Implementasi Modbus TCP dalam Pengiriman Data Deteksi Objek ke PLC Schneider M221

Enas Dhuhri Kusuma[1*], Tariq Fitria Aziz[1], dan Wahyu Dewanto[1]

[1]Departemen Teknik Elektro dan Teknologi Informasi, Universitas Gadjah Mada, Yogyakarta, Indonesia

[*]Email: enas@ugm.ac.id

***Abstrak*—Visi komputer telah diaplikasikan dalam berbagai sektor termasuk sektor industri. Meskipun demikian, kurangnya alat praktikum yang memadai di perguruan tinggi dapat menyebabkan celah pengetahuan antara industri dan perguruan tinggi sebagai pencetak insinyur. Untuk menutup kekurangan tersebut, sebuah alat peraga deteksi objek di atas konveyor peraga berbasis Raspberry Pi 4 telah didesain di Laboratorium Schneider Electric Universitas Gadjah Mada. Pada penelitian ini dilakukan implementasi sistem komunikasi antara Raspberry Pi 4 pada sistem konveyor dengan PLC Schneider M221 sehingga dapat terjadi pengiriman data deteksi objek. Komunikasi ini diimplementasikan dengan protokol Modbus TCP. Data deteksi berhasil ditransfer secara akurat ke memori dan mengubah kondisi *relay output* PLC M221 sehingga dapat menjadi dasar pengendalian lebih lanjut. Pengiriman data memiliki latensi sebesar masing-masing 0,007249 detik dan 0,009185 detik untuk data dalam bit dan *word*.**

Kata kunci: Modbus TCP, PLC, Raspberry Pi 4, Visi komputer

***Abstract*-Computer vision has been already implemented in many sectors, including industrial. Nevertheless, lack of decent laboratory props in the university can cause a knowledge gap between industry and higher education. To fill that gap, a conveyor-top object detection system based on Raspberry Pi 4 has been devised in Schneider Electric Engineering lab Universitas Gadjah Mada. On this research, communication system between Raspberry Pi 4 on the conveyor system to Schneider M221 PLC is implemented to make the PLC able to receive object detection data. The communication is implemented with Modbus TCP protocol. Detection data can be transferred accurately to memory and able to switch PLC output relay conditions. Data transmission has latencies of 0,007249 second and 0,009185 second for data bit and word respectively.**

Keywords: Modbus TCP, PLC, Raspberry Pi 4, Computer Vision

I. PENDAHULUAN

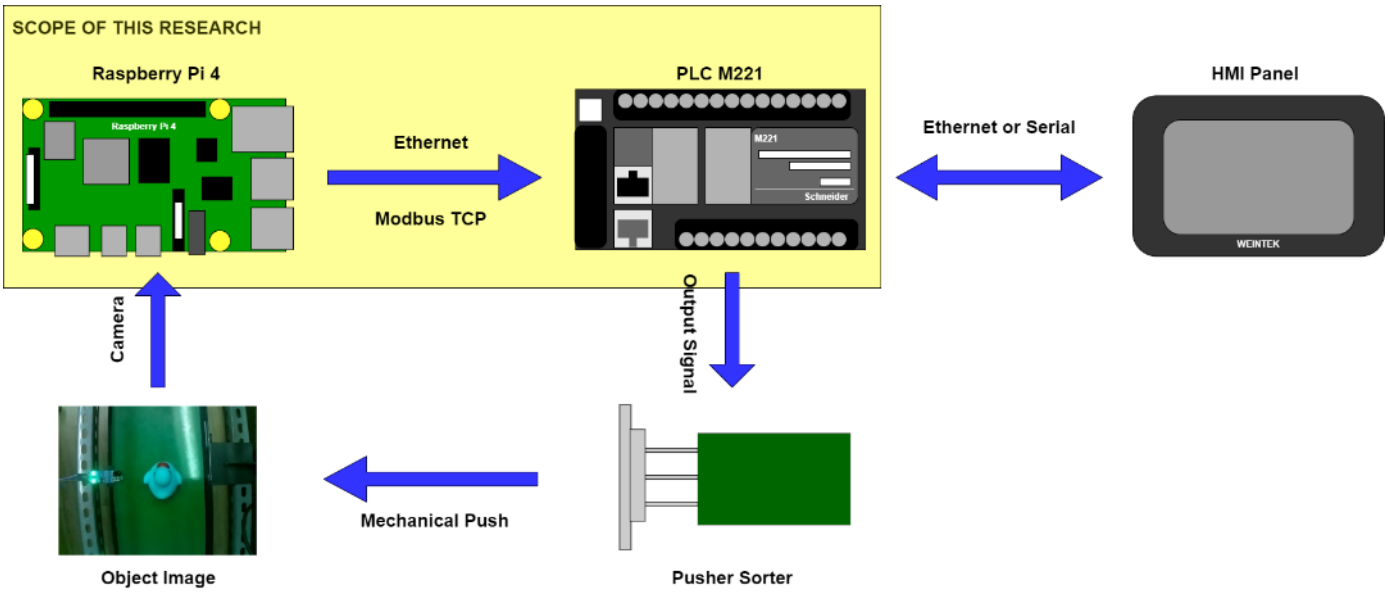
PERKEMBANGAN visi komputer menyebabkan aplikasi teknologi ini menjadi beragam. Teknologi visi komputer dapat digunakan untuk membantu kehidupan sehari-hari seperti untuk menentukan lokasi parkir yang kosong [1]. Teknologi ini juga dapat diaplikasikan pada sistem yang spesifik seperti sistem di industri manufaktur.

Pada industri manufaktur, visi komputer dapat digunakan untuk tiga hal: deteksi kecacatan objek, deteksi alat pelindung diri (*personal protective equipment*), dan pengawasan[2]. Sebagai contoh, visi komputer telah dipakai dalam mendeteksi kecacatan papan sirkuit cetak (*printed circuit board*)[3]. Hal ini menunjukkan bahwa visi komputer mampu mendorong terciptanya sistem manufaktur yang cerdas (*smart manufacturing system*).

