# BAB III
# PELAKSANAAN KERJA PROFESI

## 3.1 Bidang Kerja

Dalam pelaksanaan kerja profesi di PT MARKTEL, praktikan ditempatkan di dalam tim Research and Development (R&D). Posisi praktikan dalam proyek ini adalah sebagai anggota tim *Back-end*, yang memiliki peran penting dalam mengembangkan program untuk sensor HCD6818 yang digunakan untuk pemantauan kualitas udara. Deskripsi Pekerjaan:

(1) Pengembangan Alat: Dalam pengembangan alat dalam proyek ini praktikan hanya mendapatkan gambaran dan ilmu mengenai rangkaian yang diperlukan dalam pengembangan. Walaupun praktikan tidak terjun langsung kedalam pembuatan, namun praktikan diberikan pemahaman tentang alur blok diagram sebagai dasar dari pembuatan alat. Dengan bantuan dari tim produksi yang ahli dalam bidang elektronika praktikan dikenali dengan berbagai komponen yang berkaitan.

(2) Pengembangan Program: Tugas utama praktikan dalam proyek ini adalah mengembangkan mengembangkan algoritma untuk menarik data sensor melalui mikrokontroller Beaglebone yang terintegrasi dengan sensor HCD6818. Program yang dirancang bertujuan untuk mengumpulkan data kualitas udara yang diperoleh dari sensor tersebut.

(3) Analisis Data: Setelah data kualitas udara terkumpul, praktikan melakukan analisis untuk menghasilkan informasi yang berguna dalam pemantauan kualitas udara. Sebelum data tersebut di masukkan kedalam *database* data ini perlu disesuaikan dengan melalui proses perhitungan sesuai standar baku mutu pemerintah PP Nomor 22 tahun 2021 tentang Penyelenggaraan Perlingungan dan Pengelolaan Lingkungan Hidup.

Bidang pekerjaan yang praktikan lakukan memiliki kaitan erat dengan beberapa bidang kerja lain dalam PT MARKTEL:

(1) Produksi: Tim praktikan bekerja sama dengan tim produksi yang bertanggung jawab atas pembuatan alat pemantau kualitas udara.

Praktikan memastikan bahwa program yang praktikan kembangkan dapat berfungsi dengan baik dengan produk yang dihasilkan oleh tim produksi.

(2) *Front-end*: Tim *Front-end* yang berkolaborasi dengan praktikan merupakan bagian dari tim R&D perusahaan. Kerja sama antara *Back-end* dan *Front-end* sangat penting untuk menciptakan solusi yang lengkap dan fungsional. Selain itu, tim *Front-end* akan menangkap data yang dikirim dari tim *Back-end* untuk ditampilkan kedalam *Dashboard Web*.

(3) Direktur Teknis: PT MARKTEL memiliki divisi Direktur Teknis yang memastikan proyek-proyek, seperti yang praktikan lakukan, berjalan sesuai jadwal dan anggaran. Praktikan berkoordinasi dengan divisi ini untuk mengikuti proses dan arahan untuk pengembangan yang efisien.

Dengan demikian, pekerjaan praktikan di bidang R&D tidak hanya berdampak pada pengembangan teknologi perusahaan tetapi juga terintegrasi dengan berbagai aspek bisnis di PT MARKTEL.

### 3.2 Pelaksanaan Kerja

Pekerjaan yang di lakukan oleh praktikan dalam mengembangkan program untuk sensor HCD6818 yang digunakan dalam pemantauan kualitas udara di PT. MARKTEL terdiri dari beberapa tahap. Tahap ini dibagi menjadi beberapa bagian seperti berikut:

**Tabel 3.1 Aktivitas Program Kerja Profesi**

| Aktivitas | Juni | | | | Juli | | | | Agustus | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | M1 | M2 | M3 | M4 | M1 | M2 | M3 | M4 | M1 | M2 | M3 | M4 |
| Analisis Kebutuhan Projek | | | | | | | | | | | | |
| Analisis Kebutuhan Program | | | | | | | | | | | | |
| Perancangan Koneksi Sensor | | | | | | | | | | | | |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Perancangan Program *Input* Data Sensor | | | | | | | | | | | | |
| Perancangan Program Konversi | | | | | | | | | | | | |
| Perancangan Program Koreksi | | | | | | | | | | | | |
| Perancangan Program Komparasi | | | | | | | | | | | | |
| Pengujian Proses *Input* dan *Output* Program | | | | | | | | | | | | |
| Presentasi Akhir | | | | | | | | | | | | |

Tabel 3.1 menjelaskan tahapan mingguan yang telah dilaksanakan oleh praktikan selama program kerja profesi. Pada tahapan awal praktikan menganalisis kebutuhan projek untuk dapat mengerti dan memahami maksud dan tujuan pada projek. Pada tahap analisis kebutuhan praktikan mencari informasi dalam pengembangan program. Kebutuhan pada program yang dirancang di ambil dari spesifikasi perangkat, cara penggunaan, dan hingga perancangan alat. Pada tahap perancangan koneksi dengan sensor perlu penyesuaian yang dilakukan praktikan, dikarenakan perangkat yang digunakan masuk kedalam skala industri dan juga perlunya *database* dan server sebagai bantuan dalam pengerjaan projek ini. *Input* yang perlu diberikan kepada sensor harus sesuai dengan spesifikasi dari sensor, sehingga diperlukan penyesuaian data yang dikirim. Setelahnya ada tiga tahap pengolahan data yang pertama koreksi yaitu data yang ditangkap dari hasil sensor. Koreksi merupakan tahapan data yang disesuaikan satuannya dengan peraturan baku mutu udara. Dan terakhir tahap komparasi yang mana pengkategorian data dengan peraturan Indeks Standar Pencemar Udara sehingga data bisa menjadi bahan informasi untuk ditampilkan. Tahap pengujian *input* dan *output* program dibutuhkan untuk pengecekan kesesuaian hitungan data dan memastikan data yang kredibel. Pada akhir program praktikan mempresentasikan

hasil program yang telah dikembangkan dan melakukan revisi akhir sebelum diluncurkan.

Adapun beberapa bahasan yang dilakukan praktikan selama melaksanakan program kerja profesi yang dilakukan di PT MARKTEL, sebagai berikut.

### 3.2.1 Pemahaman Proyek

Proses pengembangan proyek pemantau kualitas udara ambien dengan sensor HCD6818, praktikan melakukan beberapa riset mengenai hal-hal yang akan berkaitan dengan proses pelaksanaan. Hal ini dilakukan agar praktikan dapat menguasai tentang proyek yang dikerjakan, di antaranya:

(1) Latar belakang proyek

Kesadaran akan kualitas udara buruk akibat polusi udara di jalan yang meningkat mendorong perlunya pemantauan udara yang akurat dan efektif. Pemantauan udara ambien yang akurat dan real-time memberikan informasi penting tentang tingkat polutan seperti $PM2.5$ (Particulate Matter), $PM10$ (Particulate Matter), $CO$ (Karbon Monoksida), $SO2$ (Sulfur Dioksida), $NO2$ (Nitrogen Dioksida), $O3$ (Ozon), $T$ (Temperature), RH (Relative Humidity), dan beberapa informasi terkait dampaknya terhadap kesehatan dan lingkungan. Sistem sensor udara yang akurat dan terukur diharapkan dapat digunakan oleh pihak berwenang, peneliti, dan masyarakat untuk mengambil tindakan yang diperlukan guna meningkatkan kualitas udara dan meningkatkan kesadaran sosial.

(2) Tujuan dan manfaat pengembangan proyek

Membangun sistem pemantauan udara ambien yang handal dan efisien untuk sistem yang dapat mengumpulkan data tentang kualitas udara secara akurat. Menampilkan informasi kualitas udara secara realtime. Mengidentifikasi sumber polusi udara ambien dan mengukur tingkat paparan polutan di lingkungan.

Sistem ini dapat membantu mengidentifikasi masalah lingkungan seperti titik-titik polusi atau kebisingan yang tinggi di

jalan raya. Data lingkungan dianalisis untuk pengambilan keputusan yang lebih baik dalam pengelolaan lingkungan, termasuk perencanaan rute transportasi, pengaturan lalu lintas, dan kebijakan peningkatan kualitas udara. Sistem ini dapat membantu mengurangi emisi kendaraan, polusi udara, dankontribusi negatif lainnya terhadap lingkungan.

(3) Peraturan baku mutu udara

Baku mutu udara ambien merupakan ukuran batas atau kadar zat, energi, dan/atau komponen yang ada atau seharusnya ada dan/atau unsur pencemar yang ditenggang keberadaannya dalam udara ambien. Berdasarkan Peraturan Republik Indonesia (2021) Lampiran VII pada PP Nomor. 22 tahun 2021 ini, baku mutu sudah ditetapkan sebagai perhitungan batasan maksimal (dengan suhu 25℃ dan tekanan 1 Atm).

**Tabel 2.2 Batas Baku Mutu Udara**

| NO | Parameter | Waktu Pengukuran | Baku Mutu ($ug/m^3$) | Proses |
|---|---|---|---|---|
| 1 | Sulfur Dioksida ($SO_2$) | 1 jam | 150 | KM |
| | | 24 jam | 75 | K |
| | | 1 tahun | 45 | K |
| 2 | Karbon Monoksida (CO) | 1 jam | 10.000 | K |
| | | 8 jam | 4.000 | K |
| 3 | Nitrogen Dioksida ($NO_2$) | 1 jam | 200 | KM |
| | | 24 jam | 65 | K |
| | | 1 tahun | 50 | K |
| 4 | Oksidan / Ozon ($O_3$) | 1 jam | 150 | KM |
| | | 8 jam | 100 | K |
| | | 1 tahun | 35 | K |

| | | | | |
|---|---|---|---|---|
| **5** | Partikulat debu < 100 um (TSP) | 24 jam | 230 | M |
| | Partikulat debu < 10 um ($PM_{10}$) | 24 jam | 75 | KM |
| | | 1 tahun | 40 | K |
| | Partikulat debu < 2.5 um ($PM_{2.5}$) | 24 jam | 55 | KM |
| | | 1 tahun | 15 | K |
| **6** | Hidro Karbon non Metan (NMHC) | 3 jam | 160 | K |
| **7** | Timbal (Pb) | 24 jam | 2 | M |

*Sumber: Peraturan Republik Indonesia 2021.*

keterangan :

(1) $\mu g/m^3$ = konsentrasi dalam mikrogram per meter kubik, pada kondisi normal, yaitu tekanan (P) 1 $atm$ dan temperatur (T) 25℃.

(2) Baku Mutu adalah ukuran maksimum polutan yang diperkenankan.

(3) Proses : K – Kontinu M – Manual.

Tabel 3.2 diatas memberikan informasi batasan baku mutu udara yang telah di terapkan pemerintah. Tabel ini mengacu angka tiap komponen untuk masuk kedalam perhitungan tahap konversi dan koreksi. Perhitungan tersebut merupakan batasan dalam hitungan dari satuan waktu, tekanan, dan temperatur. Berdasarkan tabel ini dapat ditetapkan suhu standar dan tekanan standar yang bisa digunakan untuk perhitungan dengan masukkan data dari sensor.

