4,218
14			4,218
15			4,223
16			4,223
17			4,223
18			4,218
19			4,218
20			4,218
Rata-rata Nilai Tegangan			4,220

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	35	4,213
2			4,218
3			4,218
4			4,223
5			4,218
6			4,218
7			4,218
8			4,218
9			4,218
10			4,213
11			4,218
12			4,223
13			4,223
14			4,227
15			4,227
16			4,218
17			4,218
18			4,218
19			4,218
20			4,218
Rata-rata Nilai Tegangan			4,219

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	35	4,218
2			4,213
3			4,218
4			4,223
5			4,218
6			4,223
7			4,223
8			4,223
9			4,218
10			4,213
11			4,213
12			4,218
13			4,223
14			4,218
15			4,213
16			4,218
17			4,218
18			4,213
19			4,218
20			4,218
Rata-rata Nilai Tegangan			4,218

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	35	4,208
2			4,203
3			4,208
4			4,213
5			4,213
6			4,213
7			4,208
8			4,213
9			4,218
10			4,223
11			4,227
12			4,218
13			4,218
14			4,208
15			4,208
16			4,213
17			4,223
18			4,223
19			4,218
20			4,218
Rata-rata Nilai Tegangan			4,215

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	35	4,227
2			4,220
3			4,219
4			4,218
5			4,215
Rata-rata Nilai Tegangan			4,220

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	30	4,237
2			4,237
3			4,232
4			4,237
5			4,237
6			4,227
7			4,227
8			4,237
9			4,232
10			4,227
11			4,232
12			4,237
13			4,218
14			4,208
15			4,223
16			4,232
17			4,223
18			4,223
19			4,232
20			4,227
Rata-rata Nilai Tegangan			4,229

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	30	4,232
2			4,227
3			4,227
4			4,223
5			4,223
6			4,223
7			4,227
8			4,227
9			4,227
10			4,227
11			4,232
12			4,232
13			4,237
14			4,232
15			4,237
16			4,237
17			4,232
18			4,232
19			4,232
20			4,232
Rata-rata Nilai Tegangan			4,230

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	30	4,232
2			4,227
3			4,232
4			4,227
5			4,227
6			4,232
7			4,232
8			4,232
9			4,232
10			4,232
11			4,237
12			4,232
13			4,232
14			4,237
15			4,232
16			4,232
17			4,237
18			4,237
19			4,232
20			4,227
Rata-rata Nilai Tegangan			4,232

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	30	4,232
2			4,232
3			4,237
4			4,232
5			4,232
6			4,232
7			4,237
8			4,237
9			4,242
10			4,242
11			4,237
12			4,242
13			4,237
14			4,232
15			4,232
16			4,227
17			4,223
18			4,223
19			4,218
20			4,223
Rata-rata Nilai Tegangan			4,232

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	30	4,223
2			4,223
3			4,227
4			4,227
5			4,232
6			4,227
7			4,232
8			4,232
9			4,232
10			4,227
11			4,232
12			4,227
13			4,227
14			4,232
15			4,227
16			4,223
17			4,223
18			4,223
19			4,223
20			4,227
Rata-rata Nilai Tegangan			4,227

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	30	4,229
2			4,230
3			4,232
4			4,232
5			4,227
Rata-rata Nilai Tegangan			4,230

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	25	4,257
2			4,262
3			4,257
4			4,257
5			4,257
6			4,257
7			4,262
8			4,257
9			4,262
10			4,262
11			4,262
12			4,262
13			4,262
14			4,267
15			4,262
16			4,267
17			4,262
18			4,262
19			4,262
20			4,267
Rata-rata Nilai Tegangan			4,261

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	25	4,267
2			4,267
3			4,267
4			4,272
5			4,267
6			4,267
7			4,272
8			4,272
9			4,272
10			4,277
11			4,272
12			4,272
13			4,272
14			4,272
15			4,272
16			4,272
17			4,272
18			4,272
19			4,272
20			4,267
Rata-rata Nilai Tegangan			4,271