Sistem manufaktur cerdas memiliki beberapa fitur. Salah satu fitur yang dapat ditemui adalah *otonomi* [4]. Otonomi menunjukkan kemampuan sistem untuk menyelesaikan tugas tanpa bantuan manusia. Otonomi dapat dicapai dengan memanfaatkan teknologi tertentu pada sistem manufaktur seperti misalnya teknologi kecerdasan buatan (*artificial intelligence*) [4]. Sistem yang otonom akan memiliki produktivitas dan fleksibilitas dalam produksi sehingga dapat menguntungkan pemilik industri.

Sistem manufaktur cerdas juga ditandai oleh aktivitas pengumpulan dan analisis data yang masif [5]. Pemasangan sensor yang luas dan beragam memungkinkan sistem manufaktur untuk mengumpulkan data lingkungan seperti suhu, konsumsi energi, dan tekanan. Data ini dapat diolah menjadi sebuah tindakan dalam sistem atau digunakan untuk melatih program pembelajaran mesin (*machine learning*) [4]. Program tersebut kemudian dapat digunakan untuk mengenali pola pada data sehingga membantu sistem menjadi otonom.

Sebagai bagian dari teknologi kecerdasan buatan dan pembelajaran mesin, aplikasi visi komputer di industri perlu diajarkan ke calon insinyur yang sedang menempuh studi. Hal ini ditujukan untuk menutup celah (*gap*) pengetahuan antara



Gambar. 1. Cakupan penelitian relatif terhadap rencana pembuatan alat peraga deteksi objek di atas konveyor.

universitas dan industri. Alasan lain adalah karena pelatihan dan edukasi yang inovatif merupakan faktor pendukung (*enabling factor*) transformasi sistem manufaktur cerdas [4]. Oleh karena itu, sebuah alat peraga industri dengan kemampuan visi komputer perlu dirancang untuk mendukung transformasi tersebut.

Proses perancangan alat peraga deteksi di industri telah berhasil dijalankan di Laboratorium Schneider Electric Universitas Gadjah Mada berdasarkan sistem yang sebelumnya telah dibuat [6]. Alat deteksi ini mampu mendeteksi objek yang bergerak di atas konveyor berdasarkan perbedaan warna. Meskipun demikian, alat peraga tersebut masih berdiri sendiri dan belum terintegrasi ke sistem yang lebih besar seperti layaknya di industri.

Penelitian ini bertujuan untuk menghubungkan alat peraga deteksi dengan *programmble logic controller* (PLC) Schneider M221 sebagai pengendali. Hal ini dicapai dengan memanfaatkan protokol Modbus TCP untuk komunikasi antara dua sistem. Dengan Modbus TCP, data deteksi yang dihasilkan alat peraga mampu dikirimkan ke PLC sehingga dapat digunakan untuk mengambil tindakan lebih lanjut. Kemampuan dari sistem berbeda untuk saling bertukar informasi dapat disebut sebagai *interoperabilitas* dan hal ini termasuk karakteristik sistem manufaktur cerdas [4]. Dengan adanya interoperabilitas ke PLC, alat peraga yang dirancang diharapkan lebih menyerupai sistem di industri sehingga meningkatkan pemahaman mahasiswa yang menggunakannya.

Dalam menjelaskan penelitian yang dilakukan, artikel ini akan dibagi menjadi empat bagian. Bagian pertama, *Pendahuluan*, berisi latar belakang yang mendorong pelaksanaan penelitian. Bagian kedua, *Metode*, menjelaskan teknis pelaksanaan dan sumber daya yang digunakan dalam penilitian. Bagian ketiga, *Hasil dan Pembahasan*, berisi pembahasan terhadap hasil pengujian rancangan yang dibuat. Artikel ditutup dengan bagian *Kesimpulan* yang merangkum temuan dalam penelitian.

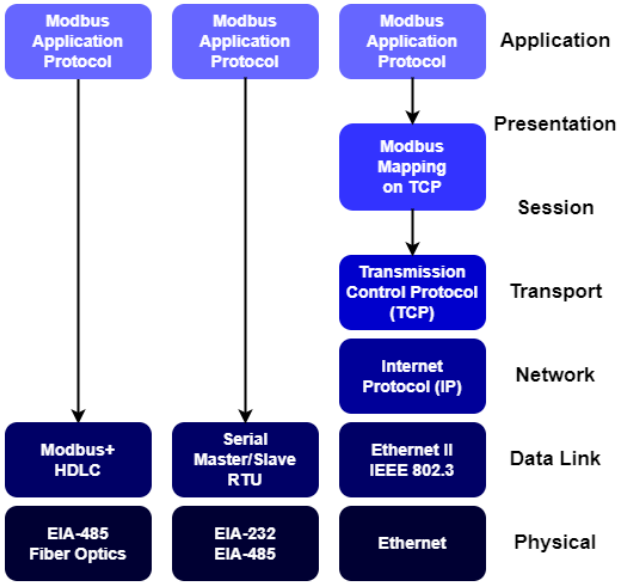
II. METODE

A. Cakupan Penelitian

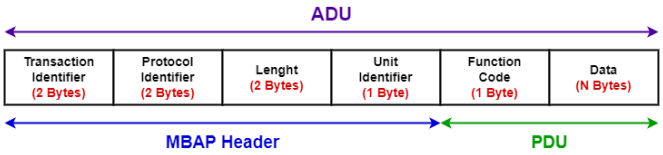
Penelitian ini merupakan bagian dari proyek pembuatan alat peraga deteksi objek di atas konveyor yang dilaksanakan di Laboratorium Schneider Electric Universitas Gadjah Mada. Gambaran umum proyek ini dapat diamati pada Gambar 1. Penelitian ini difokuskan pada upaya menghubungkan alat deteksi berbasis Raspberry Pi 4 dengan PLC Schneider M221. Penelitian sebelumnya telah berhasil membuat alat peraga deteksi yang mampu mengenali objek bergerak di atas konveyor [6]. Alat peraga ini dibangun dengan memakai Raspberry Pi 4. Sistem deteksi objek berbasis Raspberry Pi 4 tersebut, akan dihubungkan dengan perangkat kendali industri standar seperti PLC. Alat peraga tersebut mampu mengumpulkan citra objek, mengolahnya dengan model pembelajaran mesin, dan menghasilkan data deteksi. Data deteksi tersebut terdiri dari prediksi warna, rasio deteksi, waktu tunda deteksi, dan nilai *frame rate*.