Selanjutnya adalah perhitungan untuk perbandingan yang diperuntukkan untuk masuk kategori tingkat keberbahayaan dari tiap unsur. Menurut Kementerian Lingkungan Hidup dan Kehutanan (2020) Peraturan yang berhubungan dengan ini bisa dikaitkan dengan Peraturan Menteri Nomor P.14/MENLHK/SETJEN/KUM.1/7/2020 Tentang Indeks Standar

Pencemar Udara (ISPU). Perhitungan di mulai setelah mendapatkan data tiap parameter zat yang sudah sesuai standar baku mutu. Setelahnya akan di rerata berdasarkan perhitungan selama 24 jam dari tiap parameter zat. Akhir dari perhitungannya adalah masuk kedalam perhitungan ISPU dari hasil rata-rata selama 24 jam untuk tiap zat parameter.

| ISPU | PM10 | PM2.5 | SO2 | CO | O3 | NO2 | HC | warna | kategori |
|---|---|---|---|---|---|---|---|---|---|
| 0 – 50 | 50 | 15.5 | 52 | 4,000 | 120 | 80 | 45 | | baik |
| 51 - 100 | 150 | 55.4 | 180 | 8,000 | 235 | 200 | 100 | | sedang |
| 101 - 200 | 350 | 150.4 | 400 | 15,000 | 400 | 1,130 | 215 | | tidak sehat |
| 201 - 300 | 420 | 250.4 | 800 | 30,000 | 800 | 2,260 | 432 | | sangat tak sehat |
| >300 | 500 | 500 | 1,200 | 45,000 | 1,000 | 3,000 | 648 | | berbahaya |

**Gambar 3.1 Batas Indeks Standar Pencemar Udara**
Kementerian Lingkungan Hidup dan Kehutanan (2020)

Pada Gambar 3.1 ditampilkan sebuah tabel yang berisikan kategori dari keberbahayaan dari tiap elemen. Tabel ini berisikan jangkauan ukur ISPU dari yang terkecil hingga terbesar. Dari tiap jangkauan akan memiliki kategori yang berbeda dan memiliki pewarnaan sebagai informasi. Untuk perhitungan ISPU akan mengacu pada tabel ini, yang mana akan digunakannya batas awal dan batas akhir. Batas ini akan berbeda tiap elemen, dan juga batasan ini telah disesuaikan dengan aturan yang berlaku.

Suatu perhitungan atau indeks yang mengukur tingkat polusi udara di suatu wilayah pada suatu periode waktu tertentu. Maksud dari perhitungan ISPU adalah untuk memberikan gambaran atau informasi mengenai tingkat kualitas udara dalam suatu wilayah, dengan memperhatikan beberapa parameter pencemar udara yang umumnya memiliki dampak buruk terhadap kesehatan manusia dan lingkungan.

### 3.2.2 Spesifikasi Kebutuhan Sistem

Berikut akan dipaparkan beberapa penjelasan dari kebutuhan yang berkaitan dengan sistem:

(1) Spesifikasi kebutuhan input data ke system

Sistem ini perlu mendapatkan perintah yang berupa input permintaan data. Input permintaan ini didasarkan dengan cara yang dikhususkan oleh sensor yang berkaitan. Dengan menggunakan Modbus RTU (*Remote Terminal Unit*) merupakan varian Modbus yang umum digunakan pada komunikasi serial dengan bentuk yang ringkas. Format RTU menggunakan mekanisme *Cyclic Redundancy Check* (CRC) untuk memastikan data yang dikirim lengkap dan benar. Adapun penjelasan tentang input yang mampu diterima oleh sensor sebagai berikut:

| Address | operation | Contents | Notes |
|---|---|---|---|
| 0x0001 | read-only | Noise, Hexadecimal number is magnified as 10 times, such as 0x021C indicates noise of 54.0db | |
| 0x0002 | read-only | Keep | |
| 0x0003 | read-only | SO2 concentration,hexadecimal ,such as 0x0172, indicates that SO2 concentration is 370ppb. | |
| 0x0004 | read-only | NO2 concentration,hexadecimal ,such as 0x0036, indicates that NO2 concentration is 54ppb | |
| 0x0005 | read-only | CO concentration,hexadecimal ,such as 0x0A00, indicates that CO concentration is 2560ppb | |
| 0x0006 | read-only | O3 concentration,hexadecimal ,such as 0x0123, indicates that O3 concentration is 2560ppb | |
| 0x0007 | read-only | PM2.5 concentration,hexadecimal ,such as 0x0172, indicates that PM2.5 concentration is 370ug/m³ | |

**Gambar 3.2 Spesifikasi Register MODBUS Standar**
Handan Yunnong Intelligent Agricultural Technology Co., Ltd. (2020)

| | | | |
|---|---|---|---|
| 0x0008 | read-only | PM10 concentration,hexadecimal ,such as 0x0193, indicates that PM10 concentration is 403ug/m³ | |
| 0x0009 | read-only | Atmospheric temperature, Hexadecimal number plus 40 then it is magnified as 100 times, such as 0x1B00 indicates 6912/100-40=29.12℃ | |
| 0x000A | read-only | Atmospheric humidity, Hexadecimal number is magnified as 100 times, such as 0x1603 indicates 5635/100=56.35% | |
| 0x000B | read-only | Atmospheric tempress, Hexadecimal number is magnified as 10 times, such as 0x2784 indicates 10116/10=1011.6hPa | |
| 0x000C | read-only | Wind speed, Hexadecimal number is magnified as 100 times, such as 0x0125 indicates 293/100=2.93m/s | |
| 0x000D | read-only | Wind direction, Hexadecimal number is magnified as 10 times, such as 0x0C14 indicates 3092/10=309.2° | North is 0 ° |

**Gambar 3.3 Spesifikasi Register MODBUS Standar**
Handan Yunnong Intelligent Agricultural Technology Co., Ltd. (2020)

| | | | |
|---|---|---|---|
| 0x000E | read-only | Rainfall, Hexadecimal number is magnified as 10 times, such as 0x0C14 indicates 2.2mm | The default is 10 minutes of rain |
| 0x000F | read-only | solar radiation, Hexadecimal number, such as 0x0172 indicates 370W/㎡ | |
| 0x0010 | read-only | illumination, Hexadecimal number is magnified as 100 times, such as 0x0123 indicates 2.91.Klux | |
| 0x0011 | read-only | ultraviolet ray, Hexadecimal number, such as 0x05 indicates 5 | |
| 0x0012 | read-only | CO2, Hexadecimal number, such as 0x01F4 indicates 500ppm | |

**Gambar 3.4 Spesifikasi Register MODBUS Standar**
Handan Yunnong Intelligent Agricultural Technology Co., Ltd. (2020)

| | | | |
|---|---|---|---|
| 0x0013-0x001f | read-only | Keep | |
| 0x0105 | read-only | electronic compass, Hexadecimal number, such as 0x0036 indicates 54° | North is 0 |

**Gambar 3.5 Spesifikasi Register MODBUS Standar**
Handan Yunnong Intelligent Agricultural Technology Co., Ltd. (2020)

Pada Gambar 3.2 hingga Gambar 3.5 menjelaskan bahwa spesikasi data yang disediakan dari sensor. Dapat terlihat elemen-elemen apa saja yang dapat user minta terhadap sensor. Beberapa elemen yang dapat diambil antara lain $CO$ (Karbon Monoksida), $SO2$ (Sulfur Dioksida), $NO2$ (Nitrogen Dioksida), $O3$ (Ozon), $PM2.5$ (Particulate Matter), $PM10$ (Particulate Matter), Atmospheric Temperatur, Atmospheric Humidity, Atmospheric Tempress, Wind Speed, Wind Direction, Rainfall, Solar Radiation, Illumination, Ultraviolet Ray, $CO2$ (Karbon Dioksida), dan Electronic Compass. Dengan spesifikasi register Modbus standar ini user perlu mengirimkan permintaan terhadap sensor dengan kesesuaian yang telah ditentukan.

Berikut tampilan contoh untuk data *input* kepada sensor seperti yang telah ditetapkan oleh aturan sensor.

Send: FF 03 00 09 00 05 40 15

| FF | 03 | 00 09 | 00 05 | 40 15 |
|---|---|---|---|---|
| System address | function code | Register address | Register number | CRC16 check bit generated automatically by software |

Answer: FF 03 0A 1A 57 0C 5F 27 83 00 00 0C 14 36 E0

| FF | 03 | 0A | 1A 57 0C 5F 27 83 00 00 0C 14 | 36 E0 |
|---|---|---|---|---|
| System address | function code | Number of bytes of data segment | Data segment data | CRC16 check bit |

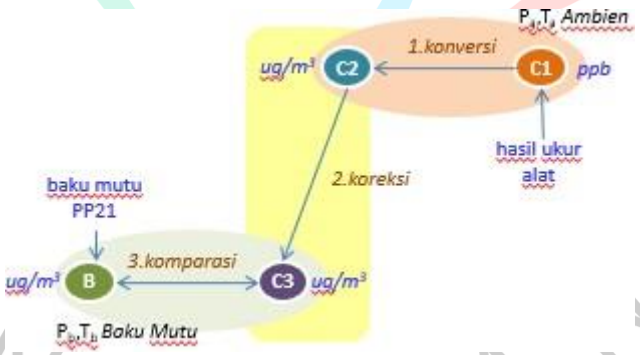
**Gambar 3.6 Contoh Spesifikasi Register MODBUS Standar**
Handan Yunnong Intelligent Agricultural Technology Co., Ltd. (2020)

Pada Gambar 3.6 merupakan contoh *string* data yang dikirim kepada sensor. FF 03 00 09 00 05 40 15 merupakan data hexadecimal yang dikirim menggunakan protokol Modbus RTU. FF merupakan alamat sistem, 03 merupakan kode fungsi *read*, 00 09 merupakan data awal yang akan di ambil, 00 05 merupakan banyaknya turunan data dari data awal yang mana seperti contoh ini berarti ada 5 data lanjut dari 00 09, dan terakhir 40 15 merupakan *check bit* untuk memverifikasi data agar valid, yang mana data ini terisi otomatis menggunakan pemrograman tambahan.

Sehingga balasan dari sensor akan mengirimkan data hexadecimal pula yang telah mencangkup nilai dari data yang diminta. FF 03 0A 1A 57 0C 5F 27 83 00 00 0C 14 36 E0 merupakan contoh data balasan berupa hexadecimal dari sensor. FF merupakan alamat sistem, 03 merupakan kode fungsi *read*, 0A merupakan banyaknya segmen data yang dikirim, 1A 57 0C 5F 27 83 00 00 0C 14 merupakan data dari tiap segmen yang diminta dan tiap data merupak dua pasang hexadecimal, dan terakhir 36 E0 *check bit* dari sensor yang menyatakan data valid.