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	25	4,262
2			4,267
3			4,267
4			4,267
5			4,267
6			4,267
7			4,262
8			4,262
9			4,267
10			4,267
11			4,267
12			4,262
13			4,267
14			4,262
15			4,262
16			4,262
17			4,262
18			4,262
19			4,267
20			4,272
Rata-rata Nilai Tegangan			4,265

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	25	4,262
2			4,267
3			4,267
4			4,267
5			4,267
6			4,267
7			4,262
8			4,262
9			4,267
10			4,267
11			4,267
12			4,262
13			4,267
14			4,262
15			4,262
16			4,262
17			4,262
18			4,262
19			4,267
20			4,272
Rata-rata Nilai Tegangan			4,265

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	25	4,267
2			4,262
3			4,262
4			4,262
5			4,262
6			4,262
7			4,267
8			4,262
9			4,262
10			4,262
11			4,262
12			4,267
13			4,267
14			4,267
15			4,262
16			4,267
17			4,262
18			4,262
19			4,262
20			4,267
Rata-rata Nilai Tegangan			4,264

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	25	4,262
2			4,262
3			4,262
4			4,267
5			4,267
6			4,267
7			4,267
8			4,272
9			4,272
10			4,272
11			4,272
12			4,272
13			4,267
14			4,267
15			4,262
16			4,262
17			4,267
18			4,272
19			4,272
20			4,267
Rata-rata Nilai Tegangan			4,268

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	25	4,261
2			4,271
3			4,265
4			4,264

5			4,268
Rata-rata Nilai Tegangan			4,266

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	20	4,292
2			4,292
3			4,287
4			4,287
5			4,282
6			4,287
7			4,287
8			4,287
9			4,287
10			4,287
11			4,282
12			4,282
13			4,282
14			4,287
15			4,282
16			4,282
17			4,287
18			4,292
19			4,287
20			4,287
Rata-rata Nilai Tegangan			4,286

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	20	4,287
2			4,282
3			4,282
4			4,282
5			4,277
6			4,282
7			4,282
8			4,282
9			4,287
10			4,282
11			4,287
12			4,287
13			4,282
14			4,287
15			4,282
16			4,282
17			4,282
18			4,282
19			4,287
20			4,287
Rata-rata Nilai Tegangan			4,284

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	20	4,287
2			4,287
3			4,292
4			4,292
5			4,292
6			4,292
7			4,292
8			4,292
9			4,292
10			4,287
11			4,292
12			4,287
13			4,292
14			4,292
15			4,287
16			4,287
17			4,287
18			4,292
19			4,292
20			4,292
Rata-rata Nilai Tegangan			4,290

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	20	4,287
2			4,287
3			4,292
4			4,292
5			4,292
6			4,292
7			4,292
8			4,292
9			4,292
10			4,287
11			4,292
12			4,287
13			4,292
14			4,292
15			4,287
16			4,287
17			4,287
18			4,292
19			4,292
20			4,292
Rata-rata Nilai Tegangan			4,290

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	20	4,282
2			4,282
3			4,282
4			4,282
5			4,282
6			4,287
7			4,282
8			4,282
9			4,277
10			4,277
11			4,277
12			4,277
13			4,277
14			4,282
15			4,287
16			4,282
17			4,287
18			4,282
19			4,282
20			4,282
Rata-rata Nilai Tegangan			4,282

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	20	4,282
2			4,287
3			4,282
4			4,282
5			4,282
6			4,282
7			4,282
8			4,282
9			4,282
10			4,282
11			4,287
12			4,287
13			4,287
14			4,292
15			4,292
16			4,292
17			4,287
18			4,292
19			4,287
20			4,287
Rata-rata Nilai Tegangan			4,286

No	Nama Sampel	Nilai Turbidity Meter	Nilai Tegangan
1	Asap Rokok	20	4,286
2			4,284
3			4,290
4			4,282
5			4,286
Rata-rata Nilai Tegangan			4,285

