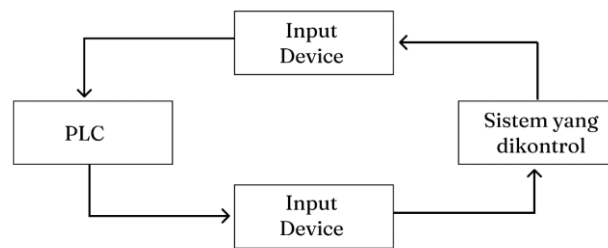


BAB II

TINJAUAN PUSTAKA

2.1 Programmable Logic Control (PLC)

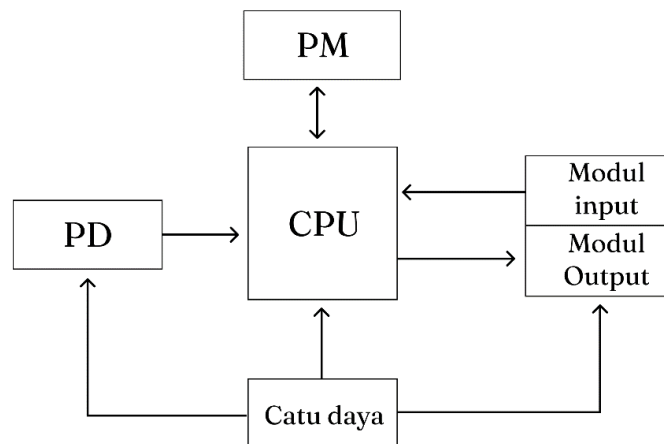
Programmable Logic Control (PLC) adalah perangkat elektronik yang bekerja secara digital, memiliki memori yang dapat diprogram, menyimpan perintah-perintah untuk melakukan tugas khusus seperti *logic*, *timing*, *counting*, aritmatika dan *sequencing* untuk mengontrol mesin atau proses melalui analog atau *digital input/output modules* (Petruzella, 2017). Data-data yang masuk ke PLC disimpan dalam memori dalam bentuk digital, oleh karena itu, nilai input yang masih berupa sinyal analog diubah oleh modul *input A/D (analog to digital input module)* menjadi sinyal digital. PLC dibuat dengan tujuan untuk mengganti pengendali konvensional seperti *relay*. Kelebihan PLC dibandingkan dengan pengendali konvensional diantaranya adalah *programmable* dan *reprogrammable*, tahan terhadap kondisi lingkungan industri, dan dimensi yang lebih kecil. PLC adalah perangkat berbasis mikroprosesor yang dirancang khusus untuk menggantikan logika dan aplikasi lain, dan didesain untuk berbagi aplikasi yang berhubungan dengan sensor-sensor industri (Setioko et al., 2019). Prinsip kerja PLC dapat dilihat pada Gambar 2.1.



Gambar 2. 1 Diagram blok cara kerja PLC

Sumber : (Amaliawati et al., 2020)

PLC dapat menerima data berupa sinyal analog dan sinyal digital dari perangkat *input*. Sinyal *input* digital dapat berasal dari saklar-saklar, *push button*, sensor dan perangkat lain yang dapat mengirimkan sinyal *High* (1) atau *Low* (0). Sinyal analog dari perangkat input dapat berasal dari potensiometer, putaran motor, dan peralatan sejenisnya. Sinyal analog yang berasal dari perangkat *input* diubah menjadi sinyal digital. Kemudian sinyal digital yang masuk tersebut diolah oleh *Central Processing Unit* (CPU) sesuai dengan program yang telah dimasukkan. Selanjutnya CPU mengirimkan perintah-perintah logika ke perangkat *output* atau peralatan yang dikontrol oleh PLC tersebut. Bagian PLC pada dasarnya terdiri dari *Central Processing Unit*, *Programming Memory* (PM), *Programming Device* (PD), modul *Input Output* dan unit catu daya. Bagian-bagian tersebut dijelaskan dengan diagram blok pada Gambar 2.2.



Gambar 2. 2 Diagram blok bagian-bagian PLC

Sumber : (Amaliawati et al., 2020)

Fungsi dari bagian-bagian dasar PLC adalah sebagai berikut.

1. *Central Processing Unit (CPU)*

CPU bekerja untuk mengambil intruksi dari memori, mendekodenya dan mengeksekusi intruksi tersebut. Bagian ini menjalankan fungsi sistematis dan logika serta mendeteksi sinyal diluar dari CPU.

2. *Programming Memory (PM)*

PM bertugas menyimpan intruksi, program dan data. PLC dapat diprogram dengan menggunakan laptop atau komputer menggunakan bahasa pemrograman tangga atau *Ladder* yang umum digunakan untuk PLC.

3. *Programming Device (PD)*

PD atau disebut juga *Programming Device Terminal (PDT)* adalah suatu perangkat yang digunakan untuk mengetik, memasukan, memodifikasi, dan memantau program yang ada pada memori PLC. contoh dari PDT adalah Laptop dan *Personal Computer (PC)*.

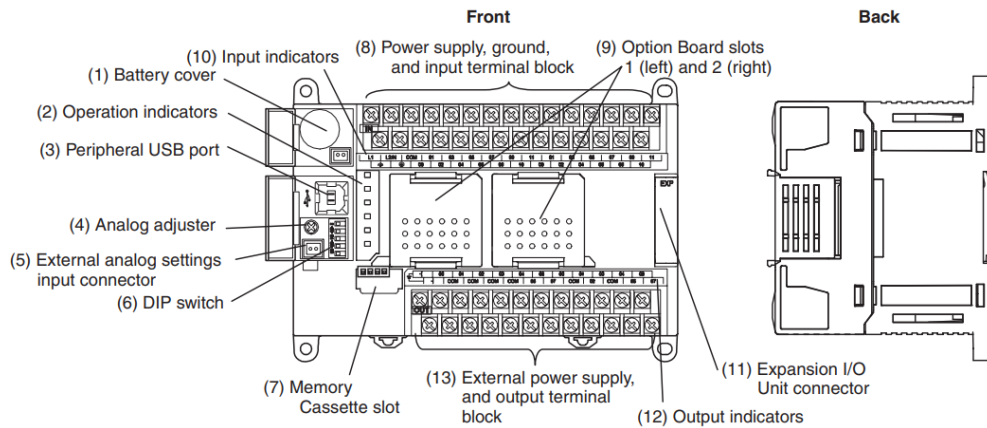
4. Modul *Input/Output*

Modul *Input/Output* (I/O) adalah perangkat elektronika yang berfungsi sebagai penghubung antara PLC dengan perangkat I/O luar. Contoh perangkat tersebut adalah sensor sebagai perangkat *input* dan relay sebagai perangkat *output*.