(2) Spesifikasi kebutuhan output data dari sistem

Alur untuk mendapatkan data, adalah mengirim permintaan data ke sensor. Tujuan permintaan data ke sensor adalah untuk mengkategorikan beberapa data yang perlu ditarik dari sensor. Setelahnya sensor akan merespon dengan mengirimkan data sesuai permintaan. Data dari sensor perlu pengolahan, untuk menyesuaikan dengan peraturan baku mutu pemerintah dan dapat dikategorikan kedalam ISPU sebagai bahan informasi.




**Gambar 3.7 Alur Proses Pengolahan Data**

Tahapan proses pengolahan data dapat terlihat dalam Gambar 3.7. Perhitungan dalam penyesuaian data untuk menampilkan informasi mutu dari udara lingkungan mengacu pada aturan yang telah disesuaikan oleh pemerintah dan peraturan menteri. Penjelasan sederhana mengenai perhitungan ini diawali dari C1 yang mana data berasalkan hasil dari *output* sensor. Data

ini perlu dikonversi dikarenakan data yang berasal dari sensor satuannya tidak sesuai yaitu ppb (Parts Per Billion) dengan baku mutu yaitu $\mu g/m^3$ sehingga data hasil konversi merupakan hasil penyesuaian satuan. C2 merupakan hasil dari penyesuaian konversi berdasarkan berat, volume, dan banyaknya molekul elemen. Selanjutnya untuk C3 merupakan gabungan perhitungan dengan melibatkan suhu dan tekanan standar dengan baku mutu untuk mencapai tahapan koreksi. Sehingga dari gabungan antar rumus yang berkaitan dapat disederhanakan untuk menghitung C3 adalah sebagai berikut pada Gambar 3.8.

$$C3 = \frac{273.15}{298.15} \times \frac{B_{Mt}}{22.414} \times C1 \quad [ug/m^3]$$

**Gambar 3.8 Rumus Perhitungan Data C3**

Dengan contoh perhitungan sebagai berikut: Hasil pengukuran gas $SO2$ pada suatu siang di suatu lokasi di Bandung, menunjukkan angka $69\ ppb$. Temperatur di lokasi tsb $34°C$ sedangkan tekanan atmosfirnya $919\ mbar$. Berapakah konsentrasi gas $SO2$ tersebut dinyatakan dalam $\mu g/m^3$ pada tekanan dan temperatur baku?

$$BMt = Berat\ 1\ Mol\ SO2 = 1\ x\ Berat\ atom\ S + 2\ x\ Berat\ atom\ O$$

$$= 1\ x\ 32.066 + 2\ x\ 15.999$$

$$= 64.064\ g$$

$$C3 = 0.916\ x\ 64.064 / 22.414\ x\ 69$$

$$= 0.916\ x\ 2.859\ x\ 69$$

$$= 180.65\ \mu g/m^3$$

Setelah mendapatkan hasil data dari seluruh zat/elemen dari polutan perlu untuk dihitung rerata berdasarkan waktu yaitu 24 jam. Selanjutnya nilai tersebut akan masuk kedalam perhitungan ISPU. Dengan perhitungan ini tiap-tiap dari zat/elemen akan masuk kedalam kategori dari tingkat keberbahayaannya, perhitungan mengacu pada Peraturan Menteri Nomor P.14/MENLHK/SETJEN/KUM.1/7/2020 Tentang Indeks Standar Pencemar Udara (ISPU).

$$I = \frac{(I_a - I_b)}{(X_a - X_b)} (X_x - X_b) + I_b \quad \text{.....(1)}$$

I = ISPU terhitung
Ia = ISPU batas atas
Ib = ISPU batas bawah
Xa = Konsentrasi ambien batas atas (µg/m³)
Xb = Konsentrasi ambien batas bawah (µg/m³)
Xx = Konsentrasi ambien nyata hasil pengukuran (µg/m³)

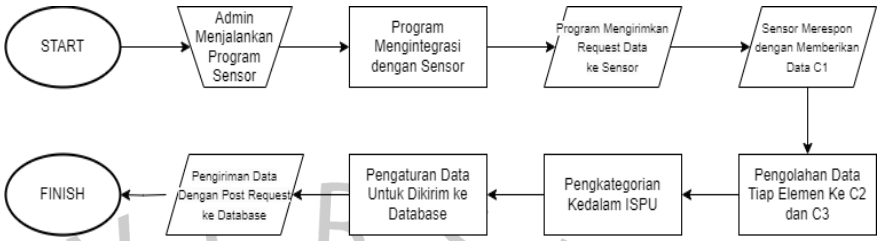
**Gambar 3.9 Rumus Perhitungan ISPU**

Pada perhitungan ISPU ini ditujukan untuk mendapat tingkat keberbahayaan dari tiap zat/elemen. Beberapa konstanta dalam rumus perhitungan ISPU ada pada Gambar 3.9 Setelah perhitungan ISPU inilah yang semua data akan dikirim kedalam *database*. Dengan data ini akan masuk kedalam tahapan *output*.

Pada akhirnya sistem akan menghasilkan *output* data kualitas udara ambien secara waktu nyata yang sudah disesuaikan. *Output* berupa data yang akan ditampilkan oleh bagian *Front-end*. Data yang dikirim semua zat/elemen dari C1 (data sensor), C2 (data konversi), C3 (data koreksi), data hasil komparasi ISPU (B), dan beberapa data dukung lainnya. Data yang diberikan akandigunakan dalam beberapa hal berikut:

(1) Grafik berdasarkan waktu nyata kualitas udara untuk setiap parameter.
(2) Pemberitahuan atau peringatan otomatis jika tingkat polutan melebihi ambang batas yang ditentukan.
(3) Laporan harian/mingguan/bulanan tentang kualitas udara.

(3) Rekasyasa proses




**Gambar 3.10 Diagram Alir Sistem Monitoring Kualitas Udara Ambien**

Admin menjalankan program yang akan mengintegrasikan rekayasa proses ini dengan sensor dan *database*. Program ini akan mengeksekusi perintah yang ditetapkan untuk sensor untuk mengambil data sensor (C1). Program akan memproses data yang diterima dari sensor dengan perhitungan menjadi data C2 dan C3, setelahnya program mengeksekusi perintah untuk mengatur data berdasarkan pengaturan yang telah ditetapkan. Program akan mengkategorikan data ke dalam peraturan ISPU dan mengirimkan data ke *database* melalui metode POST. Data yang telah dikirim akan diproses oleh *database*, dan Admin dapat memantau hasil proses dan memverifikasi bahwa data telah berhasil dikirim ke *database*.

### 3.2.3 Implementasi

(1) Spesifikasi kebutuhan perangkat lunak

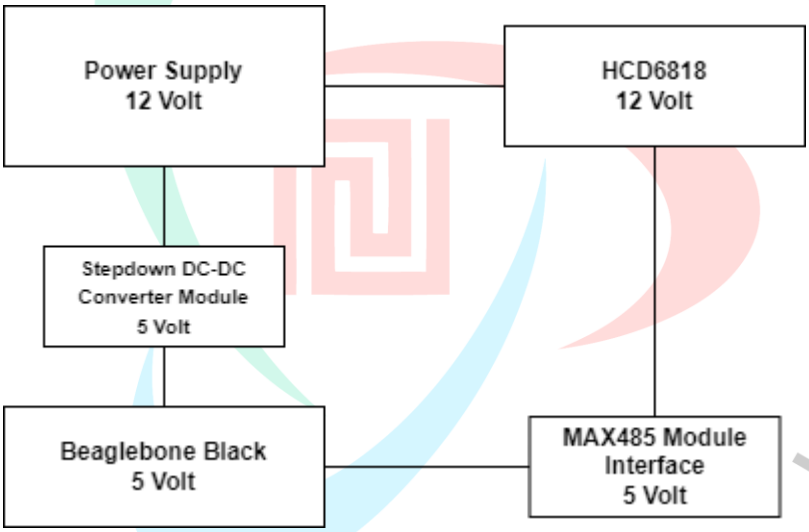
Perangkat lunak yang praktikan gunakan adalah Cloud9 IDE. Cloud9 IDE adalah lingkungan pengembangan terintegrasi (IDE) berbasis web yang digunakan dalam proyek ini. Fungsi utama dari Cloud9 adalah sebagai tempat pembuatan, penyuntingan, dan pelaksanaan program yang dikembangkan untuk mikrokontroller Beaglebone. Cloud9 digunakan sebagai IDE utama untuk pengembangan perangkat lunak pada mikrokontroller Beaglebone.

Fitur Utama:

(1) Penyuntingan Kode: Fasilitas penyuntingan kode dengan berbagai fitur penyorotan sintaks dan otomatisasi.
(2) Debugging: Kemampuan untuk melakukan debugging secara efisien pada kode yang dikembangkan.
(3) Integrasi Git: Mendukung manajemen versi menggunakan Git untuk kontrol revisi proyek.
(4) Terminal: Terdapat terminal terintegrasi untuk menjalankan perintah sistem dan skrip.

(2) Spesifikasi kebutuhan perangkat keras

Beberapa perangkat keras yang praktikan gunakan selama program kerja profesi adalah sebagai berikut.




**Gambar 3.11 Blok Diagram Perangkat Monitoring Kualitas Udara**

(1) PSU (Power Supply Unit) - 12V: Ini adalah sumber daya utama sistem yang menyediakan tegangan 12V. PSU digunakan sebagai input daya utama untuk sistem elektronik.
(2) Stepdown DC-DC Converter Module (5V): Modul konverter DC-DC ini berfungsi mengubah tegangan dari PSU (12V) menjadi 3V. Langkah ini dilakukan untuk menurunkan tegangan ke tingkat yang sesuai dengan kebutuhan komponen-komponen berikutnya dalam sistem.

(3) Mikrokontroller Beaglebone Black(5V): Mikrokontroller Beaglebone Black (BBB) adalah papan pengembangan open-source yang kompak dan kuat, dirancang oleh BeagleBoard.org. Dilengkapi dengan processor ARM Cortex-A8 @ 1GHz, 512MB DDR3 RAM, dan 4GB eMMC Flash Storage, BBB mendukung berbagai antarmuka seperti USB, HDMI, Ethernet, dan GPIO pins. Dukungan untuk Linux, Wi-Fi, dan Bluetooth menjadikannya ideal untuk IoT, sistem kendali, dan proyek perangkat keras lainnya. Kelebihannya meliputi ukuran fisik yang kecil, dukungan bahasa pemrograman seperti Python, C, dan JavaScript, serta komunitas yang besar. Integrasi dengan Cloud9 IDE memungkinkan pengembang mengakses dan mengelola program secara efisien melalui platform berbasis web. BBB dengan Cloud9 IDE memberikan platform fleksibel untuk prototyping dan pengembangan sistem tertanam.