Alat peraga deteksi yang telah dirancang akan dihubungkan dengan komponen lain seperti PLC, panel *human machine interface* (HMI), dan *pusher* berbasis mesin listrik. Apabila semua komponen ini berhasil dihubungkan seperti Gambar. 1., alat peraga deteksi akan mampu bekerja selayaknya alat di industri. Hal ini akan meningkatkan kualitas edukasi mahasiswa dan calon insinyur di Laboratorium Schneider Electric.

Sebagai bagian dari proyek, penelitian ini dilakukan untuk menjalin dan menguji komunikasi antara Raspberry Pi 4 dengan PLC Schneider M221. Komunikasi akan dirancang sedemikian rupa sehingga alat peraga dapat mengirimkan data deteksi yang telah dikumpulkan ke PLC. Komunikasi antara Raspberry Pi 4 dan PLC Schneider M221 akan memanfaatkan protokol Modbus TCP yang berjalan di atas Ethernet.



Gambar. 2. Perbandingan model lapisan OSI dari varian Modbus



Gambar. 3. *Application Data Unit* (ADU) pada Modbus TCP

B. *Mekanisme Komunikasi Modbus TCP*

Modbus merupakan protokol komunikasi industri yang dikembangkan oleh Gould Modicon (sekarang berada di bawah Schneider Electric). Protokol ini telah menjadi standar komunikasi serial di industri sejak tahun 1979. Di luar industri, protokol ini juga diaplikasikan pada beragam skenario seperti sistem pengawasan bangunan ataupun modul komunikasi untuk pelatihan [7], [8]

Dalam model *Open Systems Interconnection* (OSI), Modbus termasuk protokol di lapisan ke-7, *application layer*[9]. Modbus memiliki varian berbeda tergantung jenis protokol yang digunakan pada lapisan di bawahnya. Hal ini dapat diamati pada Gambar. 2. Pada gambar tersebut, Modbus dapat dibagi menjadi Modbus+, Modbus RTU, dan Modbus TCP [9], [10].

Dalam penelitian ini, varian Modbus yang digunakan adalah Modbus TCP. Modbus TCP memanfaatkan tumpukan (*stack*) TCP/IP sebagai protokol pada *transport layer* hingga *physical layer*. Pada *physical* layer, perangkat akan dihubungkan melalui Ethernet (*port RJ-45)* dengan kabel Cat 5E atau Cat 6.

Komunikasi dalam Modbus TCP dijalankan dengan skema *query/response* atau *request/reply* antara *client* dan *server*. *Client* menyatakan perangkat yang memulai komunikasi dengan mengirimkan *query* ke *server*. *Server* kemudian bertugas untuk menerima *query* dan mengirimkan *response* yang sesuai. *Client* dan *server* dapat disebut juga sebagai *master* dan *slave*.

Dalam berkomunikasi, Modbus mendefinisikan kerangka (*frame*) data yang dikenal dengan *Application Data Unit* (ADU) [9]. ADU pada Modbus TCP diilustrasikan oleh Gambar 3. ADU pada Modbus TCP dibentuk oleh *Modbus Application Protocol* (MBAP) *Header* dan *Protocol Data Unit* (PDU) [11]. MBAP dan PDU memiliki bidang (*field*) yang berisi informasi pada Tabel I.

TABLE I
BIDANG KERANGKA DATA PADA ADU

Bidang	Keterangan
Transaction Id	Nilai unik untuk menandai *request* dan *response* yang bersangkutan pada transaksi data.
Protocol Id	Jenis protokol yang digunakan dalam komunikasi. Secara umum, nilai dibuat tetap, yaitu nilai *nol*, untuk protokol Modbus.
Length	Panjang data dalam *Bytes* untuk bidang-bidang selanjutnya (*Unit ID* hingga *Data*).
Unit Identifier	Identifikasi bagi perangkat Modbus *slave* yang dihubungkan melalui komunikasi serial
Function Code	Kode identifikasi untuk menentukan jenis transaksi data mulai dari transaksi pembacaan hingga penulisan memori
Data	Bidang ini memiliki isi yang bervariasi tergantung pada jenis transaksi dan apakah ADU merupakan bagian *request* atau *response*

TABLE II
MODEL DATA PADA MODBUS

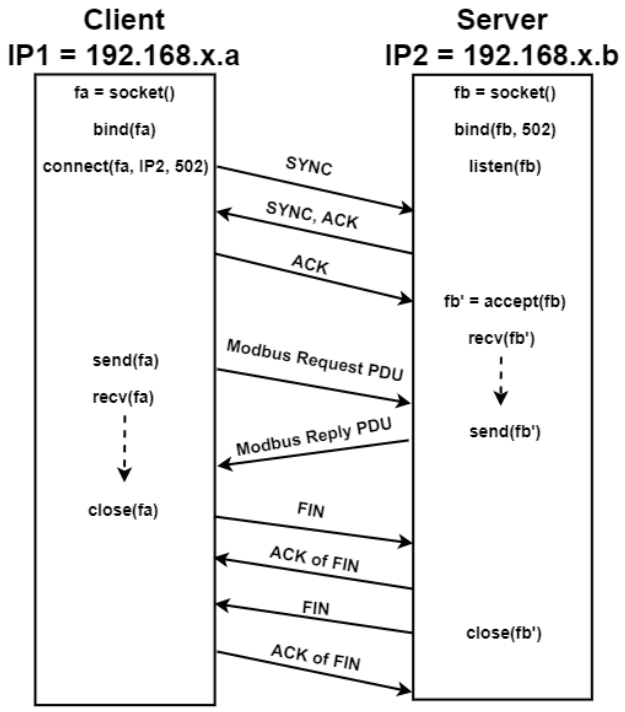
Model Data	Panjang	Jenis Operasi
Discrete Input	1-bit	*Read-only*
Coils	1-bit	*Read-Write*
Input Registers	2-Byte (word)	*Read-only*
Holding Registers	2-Byte (word)	*Read-Write*

ADU pada lapisan Modbus akan dikirimkan ke lapisan TCP/IP. Di sini, ADU akan dibungkus oleh bidang tambahan yaitu *TCP Header*, *IP Header*, dan *Ethernet Header*. Paket data ini kemudian dikirimkan ke perangkat lain melalui jaringan.