2.1.1 PLC OMRON SYSMAC CP1L

PLC OMRON SYSMAC CP1L merupakan salah satu produk PLC dari perusahaan OMRON. PLC CP1L merupakan PLC tipe paket yang tersedia dengan 10, 14, 20, 30, 40, atau 60 buah *input/output*. PLC ini dirancang dengan kecepatan proses yang tinggi, sehingga penerapan dari PLC ini dapat digunakan dalam sistem yang membutuhkan respon cepat (OMRON CORPORATION, 2014).

Selain kecepatan prosesnya, PLC CP1L juga memiliki kemudahan dalam penginstalan, pengembangan, dan pemasangan sistem. PLC ini dilengkapi dengan *interface* yang mudah untuk digunakan. Hal tersebut memudahkan pengguna dalam mempelajari dan mengoperasikannya. Untuk konektivitasnya, PLC ini dilengkapi *port* komunikasi serial dan *Ethernet*. Bagian-bagian dari PLC CP1L dijelaskan pada Gambar 2.3



Gambar 2. 3 Bagian-bagian PLC Omron CP1L

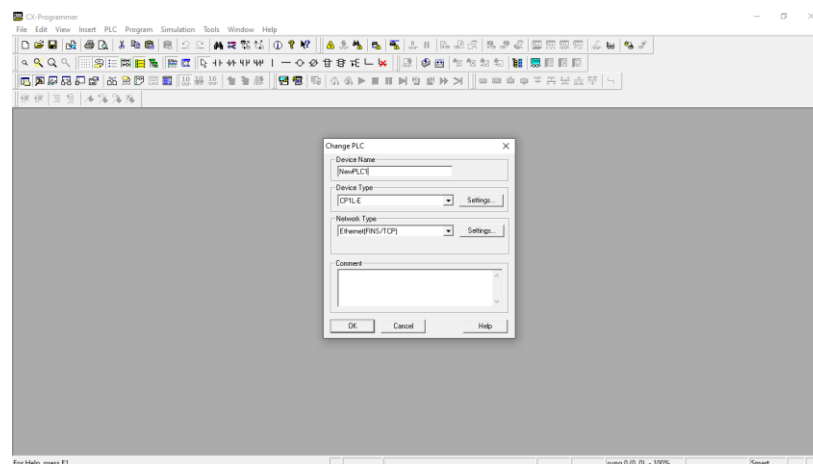
Sumber : (OMRON CORPORATION, 2014)

1. *Battery cover*, digunakan untuk tempat memasang baterai jika PLC membutuhkan *internal clock* dan isi RAM ketika *supply OFF*.
2. *Operation indicators*, mengidentifikasi status operasi dari PLC.
3. *Peripheral USB port*, digunakan untuk menghubungkan USB jika ingin mengomunikasikan dengan laptop.
4. *Analog adjuster*, digunakan untuk konfigurasi dan kalibrasi sinyal analog yang masuk dan keluar dari PLC.
5. *External analog settings input connector*, digunakan untuk menghubungkan sinyal analog eksternal ke PLC.
6. *DIP switch*, untuk mengatur konfigurasi dasar pada PLC, seperti pengaturan *Mode RUN* atau *PROGRAM*.
7. *Memory Cassette slot*, digunakan untuk memasang modul memori eksternal.
8. *Power supply, ground, and input terminal block*, digunakan untuk memasang *power supply*, *ground*, dan masukan.

9. *Option board slot*, digunakan untuk memasang *port* komunikasi tambahan jika diperlukan.
10. *Input indicator*, sebagai indikator dari terminal *input* yang menyala.
11. *Expansion I/O unit connector*, digunakan untuk menghubungkan modul *I/O* tambahan jika diperlukan.
12. *Output indicators*, indikator dari terminal *output* yang menyala.
13. *External power supply, and output terminal block*, digunakan untuk memasang keluaran dan catu daya eksternal.

2.1.2 CX-Programmer

Pemrograman PLC adalah penulisan perintah-perintah yang harus dilakukan pada PLC. Pemrograman PLC CP1L dapat menggunakan bahasa pemrograman *ladder diagram* (LD) atau *Instruction List* (IL) (Fauzi, 2016). *CX-Programmer* merupakan *software* yang dikeluarkan OMRON untuk memprogram PLC yang dikeluarkan dari perusahaan tersebut. Tampilan awal dari *software* ini ditampilkan pada Gambar 2.4