(4) MAX485 Module Interface: Interface ini digunakan untuk menghubungkan BeagleBone dengan perangkat atau sistem eksternal yang menggunakan komunikasi RS-485. MAX485 adalah chip transceiver RS-485 yang memungkinkan komunikasi serial antar perangkat.

(5) HCD6818 (12V): HCD6818 merupakan sensor yang berfokus untuk mengukur berbagai paramater metereologi untuk polusi udara. dengan mengukur beberapa parameter, yaitu : Partikel padat (PM2.5 dan PM10), gas (CO, SO2, NO2, dan O3), temperature, kelembapan atmosfir, dan tekanan atmosfir. Integrasi dengan sensor ini juga didukung dengan mikrokontroller beaglebone. Dengan algoritma yang diterapkan di mikrokontroller digunakan untuk menghubungkan perangkat, mengirim inputan, dan memperoleh output data.

| Measture item | Principle | Range | Resolution | Accuracy |
|---|---|---|---|---|
| Atmospheric temperature | MEMS | -40-60℃ | 0.1℃ | ±0.3℃（25℃） |
| Atmospheric humidity | MEMS | 0-100%RH | 0.1% | ±3%RH |
| PM2.5 | Laser way | 0-1000ug/m³ | 1ug/m³ | ±10%(Measurements) |
| PM10 | Laser way | 0-1000ug/m³ | 1ug/m³ | ±10%(Measurements) |
| CO | Electrochemical principle | 0-12.5ppm | ≤10ppb | ±1.5%F.S |
| SO2 | Electrochemical principle | 0-2000ppb | ≤10ppb | ±1.5%F.S |
| NO2 | Electrochemical principle | 0-2000ppb | ≤10ppb | ±1.5%F.S |
| O3 | Electrochemical principle | 0-2000ppb | ≤10ppb | ±1.5%F.S |
| Option | Wind speed, wind direction, atmospheric pressure, rainfall, H2S, noise, sunshine radiation, AQI index, ultraviolet ray, noise | | | |

**Gambar 3.12 Spesifikasi Teknis Sensor HCD6818**
Handan Yunnong Intelligent Agricultural Technology Co., Ltd. (2020)

| | |
|---|---|
| Supply | 12-24VDC |
| Output | RS485(Modbus) |
| fixed form | Sleeve fixed, flange fixed |
| Power consumption | <1w@12V |
| Ingress Protection | IP65 |
| Work Environment | Temperature: -40-70℃; Humidity: 5-90%RH |
| Add function | ● data collector (net port, GPRS)<br>● Fixed brackets (1.5m, 3m) constitute a complete set of monitoring station<br>●B/S architecture software platform, mobile APP client software (android version only)<br>● Solar power system |
| Main material | ASA+engineering plastics |
| Mounting frame | 1.5m, 3m Flange, 1.8m tripod |
| GPS | Device tracking and positioning |
| Data Release | PC、Telephone APP、LED、LCD TV display, etc |

**Gambar 3.13 Spesifikasi Teknis Sensor HCD6818**
Handan Yunnong Intelligent Agricultural Technology Co., Ltd. (2020)

Spesifikasi dari sensor HCD6818 menggambarkan tentang kemampuan dan kebutuhan dari sensor dalam pemakaian. Berdasarkan spesifikasi dari gambar di atas sensor ini mempuni cangkupan beberapa zat/elemen yang ditangkap, jangkauan, bobot, dan tingkat akurasi dari kemampuan sensor. Dijelaskan pula spesifikasi perangkat berupa cangkupan daya, kemampuan di lingkungan, dan kebutuhan sistem.

Dengan demikian, alur daya dan sinyal dalam sistem dimulai dari sumber daya utama (PSU) pada tegangan 12V,

diubah menjadi 3V oleh konverter DC-DC untuk memberikan daya pada Mikrokontroller. Dari Mikrokontroller, sinyal kemudian diarahkan melalui interface MAX485 untuk berkomunikasi dengan perangkat eksternal (HCD6818), yang juga menerima daya dari PSU pada tegangan 12V.

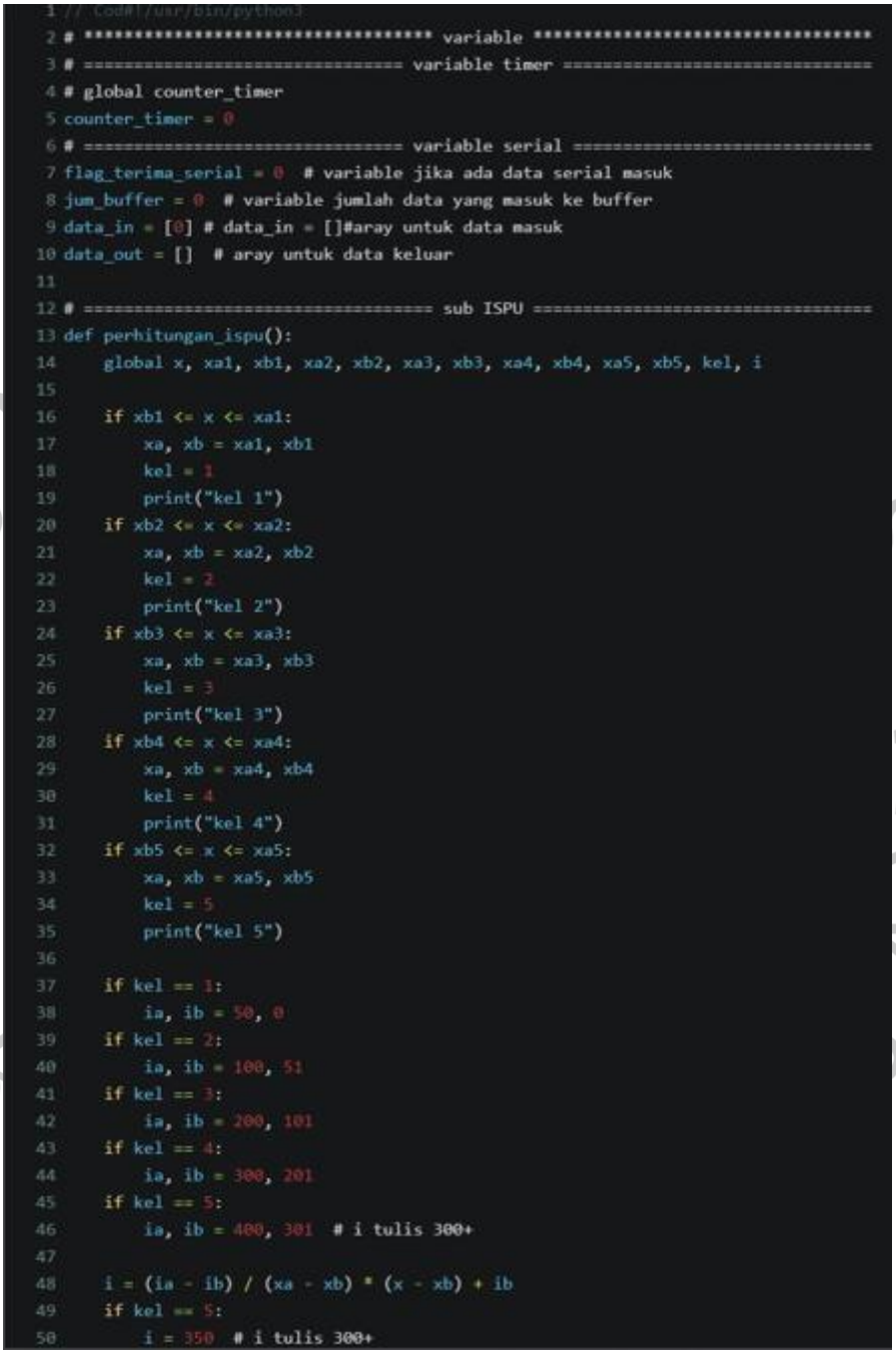
(3) Bentuk Perangkat

**Gambar 3.14 Sensor HCD6818**
Handan Yunnong Intelligent Agricultural Technology Co., Ltd. (2020)

**Gambar 3.15 Bentuk Asli Perangkat**

Berdasarkan gambar di atas merupakan bentuk nyata dari blok diagram yang telah dirancang. Bentuk nyata dari perangkat ini perlu penyesuaian untuk dapat menghadapi berbagai kondisi dilapangan. Pembuatan perangkat yang memumpuni diperlukan pembuatan khusus yang diproduksi langsung dari tim produksi PT. MARKTEL. Perangkat ini dibuat berdasarkan perhitungan untuk skala industri, sehingga ada acuan perlindungan hukum atas rancangan yang dimaksudkan.

(4) Kode program berikut penjelasannya



```
# ********************************** variable **********************************
# ================================ variable timer ================================
# global counter_timer
counter_timer = 0
# ================================ variable serial ================================
flag_terima_serial = 0  # variable jika ada data serial masuk
jum_buffer = 0  # variable jumlah data yang masuk ke buffer
data_in = [0] # data_in = []#aray untuk data masuk
data_out = []  # aray untuk data keluar

# ================================ sub ISPU ================================
def perhitungan_ispu():
    global x, xa1, xb1, xa2, xb2, xa3, xb3, xa4, xb4, xa5, xb5, kel, i

    if xb1 <= x <= xa1:
        xa, xb = xa1, xb1
        kel = 1
        print("kel 1")
    if xb2 <= x <= xa2:
        xa, xb = xa2, xb2
        kel = 2
        print("kel 2")
    if xb3 <= x <= xa3:
        xa, xb = xa3, xb3
        kel = 3
        print("kel 3")
    if xb4 <= x <= xa4:
        xa, xb = xa4, xb4
        kel = 4
        print("kel 4")
    if xb5 <= x <= xa5:
        xa, xb = xa5, xb5
        kel = 5
        print("kel 5")

    if kel == 1:
        ia, ib = 50, 0
    if kel == 2:
        ia, ib = 100, 51
    if kel == 3:
        ia, ib = 200, 101
    if kel == 4:
        ia, ib = 300, 201
    if kel == 5:
        ia, ib = 400, 301  # i tulis 300+

    i = (ia - ib) / (xa - xb) * (x - xb) + ib
    if kel == 5:
        i = 350  # i tulis 300+
```

**Gambar 3.16 Tampilan Baris Kode Program**

Pada bagian Gambar 3.16 kode program menunjukkan beberapa deklarasi untuk variabel yang dibutuhkan. Variabel ini ditujukan untuk pemrosesan data dan pengolahan data dari sensor. Selanjutnya ada fungsi untuk pengukuran atau penggolongan ISPU.