Modbus memiliki empat model data: *discrete input*, *input registers*, *coils*, dan *holding registers*. Model data ini akan menentukan panjang *Byte* data yang dapat dikirimkan beserta tipe operasi yang dapat dilakukan. Umumnya, perangkat *slave* Modbus telah menyediakan memori aplikasi untuk masing-masing model data tersebut. Model data pada Modbus dapat secara jelas diamati pada Tabel II.

Dari Tabel II, data yang dikirimkan dapat memiliki panjang 1-bit atau 2-Byte (sering disebut *word*). Sementara itu, dari jenis operasi, Modbus memiliki data yang hanya dapat diubah oleh sistem *Input/Output* (I/O) dan data yang dapat diubah oleh aplikasi program. Data yang disebut pertama adalah data *read-only* sementara data kedua adalah data *read-write*.

Dalam menjalankan komunikasi pada lapisan TCP/IP, Modbus menggunakan antarmuka *socket Berkeley Software Distribution* (BSD) [11]. Dengan antarmuka tersebut, pertukaran pesan antara *client* dan *server* pada Modbus TCP melewati proses yang ditunjukkan Gambar. 4. Di sini, komunikasi dilakukan dengan terlebih dahulu mendefinisikan *socket* pada *client* dan *server*. *Client* kemudian dapat membangun koneksi dengan memasukkan Alamat IP *server* dan nomor *port* (pada Modbus, *port* 502 selalu digunakan). Apabila koneksi diterima, *client* dapat mengirimkan *request* ke *server*. *Client* kemudian dapat menutup koneksi yang



Gambar. 4. Pertukaran pesan antara *client* dan *server* pada Modbus TCP

digunakan apabila *response* telah diterima dari *server*.

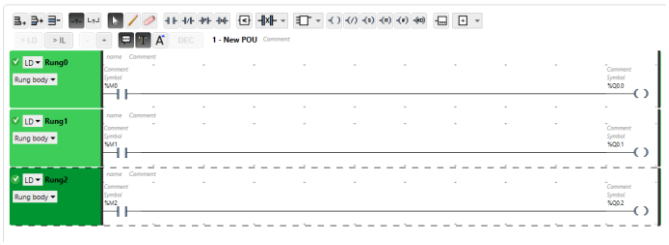
Modbus TCP memiliki beberapa keunggulan dibandingkan varian Modbus lain seperti Modbus RTU. Keunggulan pertama adalah Modbus TCP dapat memiliki *client* dan *server* yang berjumlah jamak. *Client* Modbus TCP dapat berkomunikasi dengan lebih dari satu *server* secara bersamaan, begitu juga sebaliknya. Keunggulan lain adalah sebuah perangkat Modbus TCP dapat berperan sebagai *client* dan *server* pada waktu yang sama. Keunggulan terakhir, ketika mengirimkan *request* ke *server*, *client* Modbus TCP tidak dituntut untuk menunggu *response*. Hal ini membuat *client* dapat mengirimkan lebih dari satu *request* ke *serve*r.

C. Konfigurasi PLC Schneider M221

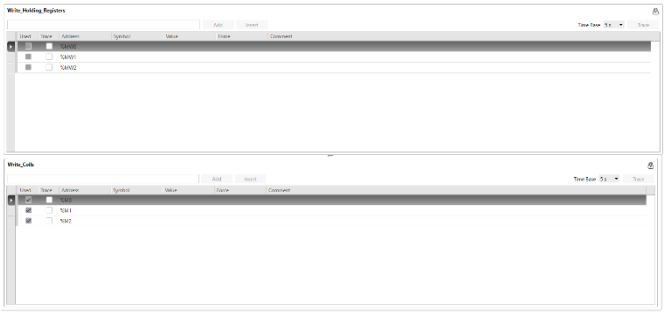
PLC Schneider M221 merupakan seri PLC dari keluarga Modicon yang diproduksi oleh Schneider Electric. PLC M221 memmiliki beberapa varian produk seperti TM221CE16T dan TM221CE40R. Pada penelitian ini, PLC yang digunakan adalah varian TM221CE40R.

PLC TM221CE40R membutuhkan suplai tegangan AC sebesar 100-240 V. PLC ini memiliki I/O yang terdiri dari 24 *input* digital, 2 *input* analog, dan 16 *output* digital. PLC ini memiliki *port* USB, Ethernet, dan Serial yang dapat dipakai untuk mengunduh program dari komputer dan berkomunikasi dengan perangkat lain.

Dalam melakukan konfigurasi dan memprogram PLC M221, Schneider menyediakan perangkat lunak Ecostruxure Machine Expert – Basic. Aplikasi ini memberikan opsi kepada pengguna untuk mengatur konfigurasi Ethernet pada PLC. Melalui *tab Configuration*, pengguna dapat mengubah alamat IP PLC, parameter keamanan, dan *client* yang terhubung melalui Modbus TCP. Sementara itu, pada *tab Programming*, pengguna



Gambar 5. Program yang dijalankan di PLC M221



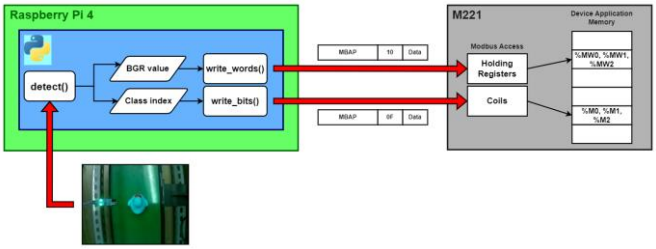
Gambar. 6. *Animation tables* berisi memori PLC yang akan menerima data deteksi

dapat membuat program yang akan dijalankan PLC dalam bentuk *Ladder Diagram*. Di *tab* ini pula pengguna dapat melihat perubahan status pada memori dan I/O PLC melalui *Animation tables*.