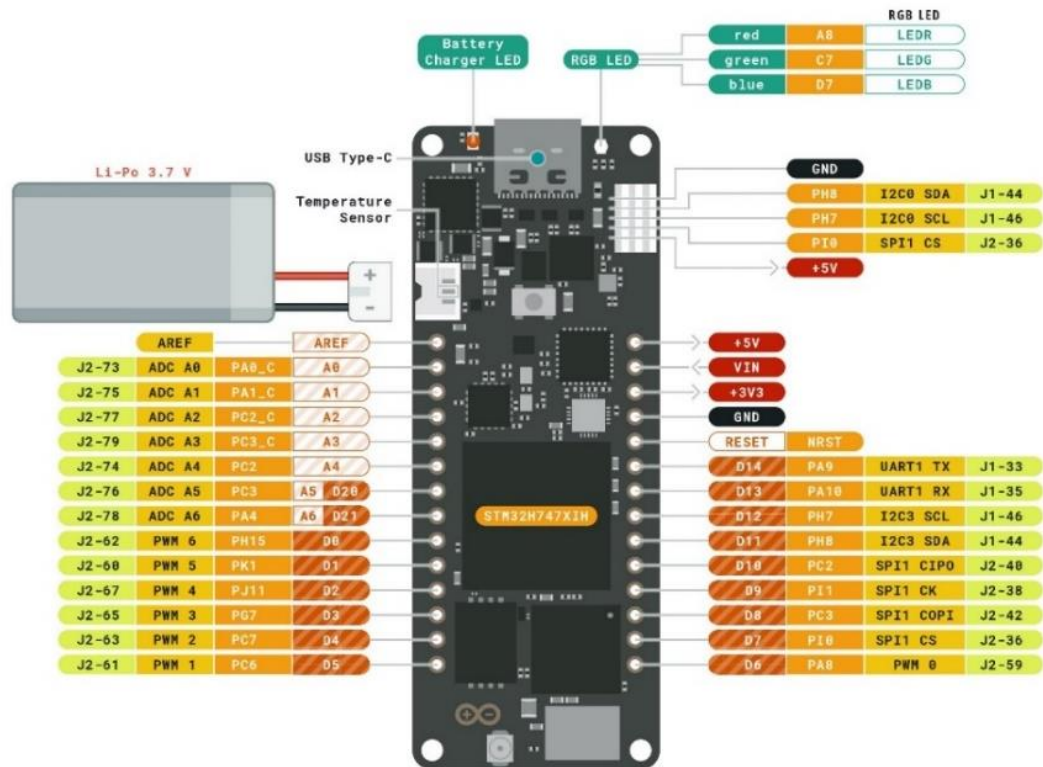


Gambar 2. 4 Tampilan awal CX-Programmer

PLC Omron dapat diprogram dengan menggunakan aplikasi CX-Programmer dengan cara menyusun *ladder* diagram. *Ladder* diagram merupakan kumpulan instruksi dan alamat yang akan digunakan di PLC nantinya. Pengalamatan pada *ladder* diagram sangat penting karena alamat yang diberikan pada setiap kontakannya dapat mempengaruhi urutan program sehingga setiap alamat *input output* yang dimiliki sebuah PLC harus diketahui dengan baik, karena setiap PLC memiliki alamat *input output* yang berbeda. CX-Programmer juga bisa digunakan untuk memonitoring PLC secara *online* untuk memonitoring kinerja sistem dengan perantara kabel serial atau *Ethernet*.

2.2 **Arduino Portenta H7**

Arduino Portenta H7 adalah mikrokontroler yang memakai *dual core* STM32H747 sebagai prosesor utama. Prosesor ini memiliki dua buah inti yaitu Cortex M7 pada 480 MHz dan Cortex M4 pada 240 MHz. Sistem *dual core* menjadi kelebihan yang dimiliki Arduino Portenta H7. Hal ini menjadikan Arduino Portenta H7 dapat menjalankan tugas secara parallel, hal ini yang menjadi kelebihan dari Portenta H7 dibanding mikrokontroler Arduino lainnya. (Arduino, 2024). Selain itu, mikrokontroler ini dirancang untuk diimplementasikan pada sistem *High-end industrial*, sistem PLC, kebutuhan laboratorium, dan kontroler robotic (Gimenez et al., 2022). *Board* Arduino Portenta dapat dilihat pada Gambar 2.5.



Gambar 2. 5 Arduino Portenta H7 board

Sumber : (Arduino, 2024)

Bagian utama pada Portenta H7 adalah sebagai berikut :

1. Prosesor *Dual Core* STM32H747

Arduino Portenta H7 memiliki prosesor utama dengan Type *Dual Core* STM32H747. Dua inti dari prosesor tersebut adalah Cortex M7 yang bekerja dengan frekuensi 480 MHz dan Cortex M4 pada 240 MHz. Portenta H7 memungkinkan pemanggilan fungsi pada prosesor lain dengan fitur multitaskingnya yang menggunakan mekanisme *Remote Procedure Call*. Dengan mekanisme ini, kedua inti dari prosesor dapat saling berkomunikasi satu sama lain.

2. On-Board Connectivity

Untuk konektivitas nirkabel, Portenta H7 memiliki modul Murata 1DX *dual* WiFi dan *Bluetooth*. Modul ini memungkinkan Portenta H7 mengelola konektivitas WiFi dan *Bluetooth* secara simultan. Portenta H7 memiliki *WiFi interface* yang dapat digunakan sebagai *Access Point*, *Station* ataupun *dual mode* simultan AP/STA. Sedangkan untuk *Bluetooth interface*-nya dapat mendukung konektivitas *Bluetooth Classic* dan *Bluetooth Low Energy*.

3. Port

Port berfungsi sebagai penghubung portenta H7 dengan berbagai perangkat lain ke mikrokontroler, misalnya sensor, perangkat mikrokontroler lain, dan lain lain. Spesifikasi dari Arduino Portenta H7 dengan prosesor STM32H747 dijelaskan pada Tabel 2.1.

Tabel 2. 1 Spesifikasi Arduino Portenta H7

Sumber : (Arduino, 2024)

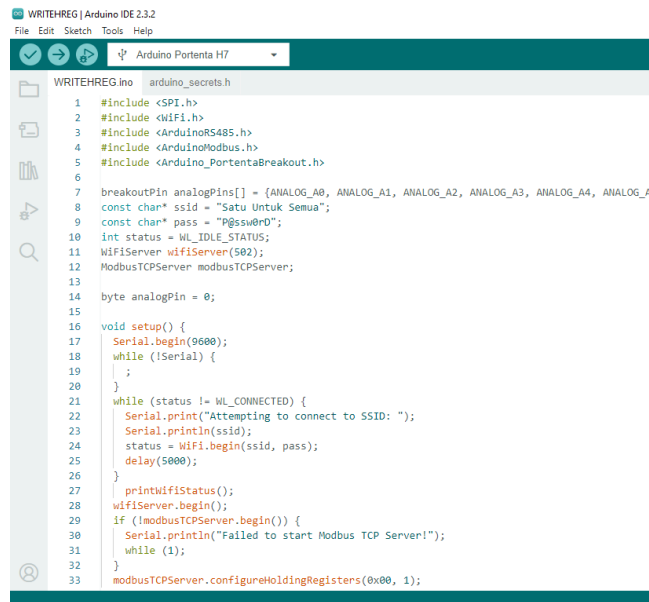
Mikrokontroler	STM32H747 <i>dual core</i> Cortex®-M7+M4 32 bit <i>low power</i> Arm® MCU
Modul Radio	Murata 1DX dual WiFi 802.11b/g/n 65 Mbps dan Bluetooth® (Bluetooth® Low Energy.5 via Cordio stack, Bluetooth® Low Energy 4.2 via Arduino Stack)
Source element	NXP SE0502
Catu daya papan sirkuit (USB/VIN)	5 V
GPU	Chrom-ART graphical hardware Accelerator™
Timers	22x timers and watchdogs
UART	4x ports (2 with flow control)
Ethernet PHY	10/100 Mbps (hanya melalui port ekspansi).
SD CARD	Interface for SD Card connector (hanya melalui port ekspansi).
Temperatur	-40° C to 85° C