Perhitungan ini didasari dari peraturan menteri tentang indeks standar pencemar udara (ISPU). Penggolongan ini ada 5 kelompok yang dimulai dari kelompok 1 yang berarti baik hingga kelompok 5 yaitu berbahaya. Hasil dari perhitungan inilah yang akan menjadi bahan informasi yang akan ditampilkan.

```
# ********************************** setup **********************************
import threading
import time
import Adafruit_BBIO.UART as UART
import serial

UART.setup("UART1")
# with serial.Serial(port = "/dev/ttyO1", baudrate=9600) as ser:
ser = serial.Serial(
    port="/dev/ttyO1",
    baudrate=9600,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS,
    timeout=None
)

# ============================ setup crc 16 modbus ============================
from modbus_crc import add_crc  # untuk bikin crc
from modbus_crc import check_crc  # untuk cek crc
# ========================== akhir setup crc 16 modbus ========================

# ============================ setup send data ================================
import sqlite3
import requests
from datetime import datetime
import json

time.sleep(1)  # delay 1 detik

# ********************************* task timer *********************************
def timer():
    while 1:
        global counter_timer
        time.sleep(0.01)  # 1=1detik
        counter_timer = counter_timer + 1
```

**Gambar 3.17 Tampilan Baris Kode Program**

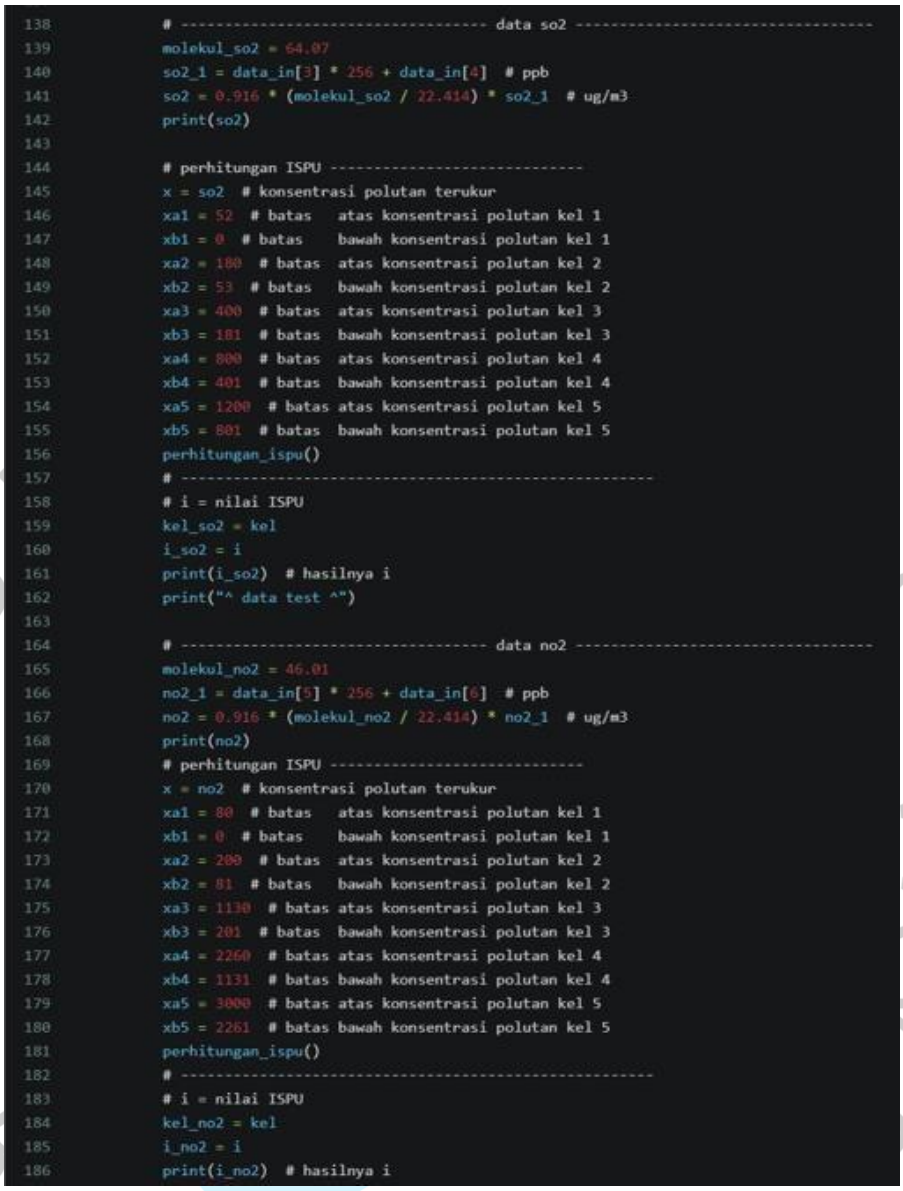
Baris kode pada Gambar 3.17 merupakan bagian untuk mengatur koneksi kepada sensor. Dituliskan beberapa kebutuhan *library* untuk kebutuhan dalam koneksi ke sensor. Sensor yang dipakai menggunakan saluran UART1 dan *port* /dev/tty01 pada mikrokontroller beaglebone, Modul Adafruit_BBIO.UART digunakan untuk mengatur UART pada perangkat BeagleBone. Selanjutnya, objek serial (ser) dibuat dengan parameter seperti *port, baudrate*, dan sebagainya. Setelah deklarasi tujuan koneksi, dibutuhkan juga *library* untuk pengolahan data yang berupa CRC dikhususkan untuk penambahan dan pengecekan. Bagian ini mengimpor fungsi-fungsi *add_crc* dan *check_crc* dari modul *modbus_crc*. CRC digunakan untuk memastikan integritas data dalam protokol Modbus.

Setelahnya deklarasi dalam penggunaan *library database* Sqlite3. Pada proyek ini kami menggunakan Sqlite3 dalam penyimpanan dan pengolahan data. *requests* digunakan untuk mengirim permintaan HTTP. *Datetime* digunakan untuk mendapatkan informasi waktu, dan Json untuk bekerja dengan data JSON. Selanjutnya ada fungsi untuk pengaturan waktu agar pengambilan data dapat terukur dan teratur.

```
def serial_1():
    while 1:

        global data_in
        # ============================== terima serial ==============================
        jum_buffer = ser.inWaiting()  # ambil jumlah data serial masuk di buffering
        if jum_buffer > 0:  # jika ada data serial masuk
            data_in = ser.read(jum_buffer)  # terima data dalam hex
            print(data_in.hex())  # print dalam hex
            # ------------------------------ cek crc modbus ------------------------------
            data_in = bytes(data_in) # ubah aray ke tipe data byte untuk bisa d proses crc
            if not check_crc(data_in):
                data_in.clear()  # kosongkan isi aray

            if check_crc(data_in):
                print("data di terima sukses")

# ********************************* task loop *********************************
def loop():
    while 1:
        global counter_timer, data_out, global data_in
        # untuk ispu
        global x, xa1, xb1, xa2, xb2, xa3, xb3, xa4, xb4, xa5, xb5, kel, i

        # ============================== kirim serial ==============================
        if counter_timer >= 1200:
            # kirim untuk minta semua data
            data_out = []
            data_out.insert(0, 0xff)
            data_out.insert(1, 0x03)
            data_out.insert(2, 0x00)
            data_out.insert(3, 0x03)  # nomer data yg d minta (mulai dari kalau lebih dr 1)#3
            data_out.insert(4, 0x00)
            data_out.insert(5, 0x10)  # jumlah data yg di minta#10

            data_out = bytes(data_out) # ubah aray ke tipe data byte untuk bisa d proses crc
            signed_package = add_crc(data_out) # tambahkan data crc nya
            data_out = signed_package
            ser.write(data_out) # kirim data lewat serial
            print(data_out)
            print(data_in[0])
            counter_timer = 0  # reset counter timer

        # ========================= olah data dari sensor =========================
        if data_in[0] == 0xff:  # jika data dari sensor awal ff (ok)
            a = len(data_in)  # untuk tau jumlah isi data aray
            conn = sqlite3.connect('/var/lib/cloud9/ProjekLingkungan/sensor.db')
            cursor = conn.cursor()

```

**Gambar 3.18 Tampilan Baris Kode Program**

Pada fungsi serial_1 disinilah proses penerimaan serial data dari sensor. Jika selama data masuk maka data serial akan di ambil dan ditampung kedalam variabel *buffer*. Selama data bukan sama dengan nol maka data akan dibaca dan dirubah tipe data menjadi *bytes*, jika tidak maka akan dikosongkan. Jika sukses maka data sensor berhasil diterima. Data ini akan bisa diterima ketika proses permintaan kepada sensor suda dilakukan.