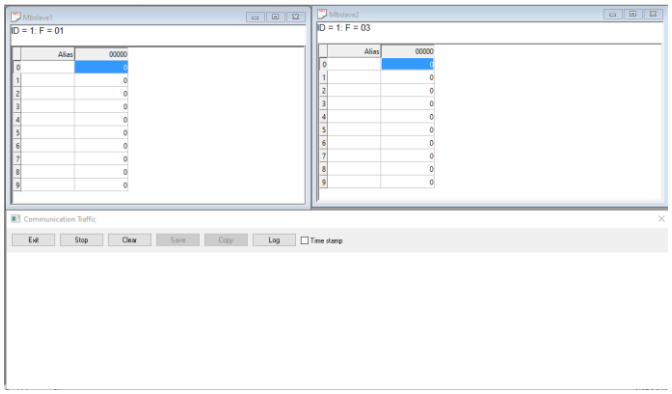
Dengan Machine Expert – Basic, PLC akan diatur agar mampu menerima *request* dari Raspbbery Pi 4 sebagai *server*. Setelah alamat IP untuk PLC ditentukan, alamat IP dari Raspberry Pi 4 akan ditambahkan ke daftar *client* Modbus TCP. Setelah itu, program yang akan dijalankan di PLC perlu dibuat dengan *Ladder Diagram*. Program yang dibuat dapat diamati pada Gambar. 5.

Program pada Gambar. 5 dibuat untuk menguji apakah data deteksi berhasil diterima oleh PLC atau tidak. Program ini terdiri dari tiga buah *contact* dan tiga buah *coil*. *Contact* diberi alamat yang sama dengan memori bit PLC yaitu di %M0, %M1, dan %M2. Sementara itu, *Coil* memiliki alamat yang sesuai dengan *output* digital: %Q0.0, %Q0.1, dan %Q0.2. Apabila *contact* berubah status dari 0 ke 1 maka *coil* juga akan ikut berubah. Perubahan *coil* ini akan menyalakan lampu indikator *relay* pada PLC M221.

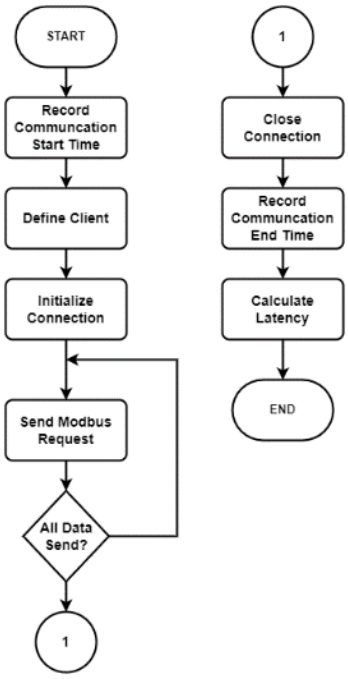
Selain melalui lampu *relay*, keberhasilan pengiriman data juga dapat diamati dari *Animation tables*. Pada *Animation tables*, pengguna dapat menembahkan alamat memori atau I/O yang akan diamati statusnya. Di sini, alamat memori bit %M0, %M1, dan %M2 akan ditambahkan pada *tables* yang bernama *Write_Coils.* Penamaan ini menyesuaikan model data yang difasilitasi Modbus TCP pada Tabel II. Sementara itu, *tables* lain yang bernama *Write_Holding_Registers* akan dibuat dengan isi alamat %MW0, %MW1, dan %MW2. Ketiga alamat ini adalah memori PLC yang mampu menyimpan *word*. Dengan demikian, *Animation tables* yang dihasilkan dapat diamati pada Gambar. 6.



Gambar. 7. Diagram proses pengiriman data deteksi dari Raspberry Pi 4



Gambar. 8. Tampilan perangkat lunak Modbus Slave



Gambar. 9. Algoritma pengiriman data dalam penelitian

Setelah program dan konfigurasi selesai, PLC akan masuk ke proses *Comissioning*. Di sini, Machine Expert – Basic akan dihubungkan ke PLC melalui proses *Login*. Program yang dibuat kemudian dapat diunduh ke PLC. Apabila semua persiapan selesai, PLC dapat dijalankan untuk menerima data deteksi dari Raspberry Pi 4.

D. Pengiriman Data dari Raspberry Pi 4

Deteksi objek dilakukan dengan program yang dikembangkan dalam bahasa Python [6]. Berangkat dari program tersebut, dalam penelitian ini, sebuah fungsi khusus dalam bahasa Python dan protokol Modbus akan dibuat untuk mengirimkan data ke PLC M221. Cara kerja dari fungsi yang dibuat dapat diamati pada Gambar. 7.

Pada Gambar. 7, citra objek akan dideteksi dengan fungsi *detect()* dan akan menghasilkan dua nilai: kombinasi warna BGR dan indeks kelas terprediksi. Kombinasi warna BGR, yang memiliki rentang nilai dari (0, 0, 0) hingga (255, 255, 255) akan dimasukkan ke fungsi *write_words()*. Sementara itu, indeks kelas yang memiliki (0, 0, 1), (0, 1, 0), atau (1, 0, 0) akan menjadi masukan fungsi *write_bits()*.