High-Density Connectors	Two 80 pin connectors
Camera Interface	8-bit, up to 80MHz
ADC	3x ADCs pada 16 bit maks. Resolution (up to 36 channels, up to 3.6 MSPS)
DAC	2x12-bit DAC (1MHz)
USB-C	Host/Device, Display Port Ou, High/Fullspeed, Power Delivery

Untuk bahasa pemrograman, Arduino Portenta H7 dapat diprogram menggunakan software Arduino IDE. Arduino IDE digunakan untuk menginput program ke mikrokontroler Portenta. Software ini diciptakan dengan antarmuka yang sederhana untuk memudahkan pengguna dalam memprogram mikrokontroler. Bahasa yang digunakan pada software ini adalah bahasa C++ yang disederhanakan dengan banyaknya library yang tersedia.

Arduino IDE merupakan software yang berperan untuk menulis program, meng-*compile* dan mengunggah program tersebut ke dalam memori pada mikrokontroler. Software ini terdiri dari tiga bagian utama, yaitu :

1. Editor program, untuk menulis dan mengedit program dalam bahasa C++.
Listing program pada Arduino IDE disebut dengan “*Sketch*”.
2. *Compiler*, berfungsi sebagai pengubah bahasa yang ditulis pengguna ke dalam kode biner. Hal ini diperlukan karena mikrokontroler hanya mengerti kode biner.
3. *Uploader*, digunakan untuk mengunggah kode yang telah di-*compile* ke dalam memori mikrokontroler. Tampilan dari *Software* Arduino IDE dapat dilihat pada Gambar 2.6.

Gambar 2. 6 Tampilan *software* Arduino IDE

Sumber : Dokumen pribadi

2.3 Protokol Modbus TCP

a. Protokol Modbus

Modbus merupakan protokol komunikasi data berstandar internasional yang dipublikasikan oleh Perusahaan Modicon pada Tahun 1979 dan sering digunakan pada *Programmable logic control* (PLC). Protokol ini berjalan pada bagian *Application Layer* pada model referensi *Open System Interconnection* (Tosin, 2021). Modbus berkerja dengan menghubungkan setiap perangkat melalui komunikasi serial. Terdapat dua kategori perangkat dalam protokol ini, yaitu *Modbus master* dan *Modbus Slave*. *Modbus master* bekerja dengan menerima informasi yang dikirimkan oleh *Modbus slave*. *Master* atau *Slave* dapat berupa PLC, mikrokontroller, atau *Human Machine Interface* (HMI). *Master* memiliki peran untuk memulai komunikasi data dengan meBaca data, menulis data, dan

mengetahui status *slave* dan mengirimkan permintaan data. Sedangkan *slave* hanya merespon jika ada permintaan dari *master*.

Master memiliki peran aktif dengan memulai komunikasi baik dengan membaca data, menulis data, mengetahui status *slave* dan mengirimkan permintaan atau *query* (Gamess et al., 2020). *Query* terdiri dari *function code* dan *data query*. Sedangkan *slave* hanya merespon permintaan dari *master*. Jika *data query* dari *master* diterima oleh *slave* tanpa kesalahan, maka *slave* akan mengirimkan responnya ke *master*. Respon tersebut berupa *function code* dan *data response*. Apabila terjadi kesalahan pada *query* yang diterima oleh *slave*, maka *slave* akan mengirim pesan *error* ke *master* yang terdiri dari *exception function code* dan *exception code* (Amaliawati et al., 2020). *Function code* merupakan data perintah yang harus dilakukan oleh *slave*. *Function code* pada modbus dapat dilihat pada Tabel 2.2 berikut.

Tabel 2. 2 *Function Code* pada protokol Modbus

Sumber : (Tosin, 2021)

Jenis Data	Fungsi			No. Awal register
	Membaca	Menulis 1 data	Menulis banyak data	
Coil	01(01 hex)	05 (05 hex)	15 (0F hex)	00001
Discrete Input	02(02 hex)			10001
Input Register	03(03 hex)			30001
Holding Register	04(04 hex)	06 (06 hex)	16 (10 hex)	40001

Saat terjadi *error* atau kesalahan pada data perintah tidak bisa ditangani oleh *slave*, maka *slave* akan mengirim pesan *error* dalam bentuk *response exception*.

Response exception terdiri dari *exception function code* dan *exception code*. *Exception function code* adalah *function code* dari *master* dengan mengubah *most significant bit* dalam bilangan biner menjadi satu atau *function code* dalam bilangan heksadesimal ditambah dengan 80. Berikut adalah Tabel 2.3 yang menjelaskan *exception function code*.

Tabel 2. 3 *Exception Function Code* protokol Modbus

Sumber : (Tosin, 2021)

Function code in request	Function code in exception response
01 (01 hex) 0000 0001	129 (81 hex) 1000 0001
02 (02 hex) 0000 0010	130 (82 hex) 1000 0010
03 (03 hex) 0000 0011	131 (83 hex) 1000 0011
04 (04 hex) 0000 0100	132 (84 hex) 1000 0100
05 (05 hex) 0000 0101	133 (85 hex) 1000 0101
06 (06 hex) 0000 0110	134 (86 hex) 1000 0110
15 (0F hex) 0000 1111	143 (8F hex) 1000 1111
16 (10 hex) 0001 0000	144 (90 hex) 1001 0000

Exception code adalah kode yang mengindikasikan kesalahan yang terjadi pada saat *slave* memberi respon kesalahan kepada *master*. *Exception code* dapat dilihat pada Tabel 2.4.