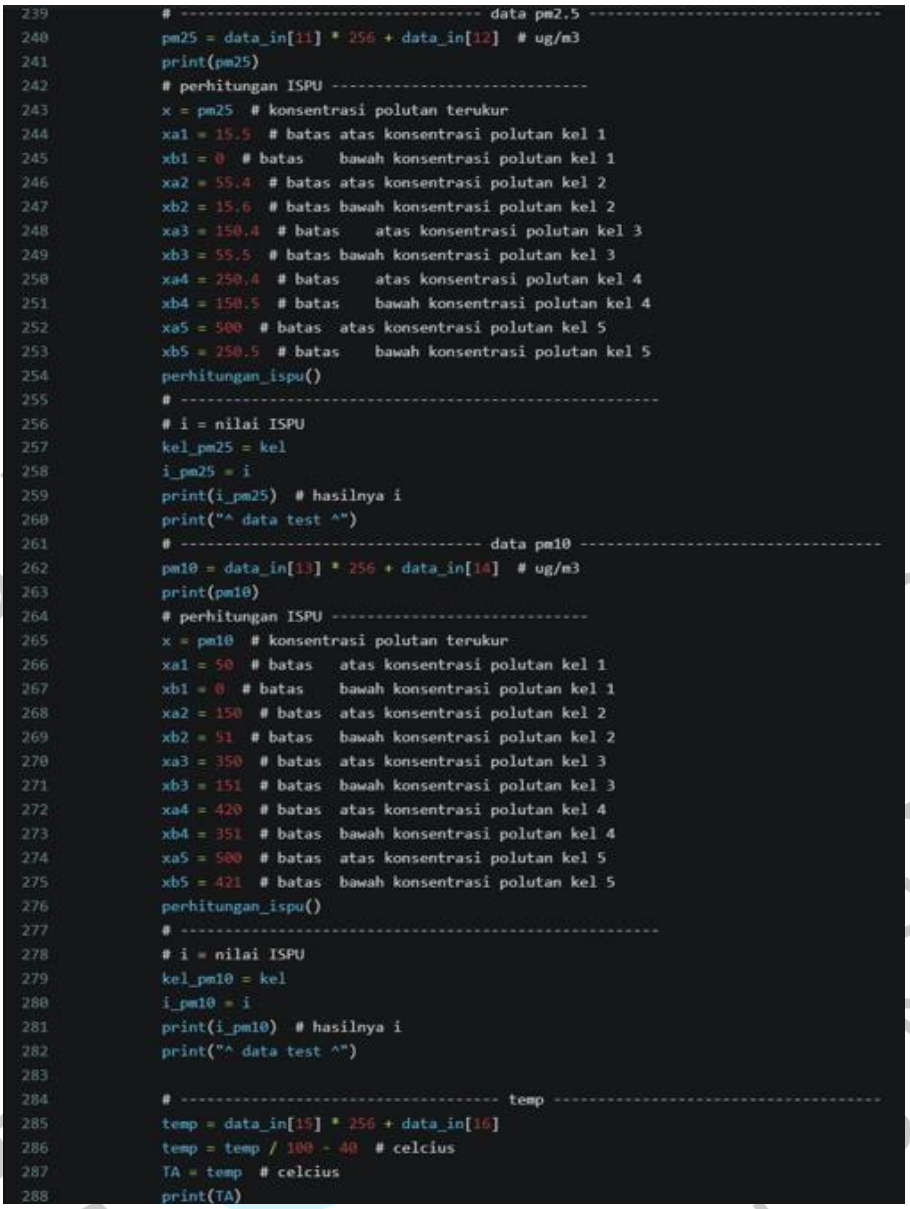


```
138         # ---------------------------------- data so2 ----------------------------------
139         molekul_so2 = 64.07
140         so2_1 = data_in[3] * 256 + data_in[4]  # ppb
141         so2 = 0.916 * (molekul_so2 / 22.414) * so2_1  # ug/m3
142         print(so2)
143
144         # perhitungan ISPU ----------------------------
145         x = so2  # konsentrasi polutan terukur
146         xa1 = 52  # batas   atas konsentrasi polutan kel 1
147         xb1 = 0  # batas    bawah konsentrasi polutan kel 1
148         xa2 = 180  # batas  atas konsentrasi polutan kel 2
149         xb2 = 53  # batas   bawah konsentrasi polutan kel 2
150         xa3 = 400  # batas  atas konsentrasi polutan kel 3
151         xb3 = 181  # batas  bawah konsentrasi polutan kel 3
152         xa4 = 800  # batas  atas konsentrasi polutan kel 4
153         xb4 = 401  # batas  bawah konsentrasi polutan kel 4
154         xa5 = 1200  # batas atas konsentrasi polutan kel 5
155         xb5 = 801  # batas  bawah konsentrasi polutan kel 5
156         perhitungan_ispu()
157         # ----------------------------------------------------
158         # i = nilai ISPU
159         kel_so2 = kel
160         i_so2 = i
161         print(i_so2)  # hasilnya i
162         print("^ data test ^")
163
164         # ---------------------------------- data no2 ----------------------------------
165         molekul_no2 = 46.01
166         no2_1 = data_in[5] * 256 + data_in[6]  # ppb
167         no2 = 0.916 * (molekul_no2 / 22.414) * no2_1  # ug/m3
168         print(no2)
169         # perhitungan ISPU ----------------------------
170         x = no2  # konsentrasi polutan terukur
171         xa1 = 80  # batas   atas konsentrasi polutan kel 1
172         xb1 = 0  # batas    bawah konsentrasi polutan kel 1
173         xa2 = 200  # batas  atas konsentrasi polutan kel 2
174         xb2 = 81  # batas   bawah konsentrasi polutan kel 2
175         xa3 = 1130  # batas atas konsentrasi polutan kel 3
176         xb3 = 201  # batas  bawah konsentrasi polutan kel 3
177         xa4 = 2260  # batas atas konsentrasi polutan kel 4
178         xb4 = 1131  # batas bawah konsentrasi polutan kel 4
179         xa5 = 3000  # batas atas konsentrasi polutan kel 5
180         xb5 = 2261  # batas bawah konsentrasi polutan kel 5
181         perhitungan_ispu()
182         # ----------------------------------------------------
183         # i = nilai ISPU
184         kel_no2 = kel
185         i_no2 = i
186         print(i_no2)  # hasilnya i
```

**Gambar 3.19 Tampilan Baris Kode Program**

```
189             # ---------------------------------- data co ----------------------------------
190             molekul_co = 28.01
191             co_1 = data_in[7] * 256 + data_in[8]  # ppb
192             co = 0.916 * (molekul_co / 22.414) * co_1  # ug/m3
193             print(co)
194             # perhitungan ISPU ----------------------------
195             x = co  # konsentrasi polutan terukur
196             xa1 = 4000  # batas atas konsentrasi polutan kel 1
197             xb1 = 0  # batas    bawah konsentrasi polutan kel 1
198             xa2 = 8000  # batas atas konsentrasi polutan kel 2
199             xb2 = 4001  # batas bawah konsentrasi polutan kel 2
200             xa3 = 15000  # batas    atas konsentrasi polutan kel 3
201             xb3 = 8001  # batas bawah konsentrasi polutan kel 3
202             xa4 = 30000  # batas    atas konsentrasi polutan kel 4
203             xb4 = 1501  # batas bawah konsentrasi polutan kel 4
204             xa5 = 45000  # batas    atas konsentrasi polutan kel 5
205             xb5 = 30001  # batas    bawah konsentrasi polutan kel 5
206             perhitungan_ispu()
207             # ----------------------------------------------------
208             # i = nilai ISPU
209             kel_co = kel
210             i_co = i
211             print(i_co)  # hasilnya i
212             print("^ data test ^")
213
214             # ---------------------------------- data o3 ----------------------------------
215             molekul_o3 = 48
216             o3_1 = data_in[9] * 256 + data_in[10]  # ppb
217             o3 = 0.916 * (molekul_o3 / 22.414) * o3_1  # ug/m3
218             print(o3)
219             # perhitungan ISPU ----------------------------
220             x = o3  # konsentrasi polutan terukur
221             xa1 = 120  # batas  atas konsentrasi polutan kel 1
222             xb1 = 0  # batas    bawah konsentrasi polutan kel 1
223             xa2 = 235  # batas  atas konsentrasi polutan kel 2
224             xb2 = 121  # batas  bawah konsentrasi polutan kel 2
225             xa3 = 400  # batas  atas konsentrasi polutan kel 3
226             xb3 = 236  # batas  bawah konsentrasi polutan kel 3
227             xa4 = 800  # batas  atas konsentrasi polutan kel 4
228             xb4 = 401  # batas  bawah konsentrasi polutan kel 4
229             xa5 = 1000  # batas atas konsentrasi polutan kel 5
230             xb5 = 801  # batas  bawah konsentrasi polutan kel 5
231             perhitungan_ispu()
232             # ----------------------------------------------------
233             # i = nilai ISPU
234             kel_o3 = kel
235             i_o3 = i
236             print(i_o3)  # hasilnya i
237             print("^ data test ^")
```

**Gambar 3.20 Tampilan Baris Kode Program**



```
239         # ---------------------------------- data pm2.5 ----------------------------------
240         pm25 = data_in[11] * 256 + data_in[12]  # ug/m3
241         print(pm25)
242         # perhitungan ISPU ----------------------------
243         x = pm25  # konsentrasi polutan terukur
244         xa1 = 15.5  # batas atas konsentrasi polutan kel 1
245         xb1 = 0  # batas    bawah konsentrasi polutan kel 1
246         xa2 = 55.4  # batas atas konsentrasi polutan kel 2
247         xb2 = 15.6  # batas bawah konsentrasi polutan kel 2
248         xa3 = 150.4  # batas    atas konsentrasi polutan kel 3
249         xb3 = 55.5  # batas bawah konsentrasi polutan kel 3
250         xa4 = 250.4  # batas    atas konsentrasi polutan kel 4
251         xb4 = 150.5  # batas    bawah konsentrasi polutan kel 4
252         xa5 = 500  # batas  atas konsentrasi polutan kel 5
253         xb5 = 250.5  # batas    bawah konsentrasi polutan kel 5
254         perhitungan_ispu()
255         # ----------------------------------------------------
256         # i = nilai ISPU
257         kel_pm25 = kel
258         i_pm25 = i
259         print(i_pm25)  # hasilnya i
260         print("^ data test ^")
261         # ---------------------------------- data pm10 ----------------------------------
262         pm10 = data_in[13] * 256 + data_in[14]  # ug/m3
263         print(pm10)
264         # perhitungan ISPU ----------------------------
265         x = pm10  # konsentrasi polutan terukur
266         xa1 = 50  # batas   atas konsentrasi polutan kel 1
267         xb1 = 0  # batas    bawah konsentrasi polutan kel 1
268         xa2 = 150  # batas  atas konsentrasi polutan kel 2
269         xb2 = 51  # batas   bawah konsentrasi polutan kel 2
270         xa3 = 350  # batas  atas konsentrasi polutan kel 3
271         xb3 = 151  # batas  bawah konsentrasi polutan kel 3
272         xa4 = 420  # batas  atas konsentrasi polutan kel 4
273         xb4 = 351  # batas  bawah konsentrasi polutan kel 4
274         xa5 = 500  # batas  atas konsentrasi polutan kel 5
275         xb5 = 421  # batas  bawah konsentrasi polutan kel 5
276         perhitungan_ispu()
277         # ----------------------------------------------------
278         # i = nilai ISPU
279         kel_pm10 = kel
280         i_pm10 = i
281         print(i_pm10)  # hasilnya i
282         print("^ data test ^")
283
284         # ---------------------------------- temp ----------------------------------
285         temp = data_in[15] * 256 + data_in[16]
286         temp = temp / 100 - 40  # celcius
287         TA = temp  # celcius
288         print(TA)
```