Tabel Akusisi Data

No	Turbidity Meter	Tegangan Sensor	Trubidity Sensor
1	50	3409,74	245,54
2		3409,74	245,54
3		3409,74	245,54
4		3399,82	248,11
5		3404,78	246,82
6		3414,69	244,25
7		3409,74	245,54
8		3419,65	242,97
9		3404,78	246,82
10		3414,69	244,25
11		3409,74	245,54
12		3454,34	233,98
13		3404,78	246,82
14		3409,74	245,54
15		3424,6	241,69
16		3414,69	244,25
17		3419,65	242,97
18		3424,6	241,69
19		3414,69	244,25
20		3414,69	244,25
Rata-Rata		3414,4445	244,318

No	Turbidity Meter	Tegangan Sensor	Trubidity Sensor
1	45	3612,93	192,89
2		3588,15	199,31
3		3603,02	195,46
4		3607,98	194,17
5		3603,02	195,46
6		3607,98	194,17
7		3607,98	194,17
8		3607,98	194,17
9		3607,98	194,17
10		3607,98	194,17
11		3612,93	192,89
12		3612,93	192,89
13		3617,89	191,61
14		3617,89	191,61
15		3612,93	192,89
16		3617,89	191,61
17		3617,89	191,61
18		3622,84	190,32
19		3617,89	191,61
20		3622,84	190,32
Rata-Rata		3611,446	193,275

No	Turbidity Meter	Tegangan Sensor	Trubidity Sensor
1	40	3682,32	174,91
2		3687,27	173,63
3		3687,27	173,63
4		3682,32	174,91
5		3682,32	174,91
6		3657,54	181,33
7		3677,36	176,2
8		3672,4	177,48
9		3677,36	176,2
10		3682,32	174,91
11		3667,45	178,76
12		3687,27	173,63
13		3677,36	176,2
14		3687,27	173,63
15		3687,27	173,63
16		3672,4	177,48
17		3682,32	174,91
18		3672,4	177,48
19		3692,23	172,34
20		3682,32	174,91
Rata-Rata		3679,8385	175,554

No	Turbidity Meter	Tegangan Sensor	Trubidity Sensor
1	35	3796,3	145,38
2		3796,3	145,38
3		3801,26	144,09
4		3801,26	144,09
5		3801,26	144,09
6		3806,22	142,81
7		3801,26	144,09
8		3801,26	144,09
9		3796,3	145,38
10		3796,3	145,38
11		3796,3	145,38
12		3796,3	145,38
13		3796,3	145,38
14		3796,3	145,38
15		3801,26	144,09
16		3796,3	145,38
17		3796,3	145,38
18		3796,3	145,38
19		3796,3	145,38
20		3796,3	145,38
Rata-Rata		3798,284	144,8645

FORMULIR LOOGBOOK BIMBINGAN DAN PENGAJUAN SIDANG TUGAS AKHIR

Nama : Maudy Arumdhyanita Ayu
NIM : 3232101080
Pembimbing I : Muhammad Jaka Wimbang Wicaksono, S.T., M.T
Pembimbing II : Asrizal Deri Futra, S.Si, M.Si
Judul : Oil in Water Analyzer Berbasis Komunikasi Menggunakan CAN Bus

No	Hari/Tgl	Rincian Kegiatan	TTD Pembimbing I & II	
1	Kamis, 21 September 2023	Revisi Bab 1		
2	Kamis, 28 September 2023	Revisi Bab 2		
3	Kamis, 05 Oktober 2023	Revisi Bab 3		
4	Kamis, 12 Oktober 2023	Revisi Bab 4		
5	Kamis, 19 Oktober 2023	Pengujian Akuisisi Data		
6	Kamis, 26 Oktober 2023	komunikasi can bus		
7	Kamis, 2 November 2023	Pengujian can bus		
8	Kamis, 9 November 2023	Revisi Paper		
9	Kamis, 16 November 2023	Pengujian tampilan LCD		
10	Kamis, 23 November 2023	Pengujian Nilai error can bus		

Berdasarkan hasil bimbingan yang telah dilaksanakan selama 6 bulan dan telah disetujui oleh dosen pembimbing, maka dengan ini saya mengajukan diri sebagai peserta Sidang Tugas Akhir.

Batam, 23 November 2023
Peserta

Maudy Arumdhyanita Ayu
NIM: 3232101080