Tabel 2. 4 *Exception code* protokol Modbus

Sumber : (Tosin, 2021)

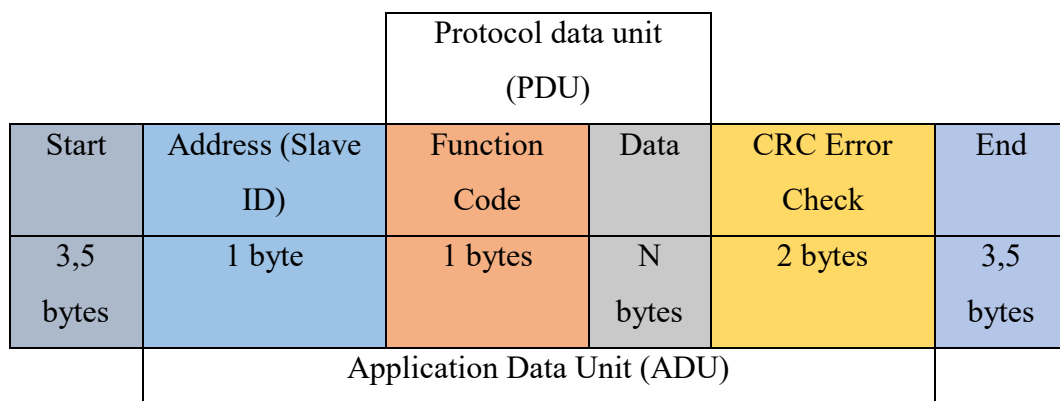
Exception code	Nama	Keterangan
01 (01 hex)	Illegal function	Kode ini menunjukkan bahwa slave dalam kondisi tidak tepat untuk memproses permintaan dari master, misalnya karena tidak dikonfigurasi

Exception code	Nama	Keterangan
		dan diminta untuk mengembalikan nilai register
02 (02 hex)	Illegal data address	Alamat register salah
03 (03 hex)	Illegal data value	Terdapat nilai data yang tidak bisa dibaca slave
04 (04 hex)	Slave device failure	Slave tidak dapat menjalankan perintah
05 (05 hex)	Acknowledgement	Pemberitahuan ke master bahwa perintah akan memakan waktu lama dan menyebabkan timeout
06 (06 hex)	Slave device busy	Slave sedang memproses perintah long duration program.
07 (07 hex)	Negative acknowledgement	Slave tidak dapat menjalankan fungsi program yang diterima. Kode ini dikembalikan untuk permintaan pemrogram yang gagal menggunakan function code 13 atau 14 desimal. Master harus meminta informasi diagnostik.
08 (08 hex)	Memory parity error	Slave berusaha membaca extended memory atau merekam file, namun terdeteksi kesalahan parity memory
10 (0A hex)	Gateway path unavailable	Menunjukkan bahwa gateway tidak bisa mengalokasikan jalur komunikasi internal dari port input ke port output untuk memproses permintaan.
11 (0B hex)	Gateway target device failed to respond	Tidak ada respon yang diperoleh dari perangkat target, biasanya karena perangkat tidak ada di jaringan.

Setiap *slave* pada jaringan *modbus* memiliki alamat yang unik, hanya alamat yang dituju oleh *master* yang dapat menerima perintah. Jaringan *modbus* dapat menghubungkan sampai dengan 247 *slave* dengan alamat yang berbeda pada satu

master. Setiap perintah memiliki mekanisme untuk memastikan data yang dikirim tidak terjadi kesalahan. Perintah pada *modbus* berupa perintah untuk mengubah nilai *register*, membaca nilai *register*, maupun membaca *port input output*.

Varian *modbus* yang umum digunakan adalah *modbus remote terminal unit* (RTU). Modbus RTU menggunakan komunikasi serial dengan bentuk yang ringkas. Format RTU menggunakan mekanisme *Cyclic Redundancy Check* (CRC) untuk memastikan data yang dikirim dan diterima lengkap dan benar. Dalam setiap 8-byte data yang dikirim berisi karakter 4 *bit* heksadesimal. Data dapat dikirim lebih banyak dengan *baud rate* yang sama. Setiap data yang dikirim bersifat *continue* dan memiliki *delay* pada awal dan akhir pesan antara 1,5 sampai 3,5 karakter dari *baudrate* yang dipakai (Tamboli et al., 2015). Format *frame data* pada protokol modbus RTU direpresentasikan pada Gambar 2.7.



Gambar 2. 7 Format *frame data* protokol Modbus RTU

Sumber : (Tosin, 2021)

Fungsi dari setiap *block frame* dari modbus RTU adalah sebagai berikut.

1. *Start*, berfungsi sebagai penanda awal *frame*.
2. *Address*, alamat dari *slave*. Alamat 0 digunakan untuk *broadcast*.
3. *Function code*, berisi perintah yang harus dilaksanakan *slave*.

4. Data, berisi alamat *register*, jumlah data, dan data yang akan ditulis.
5. *CRC Error Check*, untuk mendeteksi kesalahan data.
6. *End*, sebagai penanda akhir dari *frame* data.

Data yang diterima *slave* disimpan pada *register* yang mempunyai alamat berbeda berdasar tipe data dan fungsi yang diterima. Data pada modbus terdiri dari data *discrete* dan analog. Data *discrete* memiliki nilai satu bit 0 dan 1. Sedangkan data analog memiliki nilai 16-bit. Data dapat ditampilkan dalam bilangan biner maupun heksadesimal. Setiap data dapat diakses secara berbeda, contohnya *read-write* atau *read only*. Fungsi dari setiap *register* adalah sebagai berikut:

1. *Discrete output coil*, data ini digunakan untuk mengaktifkan *coil relay*, *register* ini berfungsi untuk mengirimkan data digital 0 atau 1.
2. *Discrete input contact*, disebut juga dengan *input relay*, *register* ini berfungsi untuk membaca data digital.
3. *Analog input registers*, digunakan untuk menyimpan data analog dari 0 sampai dengan 65535.
4. *Analog output holding registers*, digunakan untuk menyimpan nilai analog dengan *range* 0 sampai dengan 65535. Pemetaan alamat register dapat dilihat pada Tabel 2.5