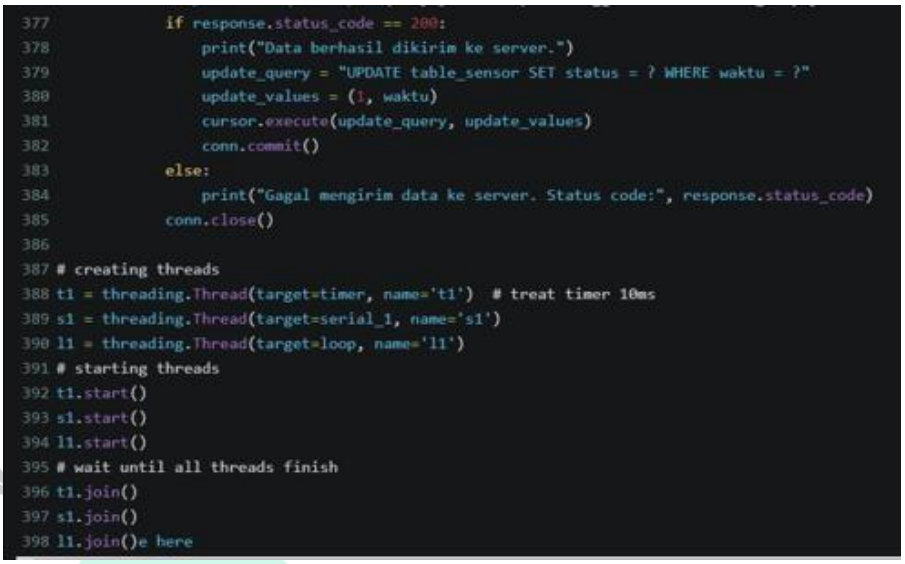
**Gambar 3.21 Tampilan Baris Kode Program**

```
            # ---------------------------------- humidity ----------------------------------
            humi = data_in[17] * 256 + data_in[18]
            humi = humi / 100  # %RH
            print(humi)
            # ------------------------------------ atm -------------------------------------
            atm = data_in[19] * 256 + data_in[20]
            atm = atm / 10  # HPa
            PA = (atm * 100) / 101325  # satuan atm
            print(PA)
            # ---------------------------------- wind speed ----------------------------------
            wind_sp = data_in[21] * 256 + data_in[22]
            wind_sp = wind_sp / 100  # m/s
            print(wind_sp)
            # ---------------------------------- wind direction ----------------------------------
            wind_dir = data_in[23] * 256 + data_in[24]
            wind_dir = wind_dir / 10  # drajad
            print(wind_dir)
            # ------------------------------------ rainfall ------------------------------------
            rain = data_in[25] * 256 + data_in[26]
            rain = rain / 10  # mm
            print(rain)
            # ---------------------------------- solar radiation ----------------------------------
            solar = data_in[27] * 256 + data_in[28]  # W/m2
            print(solar)
            # ---------------------------------- ilumination ----------------------------------
            ilumination = data_in[29] * 256 + data_in[30]
            ilumination = ilumination / 100  # Klux
            print(ilumination)
            # ---------------------------------- uv index ----------------------------------
            uv_index = data_in[31] * 256 + data_in[32]
            print(uv_index)
            # ------------------------------------ co2 -------------------------------------
            co2_1 = data_in[33] * 256 + data_in[34]
            print(co2_1)
            # ---------------------------------- perhitungan c2 ----------------------------------
            so2_2 = (molekul_so2 / 22.414) * (PA / 1) * (273.15 / (273.15 + TA)) * so2_1
            no2_2 = (molekul_no2 / 22.414) * (PA / 1) * (273.15 / (273.15 + TA)) * no2_1
            co_2 = (molekul_co / 22.414) * (PA / 1) * (273.15 / (273.15 + TA)) * co_1
            o3_2 = (molekul_o3 / 22.414) * (PA / 1) * (273.15 / (273.15 + TA)) * o3_1
            # ---------------------------------- akhir data ----------------------------------
            data_in = [0]
            waktu = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            print(waktu)
            client_ip = '192.168.203.192'
            perangkat_id = 1
            # Data untuk di-insert
            data = (
            waktu, client_ip, perangkat_id, so2_1, so2_2, so2, i_so2, kel_so2, no2_1, no2_2, no2,
    i_no2, kel_no2, co_1,
            co_2, co, i_co, kel_co, o3_1, o3_2, o3, i_o3, kel_o3, pm25, i_pm25, kel_pm25, pm10, i
```

**Gambar 3.22 Tampilan Baris Kode Program**

```
            data = (
            waktu, client_ip, perangkat_id, so2_1, so2_2, so2, i_so2, kel_so2, no2_1, no2_2, no2,
i_no2, kel_no2, co_1,
            co_2, co, i_co, kel_co, o3_1, o3_2, o3, i_o3, kel_o3, pm25, i_pm25, kel_pm25, pm10, i_pm10,
kel_pm10, TA,
            humi, PA, wind_sp, wind_dir, rain, solar, ilumination, uv_index, co2_1)
            # Query untuk mengirim data ke database dengan f-string
            sql = f'''INSERT INTO table_sensor (waktu, ip_address, perangkat_id, c1_so2, c2_so2,
c3_so2, i_so2, kel_so2, c1_no2, c2_no2, c3_no2, i_no2, kel_no2, c1_co, c2_co, c3_co, i_co, kel_co,
c1_o3, c2_o3, c3_o3, i_o3, kel_o3, pm25, i_pm25, kel_pm25, pm10, i_pm10, kel_pm10, temp, humi, atm,
wind_sp, wind_dir, rain, solar, ilumination, uv_index, co2_1) VALUES {data}'''
            # Execute the query
            cursor.execute(sql)
            conn.commit()
            print("Data berhasil ditambah ke dalam Database")
            data = {
                "waktu": waktu,
                "ip_address": client_ip,
                "perangkat_id": perangkat_id,
                "pm_25": i_pm25,
                "pm_10": i_pm10,
                "c1_co": co_1,
                "c2_co": co_2,
                "c3_co": i_co,
                "c1_so2": so2_1,
                "c2_so2": so2_2,
                "c3_so2": i_so2,
                "c1_no2": no2_1,
                "c2_no2": no2_2,
                "c3_no2": i_no2,
                "c1_o3": o3_1,
                "c2_o3": o3_2,
                "c3_o3": i_o3,
                "temp": TA,
                "rh": humi,
                "athmospheric_pressure": atm,
                "wind_speed": wind_sp,
                "wind_direction": wind_dir,
                "rainfall": rain,
                "sunshine_radiation": solar,
                "ultraviolet_ray": uv_index
            }
            # Menggabungkan nilai-nilai ke dalam JSON
            json_data = json.dumps(data)
            # Melakukan request POST untuk mengirim data ke endpoint '/data' di aplikasi Flask
            url = 'http://192.168.5.111:5400/api/log_data_live/create'
            response = requests.post(url, json=data)  # Menggunakan data sebagai payload
            if response.status_code == 200:
                print("Data berhasil dikirim ke server.")
                update_query = "UPDATE table_sensor SET status = 2 WHERE waktu = ?"
```

**Gambar 3.23 Tampilan Baris Kode Program**



```
            if response.status_code == 200:
                print("Data berhasil dikirim ke server.")
                update_query = "UPDATE table_sensor SET status = ? WHERE waktu = ?"
                update_values = (1, waktu)
                cursor.execute(update_query, update_values)
                conn.commit()
            else:
                print("Gagal mengirim data ke server. Status code:", response.status_code)
            conn.close()

# creating threads
t1 = threading.Thread(target=timer, name='t1')  # treat timer 10ms
s1 = threading.Thread(target=serial_1, name='s1')
l1 = threading.Thread(target=loop, name='l1')
# starting threads
t1.start()
s1.start()
l1.start()
# wait until all threads finish
t1.join()
s1.join()
l1.join()e here
```

**Gambar 3.24 Tampilan Baris Kode Program**

Selanjutnya masuk kedalam bagian *loop*, yang mana pada bagian inilah otak dari pemrograman ini. Disinilah terjadi pengiriman dan pengolahan data sensor. Pertama prosesnya adalah mengirimkan data permintaan kepada sensor. Pendeklarasian variabel *array* kosong yang akan diisikan sesuai dengan format yang dimengerti sensor. Data yang dikirim berupa *hexadecimal*, formatnya berupa 0xff (alamat sistem), 0x03 (kode fungsi), 0x00 0x03(awalan nomer data yang diminta), dan 0x00 0x10(akhir data yang diminta). Setelah ini tipe data diubah menjadi *bytes* dan ditambahkan CRC sebagai *bit check* permintaan sensor. Setelah dikirim maka sensor akan merespon permintaan dengan mengirimkan data hasil tangkapan dari fungsi serial_1.

Data yang diterima kemudian diolah masing masing perelemen/zat. Tiap elemen perlu dilakukan pengolahan untuk mendapatkan data yang sesuai. Elemen gas ($CO$, $SO2$, $NO2$, dan $O3$) yang ditangkap dari sensor perlu dilakukan pengolahan dari tahap C1 hingga C3, sedangkan elemen partikel padat ($PM2.5$ dan $PM10$) hanya perlu pengolahan C1. Secara sederhana proses C1 merupakan proses pengolahan data yang merupakan output dari sensor menjadi angka desimal yang memiliki satuan. Selanjutnya untuk tahapan C2

merupakan perubahan satuan elemen menjadi satuan yang dipakai dalam aturan baku mutu udara. C3 merupakan tahapan koreksi dengan menggunakan beberapa konstanta tetapan sesuai dengan aturan baku mutu.

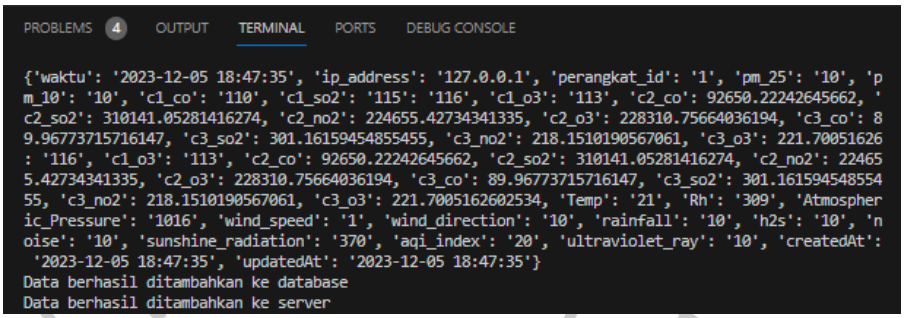
Setelah data semua berhasil di olah, ada satu lagi proses pengolahan data yaitu pengelompokan golongan ISPU. Tiap elemen yang masuk dalam kategori dalam aturan indeks standar pencemar udara akan digolongkan untuk mengetahui tingkat keberbahayaannya. Beberapa konstanta dalam perhitungan rumus ISPU tiap elemen banyak perbedaan, seperti batas untuk tiap kelompoknya. Proses perhitungan ISPU ini dibantu dengan fungsi perhitungan_ispu. Dan terlihat ada beberapa elemen tambahan sebagai pendukung dalam sumber informasi dan sebagai bahan perhitungan yang sama seperti elemen partikel padat hanya perlu proses C1.

Selanjutnya proses pengiriman data kedalam *database*. Data akan disusun dalam variabel data, kemudian akan dideklarasikanquery pengiriman data sensor kedalam *database* yang memiliki tujuanalamat tabel yang bernama tabel_sensor. Setalah itu program akan melakukan koneksi ke *database* dan melakukan pengiriman *query* yang sudah dideklarasikan tadi. Jika berhasil maka akan terlihat respon dan juga data akan muncul pada *database*.

Proses untuk pengiriman data ke *endpoint*, data yang sudah disusun harus dirubah formatnya menjadi format JSON agar sesuai dengan format aturannya. Setelahnya untuk melakukan pengiriman perlunya di deklarasikan alamat tujuannya, dan juga perlu pengecekan respon. Apabila respon setelah pengiriman sama dengan 200 maka proses pengiriman ke server/*endpoint* berhasil.

Yang terakhir merupakan bagian ini adalah inti program. Terdapat tiga fungsi untuk setiap *thread*: *timer*, serial_1, dan *loop*. Masing-masing berisi logika eksekusi untuk tugas tertentu. Tiga *thread* tersebut diinisialisasi dan dijalankan.