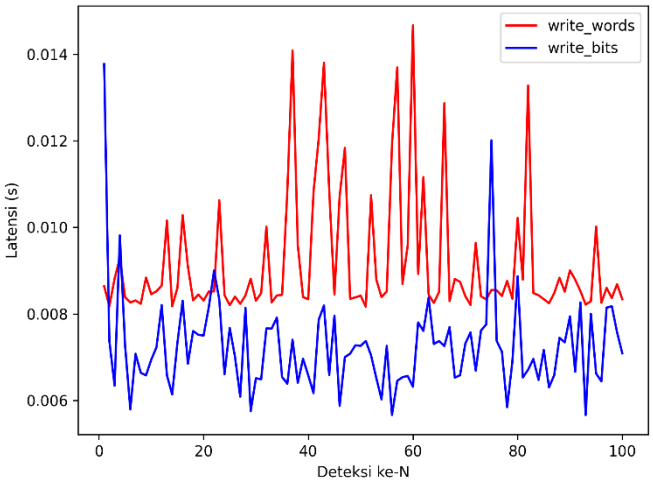
Seperti namanya, fungsi *write_words()* dan *write_bits()* akan menuliskan data ke memori *word* PLC dan memori bit PLC. Ketika fungsi tersebut dipanggil, Raspberry Pi 4 sebagai *client* akan mengirimkan *request* ke PLC. *Request* yang dikirimkan akan mengikuti format pada Gambar. 3. dengan *function code* dalam heksadesimal 10 dan 0F. *Function code* 10 dan 0F secara urut menyatakan transaksi untuk menulis register jamak (*write multiple registers*) dan menulis *coils* jamak (*write multiple coils*).

Demi mengamati ADU yang dikirimkan dari Raspberry Pi 4 secara detail, perangkat lunak Modbus Slave akan digunakan. Perangkat lunak ini dikeluarkan oleh *Witte Software* dan dapat diamati pada Gambar 8. Seperti namanya, perangkat lunak ini berfungsi untuk membuat *slave* (*server*) Modbus di komputer personal sehingga dapat menerima *request* dari *client*. Agar dapat menerima *request*, alamat IP dari *client* (dalam hal ini Raspberry Pi 4) perlu dimasukkan ke perangka lunak. Setelah itu, jenis model data yang diterima *slave* dapat diganti dengan menyesuaikan Tabel II. Apabila *slave* berhasil terhubung dengan *client*, perubahan nilai dapat diamati pada tabel dan detail ADU akan muncul pada *Communication Traffic*.

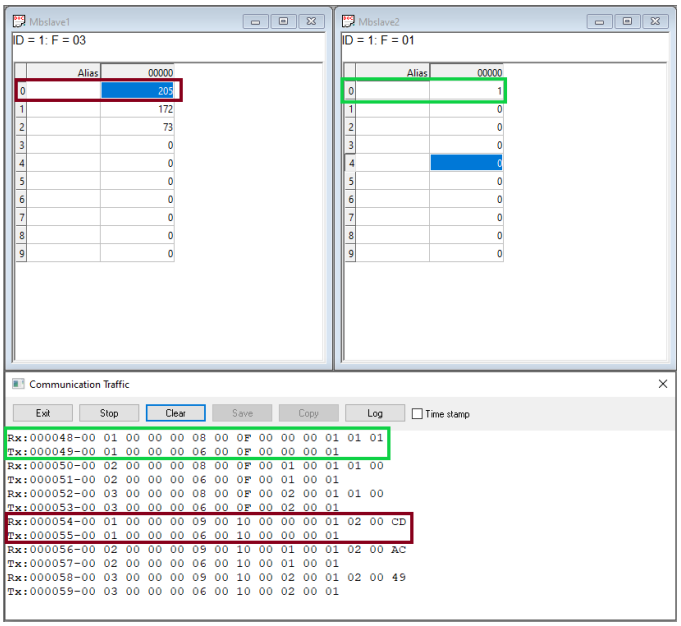
Selain mengamati detail ADU, penelitian juga dilakukan untuk menguji seberapa besar latensi dalam komunikasi Modbus TCP. Latensi diperoleh dengan menghitung selisih antara waktu yang tercatat di awal komunikasi dengan waktu yang tercatat di akhir komunikasi. Secara matematis, hal ini ditunjukkan oleh (1).

$$Latency = Comm\ Start\ time - Comm\ End\ time \quad (1)$$

Perhitungan latensi akan dimasukkan ke dalam algoritma yang digunakan untuk mengirimkan data deteksi. Algoritma ini dapat diamati pada Gambar. 9. Dalam mengimplementasikan algoritma tersebut dalam bahasa Python, pustaka *Pymodbus* dan modul *time* digunakan. Pymodbus merupakan pustaka khusus dalam bahasa Python untuk mengimplementasikan protokol Modbus. Pustaka ini berisi beberapa API untuk mendefinisikan *client* dan *server* serta untuk memulai komunikasi sinkron atau asinkron. Sementara itu, modul time menyediakan fungsi-fungsi yang berkaitan dengan waktu.



Gambar. 10. Perubahan latensi dalam pengiriman 100 data deteksi



Gambar. 11. Detail komunikasi antara Raspberry Pi 4 dengan Modbus Slave

III. HASIL DAN PEMBAHASAN

A. *Pengujian Latensi*

Pengujian pertama dilakukan terhadap latensi komunikasi dengan Modbus TCP. Komunikasi dilakukan untuk mengirimkan 100 data deteksi sehingga diperoleh 100 sampel data. Latensi yang tercatat selama proses pengiriman ini dapat diamati pada Gambar. 10.

Pada Gambar. 10. operasi *write_words* memiliki latensi yang lebih besar dibandingkan *write_bits*. Hal ini dapat disebabkan oleh ukuran paket data yang dikirimkan ketika fungsi tersebut dipanggil. Fungsi *write_words*() akan mengirimkan dua *Byte* data untuk setiap ADU. Apabila pengiriman terjadi untuk setiap nilai BGR, total data yang dikirimkan adalah sebesar 6 Byte. Sementara itu, fungsi *write_bits()* mengirimkan 3 bit data secara keseluruhan. Apabila jaringan yang digunakan memiliki *bandwidth* yang sama maka pengiriman paket data yang besar akan memakan waktu lebih lama dibandingkan paket data yang kecil.