Tabel 2. 5 Pemetaan alamat *register*

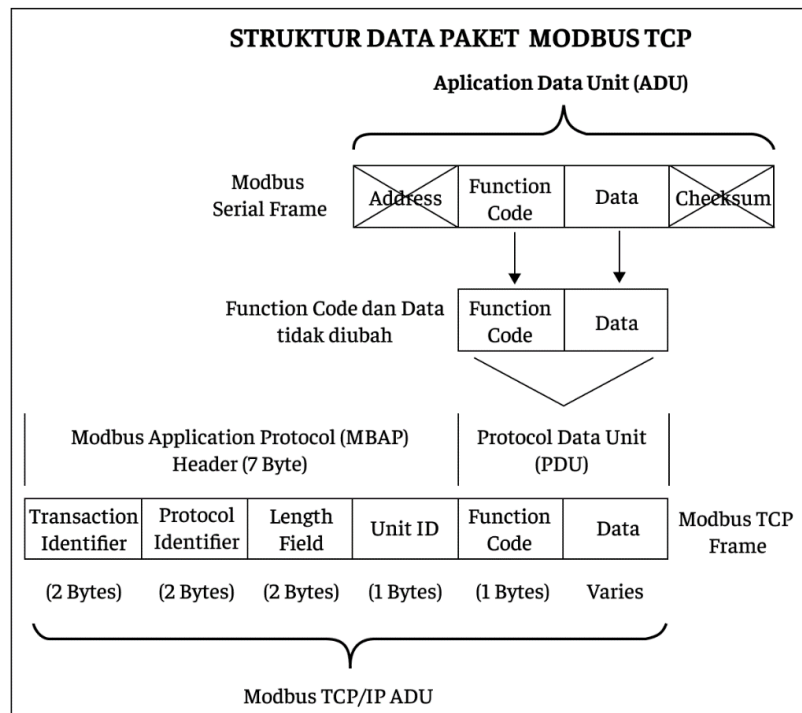
Sumber : (Tosin, 2021)

Lokasi	Nama	Nilai data	Tipe	Keterangan
00001-9999	<i>Discrete output coils</i>	1-bit	<i>Read-write</i>	<i>Master</i> dan <i>slave</i> dapat mengubah <i>data coil</i>
10001-19999	<i>Discrete input contact</i>	1-bit	<i>Read-only</i>	Data hanya diubah oleh <i>slave</i>
30001-39999	<i>Analog input registers</i>	16-bit words	<i>Read-only</i>	Data hanya diubah oleh <i>slave</i>

Lokasi	Nama	Nilai data	Tipe	Keterangan
40001-49999	Analog output holding registers	16-bit words	Read-write	Master dan slave dapat mengubah data register

b. Modbus Transmission Control Protocol (TCP)

Modbus TCP adalah protokol Modbus RTU yang memakai *interface* TCP yang berjalan pada media *ethernet* dan berjalan pada *layer* aplikasi TCP/IP. Protokol ini mengkombinasikan jaringan fisik berupa *ethernet* dengan *networking standard* (TCP/IP) dan metode standar untuk merepresentasikan data (Modbus sebagai protokol Aplikasi). Modbus TCP mengkolaborasikan *frame* data *modbus* standar ke dalam *frame* TCP tanpa *checksum*. Kemudian *frame* alamat modbus digantikan dengan identifikasi unit di modbus TCP dan menjadi bagian dari *Modbus Application Protocol* (MBAP) *header*. Paket data tersebut kemudian ditambahkan alamat IP dan dikirim dari *client* ke *server* sesuai alamat ip yang diberikan. Struktur *frame* dari Modbus TCP dapat dilihat pada gambar berikut 2.8



Gambar 2. 8 Format *frame* data Protokol Modbus TCP

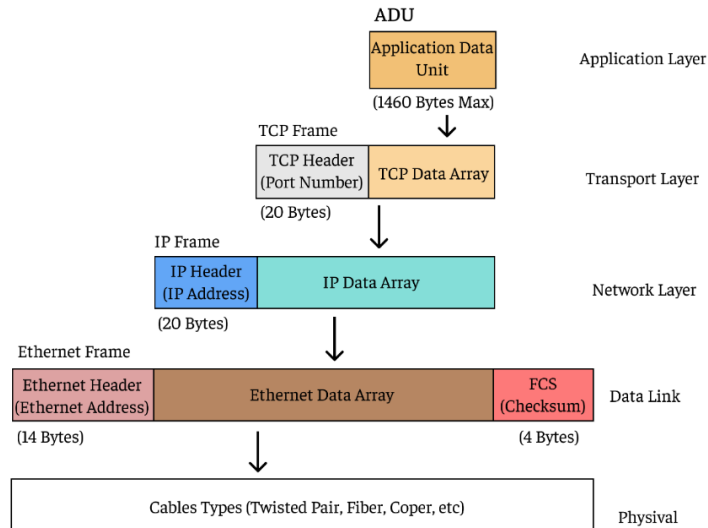
Sumber : (Tosin, 2021)

Berdasarkan gambar, MBAP *header* berjumlah 7-byte terdiri dari :

1. *Transaction identifier* (2-bytes), digunakan untuk *pairing* transaksi beberapa pesan dikirim secara bersamaan dengan koneksi TCP yang sama oleh *client* tanpa mengganggu respon sebelumnya.
2. *Protocol identifier* (2-bytes), bagian ini selalu 0 untuk modbus dan nilai lainnya dicadangkan untuk ekstensi jika diperlukan.
3. *Length field* (2-bytes), *field* ini digunakan untuk mengidentifikasi *remote server* yang berada di luar jaringan TCP, sebagai penghubung serial. Pada aplikasi server modbus TCP tertentu, unit ID ditentukan ke 00 atau FF.