(5) Output hasil program




**Gambar 3.25 Tampilan Hasil Program**

Hasil yang ditampilkan dalam terminal yang dapat dilihat mengacu pada Gambar 3.25 merupakan hasil pengaktifan kode program. Data akan terus diterima sesuai dengan interval waktu yang telah di atur dalam kode program. Selanjutnya data ini dikirim ke dalam *database* dan juga ke dalam server. Hampir semua elemen telah di tangkap dari sensor seperti $CO$ (Karbon Monoksida), $SO2$ (Sulfur Dioksida), $NO2$ (Nitrogen Dioksida), $O3$ (Ozon), $PM2.5$ (Particulate Matter), $PM10$ (Particulate Matter), *Atmospheric Temperature, Atmospheric Humidity, Atmospheric Tempress, Wind Speed, Wind Direction, Rainfall, Solar Radiation, Illumination, Ultraviolet Ray*, $CO2$ (Karbon Dioksida), dan *Electronic Compass.* Pada kode program juga memproses data dari konversi, koreksi, dan komparasi yang di tampung dalam variabel dan juga dikirimkan bersamaan dengan data yang telah disebutkan sebelumnya.

(6) Bentuk *database* hasil

**Tabel 3.3 Struktur *Database* tabel_sensor**

| No. | Nama | Jenis | No. | Nama | Jenis |
|---|---|---|---|---|---|
| 1. | ID | Int(11) | 17. | C2_o3 | Float |
| 2. | Waktu | Datetime | 18. | C3_o3 | Float |
| 3. | Ip_address | Varchar | 19. | Temp | Float |
| 4. | Perangkat_id | Int(11) | 20. | Rh | Float |
| 5. | Pm_25 | Float | 21. | Atmospheric_pressure | Float |
| 6. | Pm_10 | Float | 22. | Wind_speed | Float |
| 7. | C1_co | Float | 23. | Wind_direction | Float |
| 8. | C2_co | Float | 24. | Rainfall | Float |
| 9. | C3_co | Float | 25. | H2s | Float |
| 10. | C1_so2 | Float | 26. | Noise | Float |
| 11. | C2_so2 | Float | 27. | Sunshine_radiation | Float |
| 12. | C3_so2 | Float | 28. | Aqi_index | Float |
| 13. | C1_no2 | Float | 29. | Ultraviolet_ray | Float |
| 14. | C2_no2 | Float | 30. | CreatedAt | Datetime |
| 15. | C3_no2 | Float | 31. | DeletedAt | Datetime |
| 16. | C1_o3 | Float | | | |

Tabel 3.3 merupakan struktur tabel_sensor dimana terdapat ‘id’ berfungsi sebagai identitas unik atau *primary key* dalam kolom tersebut, dan masing-masing kolom memiliki identifikasi yang berbeda dengan tipe data bertipe *integer* dan panjang karakter sebanyak 11, terdapat juga ‘ip_address’ yang dimana berfungsi untuk menyimpan dan mengelola informasi terkait alamat IP yang di gunakan dalam suatu sistem dan jaringan dengan tipe data bertipe *varchar* dan panjang karakter sebanyak 255. Praktikan membuat output dari kode program berdasarkan tabel_sensor ini. Praktikan mengirim data ke

*database* berdasarkan parameter hasil pengukuran yaitu ‘pm_25, pm_10, c1_co, c2_co, c2_co, c1_so2, c2_so2, c3_so2, c1_no2, c2_no2, c3_no2, c1_o3, c2_o3, c3_o3, temp, rh, atmospheric_pressure, wind_speed, wind_direction, rainfall, h2s, noise, sunshine_radiation, aqi_index, ultraviolet_ray’ dengan tipe data bertipe float.

| id | waktu | ip_address | perangkat_id | pm_25 | pm_10 | c1_co | c2_co | c3_co | c1_so2 | c2_so2 | c3_so2 | c1_no2 | c2_no2 | c3_no2 | c1_o3 | c2_o3 | c3_o3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2023-06-25 04:49:00 | 192.168.1.1 | 1 | 10 | 10 | 110 | 115 | 116 | 115 | 116 | 180.65 | 150 | 151 | 153 | 140 | 141 | 142 |
| 7 | 2023-09-08 02:12:20 | 192.168.1.3 | 3 | 10 | 15 | 2 | 4 | 6 | 5 | 7 | 9 | 7 | 9 | 11 | 14 | 17 | 21 |
| 8 | 2023-06-24 04:49:00 | 192.168.1.1 | 1 | 12 | 12 | 111 | 112 | 113 | 114 | 122 | 124 | 140 | 142 | 146 | 145 | 147 | 150 |
| 9 | 2023-06-26 04:49:00 | 192.168.1.1 | 1 | 2 | 2 | 40 | 45 | 50 | 24 | 25 | 26 | 10 | 12 | 14 | 14 | 16 | 18 |
| 10 | 2023-05-26 05:49:00 | 192.168.1.1 | 1 | 15 | 30 | 12 | 14 | 16 | 20 | 24 | 26 | 21 | 24 | 25 | 23 | 25 | 27 |

**Gambar 3.26 Tampilan Isi Tabel tabel_sensor**

| temp | rh | atmospheric_pressure | wind_speed | wind_direction | rainfall | h2s | noise | sunshine_radiation | aqi_index | ultraviolet_ray | createdAt | updatedAt |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 25 | 56.35 | 1016 | 2.93 | 309.2 | 1 | 10 | 10 | 370 | 20 | 10 | 2023-07-03 04:52:17 | 2023-07-03 04:57:14 |
| 25 | 20 | 24 | 21 | 10 | 1 | 21 | 4 | 21 | 12 | 31 | 2023-09-07 21:12:19 | 2023-09-07 21:12:19 |
| 25 | 30 | 1 | 21 | 10 | 1 | 21 | 4 | 21 | 12 | 31 | 2023-09-11 11:54:12 | 2023-09-11 11:54:12 |
| 25 | 23 | 20 | 20 | 20 | 20 | 20 | 2 | 20 | 20 | 20 | 2023-09-11 12:00:53 | 2023-09-11 12:00:53 |
| 25 | 20 | 1 | 21 | 23 | 24 | 23 | 12 | 1 | 1 | 24 | 2023-09-11 12:10:05 | 2023-09-11 12:10:05 |

**Gambar 3.27 Tampilan Isi Tabel tabel_sensor**

Gambar 3.26 dan gambar 3.27 merupakan hasil tampilan tabel_sensor yang mana dapat menampilkan data yang telah tersimpan dalam sebuah tabel *database*. Praktikan dapat dengan mudah untuk mengontrol entri data yang ada di dalam tabel, seperti yang terlihat di dalam tabel_sensor terdapat beberapa data yang masuk seperti ‘waktu, ip_address, perangkat_id, pm_25, pm_10, c1_co, c2_co, c2_co, c1_so2, c2_so2, c3_so2, c1_no2, c2_no2, c3_no2, c1_o3, c2_o3, c3_o3, temp, rh, atmospheric_pressure, wind_speed, wind_direction, rainfall, h2s, noise, sunshine_radiation, aqi_index, ultraviolet_ray’. Jika respon dari kode program telah berhasil mengirim data kedalam *database,* maka secara langsung data tersebut bisa terlihat pada tabel_sensor ini.

### 3.3 Kendala yang Dihadapi

Dalam pelaksanaan program kerja profesi yang dilakukan di PT. MARKTEL ini praktikan mengalami beberapa kendala selama proses pengerjaan.

(1) Keterbatasan referensi sensor HCD6818

Sulitnya mendapatkan referensi sensor menjadi kendala awal. Proses penghubungan mikrokontroller dengan sensor memakan waktu yang cukup lama. Proses pengiriman data *input* untuk meminta data dari sensor cukup sulit karena praktikan perlu mengethui proses kerja protokol Modbus RTU yang digunakan oleh sensor. Proses perubahan data *output* sensor yang memiliki format CRC16 menjadi angka desimal menimbulkan kendala teknis.

(2) Pemahaman dalam pengolahan data

Diperlukannya pemahaman terhadap peraturan yang berkaitan dengan pengolahan data. Praktikan perlu melakukan riset dan mendapatkan pemahaman mengenai dasar dari perhitungan baku mutu udara. Praktikan perlu mengerti cara perhitungan konversi dari satuan *Parts Per Billion* ($ppb$) ke satuan massa jenis $\mu g/m^3$. Perlunya mengetahui tahapan dalam pengolahan data diawali dari konversi, koreksi, hingga komparasi. Selain itu juga praktikan perlu mengerti untuk perhitungan ISPU yang sesuai dengan peraturan menteri Nomor P.14/MENLHK/SETJEN/KUM.1/7/2020 Tentang Indeks Standar Pencemar Udara (ISPU).

(3) Perancangan alat

Kekurang pahaman praktikan mengenai komponen perangkat keras menjadi tantangan dalam perancangan blok diagram. Pemahaman hubungan antar komponen dan juga fungsi tiap komponen sangat diperlukan. Dasar elektronika sangat diperlukan ketika ingin merancang suatu perangkat. Diperlukannya ketelelitian dalam perakitan alat menjadi kunci agar terhindar dari kerusakan alat.

*(4)* Integrasi *database*

Proses pengiriman data perlu kesesuaian data yang dikirim dengan variabel tampungan data dalam *database* menjadi kesulitan praktikan. Banyaknya data yang diterima dari sensor tiap interval menyulitkan dan

membuat program berjalan melambat. Masalah teknis yang mungkin terjadi mengakibatkan data yang dikirim tidak masuk dan bahkan tidak tercatat dalam *database*

.

### 3.4 Cara Mengatasi Kendala

(1) Strategi Pencarian Referensi

Dalam mengatasi keterbatasan referensi sensor, praktikan aktif mencari sumber informasi yang dapat mendukung pengembangan algoritma. Mulai dari pengaksesan *datasheet* sensor untuk memahami spesifikasi dan ketentuan penggunaannya. Praktikan mencari sumber bacaan untuk memahami secara mendalam perhitungan konversi CRC16 dan mengkonsultasikan dengan mentor. Praktikan mencoba berbagai pendekatan dan belajar dari setiap percobaan algoritma sederhana untuk memastikan integrasi yang optimal.

(2) Pendekatan mengolah data

Dalam masalah ini praktikan melakukan pemahaman rumus dan melakukan uji coba perhitungan manual dengan data *dummy*. Praktikan juga meminta pemahaman dan arahan bahan pembelajaran dari mentor untuk memahami aturan yang berlaku dalam projek. Mencari dari jurnal yang berkaitan tentang perhitungan baku mutu udara ambien. Membuat algoritma pemrograman yang sederhana untuk proses perhitungan sebelum digabungkan dengan program utama.

(3) Kerjasama tim dan mentor

Projek ini berjalan bersama dengan tim yang telah dibentuk dan tiap anggota memiliki perannya masing-masing. Dalam dunia pekerjaan perlunya interaksi antar anggota agar dapat memecahkan masalah secara bersama dan komunikasi menjadi kunci untuk keberhasilan projek. Pembentukkan *timeline* pekerjaan akan mempermudah alur sehingga tidak ada pekerjaan yang tertinggal, sehingga pekerjaan akan rapih. Masukkan dari mentor dan rekan tim pada program kerja profesi yang telah saya laksanakan banyak membantu saya dalam proses pengerjaan.