Latensi untuk kedua operasi dapat ditentukan nilai minimum, maksimum, dan rata-ratanya. Ketiga nilai tersebut dapat diamati pada Tabel III. Dari tabel tersebut, selisih nilai maksimum-minimum latensi *write_words* dan *write_bits* berada pada angka 0,006523 detik dan 0,008108 detik. Sementara itu, rata-rata latensi berada di angka 0,009185 detik dan 0,007249 detik. Dengan demikian, operasi *write_words* rata-rata membutuhkan 126% waktu untuk mengirim data melalui operasi *write_bits*.

B. *Pengujian Paket Data*

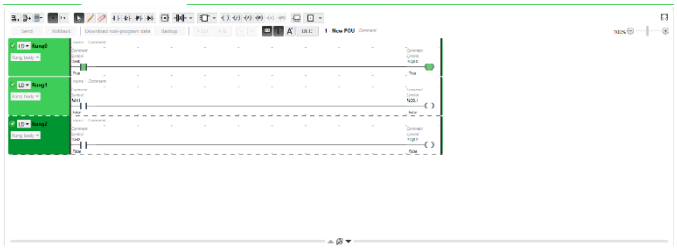
Dengan Modbus Slave, paket data yang dikirimkan dari Raspberry Pi 4 akan diamati untuk menentukan kesesuaian dengan format ADU yang ada. Hasil pengamatan paket data ini dapat dilihat pada Gambar. 11. Pada gambar tersebut, dua *slave* didefinisikan dengan data model *holding registers* pada *slave* pertama dan *coils* pada *slave* kedua. Perubahan nilai pada kedua *slave* menunjukkan bahwa data telah berhasil dikirimkan dari Raspberry Pi 4 melalui Modbus TCP.

Detail dari komunikasi yang terjadi antara *client* dan *slave* dapat dilihat pada *Communication Traffic*. Di sini, enam buah data paket tercatat yang menunjukkan tiga buah *request* dan tiga buah *response*. Data paket ini tertulis dalam format heksadesimal dan secara umum mengikuti format ADU pada Gambar. 3. Meskipun demikian, perbedaan spesifik muncul pada PDU karena mengikuti *function code* yang digunakan. Dalam mengamati perbedaan format PDU, analisis akan dilakukan pada paket data yang ditandai kotak merah dan hijau pada Gambar. 11. Nilai dari paket data tersebut pada tabel Modbus Slave juga ditandai oleh kotak dengan warna sama.

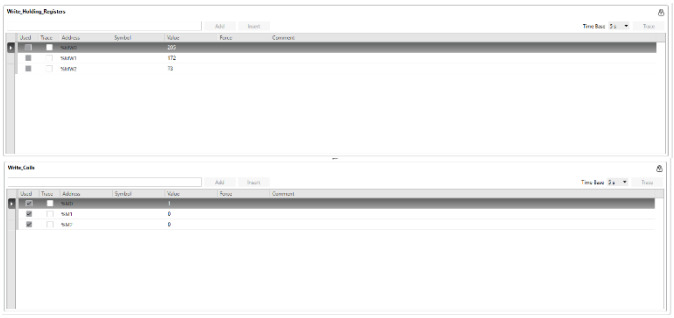
Kotak warna merah pada Gambar 11 menunjukkan transaksi untuk menuliskan nilai 205 (nilai *Blue* dari BGR) ke *slave holding register*. Paket data yang diterima pada baris RX menunjukkan *request* dari *client*. Sementara itu, TX berisi paket data yang dikirimkan sebagai *response* atau *reply*. Apabila paket data ini dipecah menjadi bagian-bagianya, hasilnya dapat diamati pada Tabel IV.

Dari Tabel IV, MBAP *Header* pada paket data sesuai dengan format pada Tabel I. Di sini, MBAP pada *request* dan *reply* memiliki nilai yang hampir sama dengan perbedaan pada bidang *Length.* Pada *request*, *Length* memilikki nilai 9 Byte sementara, pada *reply*, *Length* memiliki nilai 6 Byte. Perbedaan panjang ini disebabkan oleh *Data* PDU yang berbeda antara *request* dan *reply*.

Bidang *Data* PDU pada *request* akan berisi informasi seperti *Start Address*, *Registers Quantity, Byte Count,* dan *Data Value*. Sementara itu, *Data* PDU untuk *reply* hanya berisi *Start Address* dan *Registers Quantity*. Kedua informasi ini secara urut menunjukkan alamat awal memori dan jumlah memori yang akan diubah yaitu sebanyak 1 memori di alamat ke-0. Sementara itu, *Byte Count* dan *Data Value* menyatakan jumlah data yang akan dituliskan ke memori besar nilai data tersebut. Dalam hal ini, data yang dituliskan berjumlah 2 Byte dengan nilai 205.



Gambar. 12. Tampilan *Ladder Diagram* setelah pengiriman data



Gambar. 13. Tampilan *Animation tables* setelah pengiriman data

Pada Gambar. 11, kotak warna hijau menunjukkan transaksi penulisan nilai bit 1 (mewakili indeks kelas terprediksi) ke *slave coils*. Pada operasi *write_coils*,ADU yang diterima sebagai *request* dan dikirimkan sebagai *reply* tidak begitu berbeda dari ADU operasi *write_words*. Perbedaan utama terletak pada *function code* 0F dan nilai bidang *Data* PDU. Pada ADU, informasi yang ditukar antara *request* dan *reply* dapat diamati pada Tabel V. Di sini, bidang *Data* juga diawali oleh *Start Address* dan *Outputs Quantity*. Meskipun demikian, karena *coils* hanya menerima 1 bit data, *Data Value* hanya dapat berisi 1 Byte data dan hal ini juga mempengaruhi nilai *Byte Count*.

C. *Pengujian PLC*

Pengujian terakhir dilakukan dengan mengirimkan data dari Raspberry Pi 4 ke PLC M221. Hasil pengiriman ini diamati di Machine Expert – Basic. Hasil yang dimaksud dapat diamati pada Gambar 12 dan Gambar 13.