Saat komunikasi data pada modbus TCP terjadi, terdapat perubahan struktur data pada setiap layer yang dilewati. Proses perubahan ini disebut enkapsulasi. Pada

application layer, protokol modbus membangun format *frame* yang disebut dengan *application data unit* (ADU) yang terdiri dari MBAP dan *Protocol Data Unit Modbus*. Pada *transport layer*, ADU ditambahkan *TCP header* supaya dapat dikirim melalui jaringan TCP. *Frame* pada layer ini disebut *TCP Frame*. Pada *network layer*, *TCP Frame* ditambahkan dengan *IP Header* untuk pengalamatan data agar dapat menuju alamat yang diinginkan, *frame* pada proses ini disebut *IP Frame*. Pada *data link layer*, *IP Frame* ditambahkan *ethernet header* dan *frame Check Sequence* (FCS) dan berubah nama menjadi *ethernet frame*. Pada *physical layer*, *ethernet frame* dikirim melalui jaringan fisik seperti kabel, *fiber optic*, dan media transmisi sejenisnya. Proses enkapsulasi data direpresentasikan pada Gambar 2.9

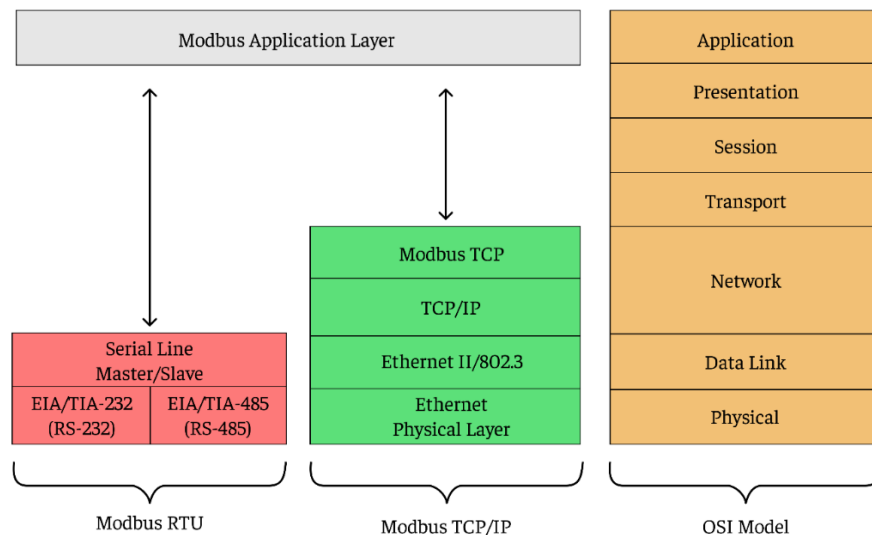


Gambar 2. 9 Representasi proses enkapsulasi data Modbus TCP

Sumber : (Tosin, 2021)

Modbus TCP menggunakan topologi jaringan *Client* dan *server* yang harus terhubung saat proses pertukaran data, pertukaran ini dijalankan melalui *port* 502.

IEEE 802.3 *Ethernet* adalah protokol jaringan yang diterapkan secara umum dan merupakan standar terbuka yang didukung banyak produsen serta infrastrukturnya tersedia secara luas. Oleh karena itu, TCP/IP digunakan sebagai dasar untuk akses ke *World Wide Web*. Modbus TCP berbagi lapisan fisik dan *data link* yang sama dengan *IEEE 802.3 ethernet* dan memakai *TCP/IP suite protocol* yang sama. Hal ini membuat Modbus TCP dapat digunakan pada infrastuktur *ethernet* yang sudah ada mulai dari *wiring*, *connector*, *network interface card*, *hub*, dan *switch*. Perbedaan antara Modbus RTU dan Modbus TCP dapat diketahui dari *stack layers protocol* berdasarkan model referensi OSI, dimana Modbus RTU yang berjalan pada *Application Layer* terhubung ke *serial line master/slave* pada *data link layer*. Sedangkan Modbus TCP berjalan pada *Application Layer* terhubung dengan *network layer*. Berikut Gambar 2.10 yang menjelaskan *stack layer* protokol modbus.

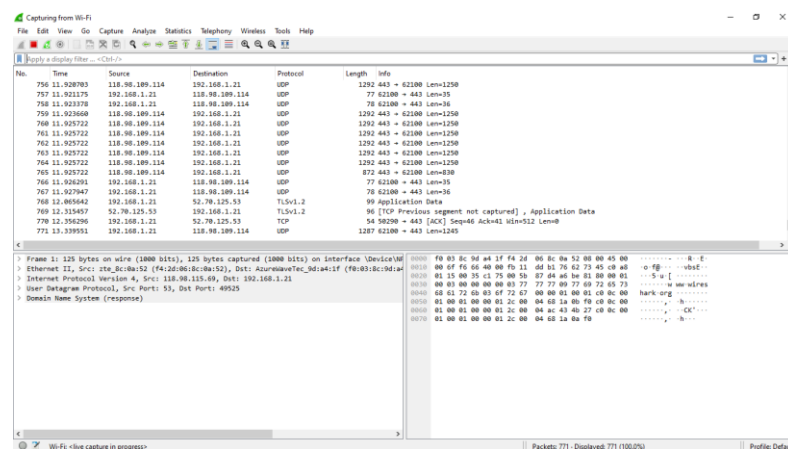


Gambar 2. 10 *Stack Layer* Protokol Modbus

Sumber : (Tosin, 2021)

2.4 Wireshark

Wireshark adalah alat ukur yang digunakan untuk mengukur parameter-parameter dalam komunikasi data dalam jaringan. Wireshark merupakan perangkat lunak yang berfungsi sebagai *network alayzer*. Wireshark bekerja dengan menangkap paket data yang berlalulintas dalam sistem (Nisa et al., 2024; Nuryani et al., 2024). Tampilan dari Wireshark ditunjukkan dengan Gambar 2.11



Gambar 2. 11 Tampilan Wireshark

2.5 Penelitian Terkait

Terdapat beberapa penelitian sebelumnya yang berkaitan dengan komunikasi antara PLC dengan mikrokontroler lain seperti Arduino, ESP32, dan sebagainya. Penelitian yang telah dipublikasikan dijelaskan dalam Tabel 2.6.

Tabel 2. 6 Penelitian terkait

No.	Judul Jurnal	Penulis, Tahun	Pembahasan Jurnal
1	Perancangan Sistem Komunikasi PLC Dengan <i>Raspberry PI</i> Via Protokol Modbus	(Amaliawati et al., 2020)	Penelitian ini membahas perancangan sistem komunikasi antara PLC dengan Raspberry Pi menggunakan Protokol Modbus RTU. Sistem tersebut dignakan

No.	Judul Jurnal	Penulis, Tahun	Pembahasan Jurnal
			untuk memonitoring data dari jarak jauh melalui platform ANTARES. Pengujian dilakukan untuk mengetahui kecepatan respon dan tingkat kesuksesan proses pembacaan data dan pengiriman data. Parameter yang dianalisis adalah baudrate dan jumlah data terhadap waktu proses Raspberry Pi. Diketahui dari hasil pengujian baudrate menunjukkan waktu proses tercepat adalah baudrate 115200 bps dengan rata-rata waktu pembacaan data 0,025 detik. Kesimpulan lainnya adalah semakin banyak data memori dan input/output yang dikirim maka waktu proses pembacaan data dan pengiriman ke ANTARES semakin lama, namun tingkat kesuksesan keutuhan data dari semua percobaan adalah 100%.
2	Analisis Kinerja Sistem Komunikasi PLC Dengan Raspberry Pi Via Protokol Modbus Dan Protokol Fins Berbasis Internet of	(Shobirin et al., 2021)	Jurnal ini membahas mengenai kinerja sistem komunikasi antara PLC dengan Raspberry Pi menggunakan dua protokol yaitu Protokol Modbus dan protokol FINS. Analisis kinerja sistemnya

No.	Judul Jurnal	Penulis, Tahun	Pembahasan Jurnal
	Things (IoT) Performance Analysis Of Plc Communication System With Raspberry Pi Via Modbus Protocol and Fins Protocol Based On Internet of Things		meliputi pengaruh <i>baudrate</i> terhadap waktu pencarian perangkat, pembacaan data, dan penulisan data. Selain itu, perbandingan kinerja dari dua protokol terhadap pengaruh jumlah data memori terhadap waktu proses Raspberry Pi dalam pembacaan dan penulisan data. Analisis tersebut bertujuan untuk mengetahui perangkat dan protokol komunikasi mana yang paling optimal berdasarkan parameter yang diuji. Hasilnya, PLC CP1L menunjukkan kinerja yang lebih optimal daripada PLC Siemens S7-1200.
3	Creating an Ethernet Communication Between a Simatic S7- 1200 PLC And Arduino Mega For An Omnidirectional Mobile Platform And Industrial Equipment	(Marosan et al., 2020)	Penelitian ini membahas desain sistem komunikasi antara PLC Siemens S7-1200 dan Arduino Mega dengan Raspberry Pi untuk memperluas jumlah <i>input/output</i> dari PLC. komunikasi antara perangkat-perangkat tersebut menggunakan media Ethernet dengan protokol Modbus TCP. Untuk <i>software interface</i> , penelitian ini menggunakan <i>platform</i> Node-RED. Pengujian

No.	Judul Jurnal	Penulis, Tahun	Pembahasan Jurnal
			dilakukan dengan mengkomunikasikan perangkat PLC, Arduino Mega, dan Raspberry Pi. Hasil dari penelitian ini adalah berhasilnya komunikasi Ethernet yang dilakukan pada semua perangkat komunikasi tersebut. Komunikasi berhasil dijalankan setelah Arduino Mega dan Raspberry Pi di-<i>install</i> library khusus di Node-RED.
4	Perancangan dan Implementasi Komunikasi RS-485 Menggunakan Protokol Modbus RTU dan Modbus TCP pada Sistem Pick-By-Light	(Tosin, 2021)	Penelitian ini membahas mengenai perancangan dan implementasi sistem Pick-by-Light dengan menggunakan komunikasi serial RS-485, Modbus RTU dan Modbus TCP/IP. Rancangan sistem menerapkan Light Indicator untuk menunjukkan lokasi barang. Penelitian ini membandingkan protokol RS-485, Modbus RTU dan Modbus TCP/IP sebagai protokol komunikasi pada mikrokontroler Arduino Mega Pro. Hasilnya semua komunikasi berhasil dijalankan pada semua protokol, sistem juga dapat

No.	Judul Jurnal	Penulis, Tahun	Pembahasan Jurnal
			berjalan pada jarak 30 M antara beberapa slave dan master pada konfigurasi multiple point.

Penelitian terkait memberi kemudahan dalam penelitian ini, seperti memberikan penjelasan tambahan, memudahkan analisa, dan perbandingan antara penelitian terkait dengan penelitian yang akan dilaksanakan. Beberapa penelitian terkait diantaranya yaitu :

1. Penelitian pada Tabel 2.6 nomor 1 digunakan sebagai sumber referensi untuk menjelaskan teori mengenai komunikasi antara PLC dan Mikrokontroler. Perbedaan tersebut memakai PLC yang dihubungkan dengan Raspberry Pi menggunakan protokol Modbus RTU dengan kabel serial. Sedangkan pada penelitian yang akan dilaksanakan, sistem menggunakan Arduino Portenta sebagai mikrokontroler dan dihubungkan dengan protokol Modbus TCP.
2. Penelitian pada Tabel 2.6 nomor 2 digunakan sebagai acuan untuk menganalisa kinerja dari sistem yang akan dibangun. Pada penelitian tersebut, protokol yang digunakan adalah Protokol Modbus RTU dan FINS, sedangkan pada penelitian yang akan dilaksanakan menggunakan Protokol Modbus TCP dengan Wi-Fi.
3. Penelitian pada Tabel 2.6 nomor 3 digunakan sebagai penjelasan tambahan mengenai Protokol Modbus TCP. Penelitian tersebut menghubungkan antara PLC Siemens S7-1200 dengan Arduino Mega, sedangkan pada penelitian ini digunakan PLC Omron CP1L dan Arduino Portenta H7.

4. Penelitian Pada Tabel 2.6 nomor 4 digunakan sebagai penjelasan tambahan mengenai Protokol Modbus TCP yang akan dipakai dalam penelitian ini. Pada penelitian tersebut menghubungkan antara PLC dengan Arduino Mega dengan protokol Modbus TCP dan Modbus RTU dan membuat perancangan Sistem Pick-by-Light. Sedangkan pada penelitian ini, akan membahas mengenai sistem komunikasi PLC Omron CP1L dengan Arduino Portenta H7 menggunakan Protokol Modbus TCP melalui Wi-Fi.