Dari Gambar 12, *Ladder Diagram* yang dibuat mengalami perubahan status *contact* pada alamat %M0 dari nilai nol ke satu. Hal ini membuat *coil* dengan alamat %Q0.0 akan ikut menyala. Perubahan status ini, disebabkan pengiriman nilai bit 1 dari Raspberry Pi 4 ke memori %M0. Sementara itu, *contact* dengan alamat lain, %M1 dan %M2, tetap mati karena Raspberry Pi 4 mengirimkan nilai bit 0. Apabila objek yang berbeda terdeteksi, *contact* dengan alamat berbeda akan bernilai 1 sementara *contact* %M0 dan satu *contact* lain akan bernilai nol. Dengan demikian, objek berbeda akan menyalakan *contact* dan *coil* berbeda pada *Ladder Diagram* PLC.

Dari Gambar. 13, perubahan memori bit dan memori *word* dapat secara lebih jelas diamati. Memori bit %M0, %M1, dan %M2 berubah sesuai dengan transaksi pada Gambar. 11. Data deteksi ini, yang dikirimkan melalui fungsi *write_bits()*, berisi nilai 1 pada indeks kelas terprediksi dan 0 untuk indeks kelas lain. Sementara itu, memori *word* %MW0, %MW1, dan %MW2 berhasil menerima nilai RGB yaitu (205, 172, 73) melalui fungsi *write_words()* seperti pada Gambar. 11. Hal ini menunjukkan bahwa Raspberry Pi 4 berhasil mengirimkan data deteksi ke memori PLC M221.

Gambar. 14. Tampilan PLC sebelum pengiriman data (atas) dan setelah pengiriman data (bawah)

Pengamatan terakhir dilakukan terhadap lampu indikator PLC M221. Perubahan lampu indikator *relay output* dapat diamati pada Gambar. 14. Dari gambar tersebut, lampu indikator yang mati akan menyala pada nomor *relay* 0. Alamat tersebut mewakili %Q0.0 pada *Ladder Diagram*. Perubahan lampu indikator ini akan memudahkan pengguna alat untuk dapat membedakan hasil deteksi yang diterima PLC.

IV. KESIMPULAN

Data deteksi objek telah berhasil dikirimkan dari alat peraga berbasis Raspberry Pi 4 ke PLC M221. Data deteksi yang dikirimkan terdiri dari indeks kelas objek yang diprediksi dan warna BGR objek. Data ini dikirimkan dalam format bit dan word ke memori PLC M221. Pengiriman data membutuhkan waktu sebesar 0,009185 detik dan 0,007249 detik untuk data word dan bit. Paket data tidak mengalami perubahan selama pengiriman, dibuktikan dengan Communication Traffic pada Modbus Slave sehingga data yang diterima bersiffat akurat. Data deteksi berhasil mengubah kondisi PLC M221 yang dapat diamati dari lampu indikator relay output. Hal ini dapat menjadi langkah awal untuk proses pengendalian lebih lanjut. Penelitian ini memberikan kontribusi mengenai kemampuan protokol komunikasi Modbus, untuk diintegrasikan pada sistem deteksi berbasis visi computer. Selama jalur komunikasi Modbus hanya digunakan untuk melewatkan hasil deteksi saja, penggunaannya sudah cukup untuk menghubungkan antara perangkat deteksi dan PLC.

REFERENSI

[1] P. R. L. de Almeida, J. H. Alves, R. S. Parpinelli, and J. P. Barddal, "A systematic review on computer vision-based parking lot management applied on public datasets," *Expert Systems with Applications*, vol. 198. Elsevier Ltd, Jul. 15, 2022. doi: 10.1016/j.eswa.2022.116731.

[2] H. M. Ahmad and A. Rahimi, "Deep learning methods for object detection in smart manufacturing: A survey," *J Manuf Syst*, vol. 64, pp. 181–196, 2022.

[3] I.-C. Chen, R.-C. Hwang, and H.-C. Huang, "PCB Defect Detection Based on Deep Learning Algorithm," *Processes*, vol. 11, no. 3, p. 775, Mar. 2023, doi: 10.3390/pr11030775.

[4] B. Wang, F. Tao, X. Fang, C. Liu, Y. Liu, and T. Freiheit, "Smart Manufacturing and Intelligent Manufacturing: A Comparative Review," *Engineering*, vol. 7, no. 6, pp. 738–757, Jun. 2021, doi: 10.1016/j.eng.2020.07.017.

[5] A. Barari, M. de Sales Guerra Tsuzuki, Y. Cohen, and M. Macchi, "Editorial: intelligent manufacturing systems towards industry 4.0 era," *J Intell Manuf*, vol. 32, no. 7, pp. 1793–1796, Oct. 2021, doi: 10.1007/s10845-021-01769-0.

[6] Y. Chen *et al.*, "Application of YOLOv4 Algorithm for Foreign Object Detection on a Belt Conveyor in a Low-Illumination Environment," *Sensors*, vol. 22, no. 18, p. 6851, Sep. 2022, doi: 10.3390/s22186851.

[7] G. Li *et al.*, "A critical review of cyber-physical security for building automation systems," *Annu Rev Control*, vol. 55, pp. 237–254, 2023, doi: 10.1016/j.arcontrol.2023.02.004.

[8] D. Mancheno and S. Gamboa, "Development of an Industrial IoT Gateway Prototype," 2022, pp. 61–75. doi: 10.1007/978-3-031-08942-8_5.

[9] J. Wang *et al.*, "A Modeling and Verification Method of Modbus TCP/IP Protocol," 2022, pp. 527–539. doi: 10.1007/978-3-030-95391-1_33.

[10] A. Gumaei *et al.*, "A robust cyberattack detection approach using optimal features of SCADA power systems in smart grids," *Appl Soft Comput*, vol. 96, p. 106658, Nov. 2020, doi: 10.1016/j.asoc.2020.106658.

[11] F. Katulić, D. Sumina, S. Groš, and I. Erceg, "Protecting Modbus/TCP-Based Industrial Automation and Control Systems Using Message Authentication Codes," *IEEE Access*, vol. 11, pp. 47007–47023, 2023, doi: 10.1109/ACCESS.2023.3275443.