https://jurnaleeccis.ub.ac.id/
p-*ISSN : 1978-3345, e-ISSN(Online): 2460-8122*

Jurnal EECCIS Vol. 14, No. 1, April 2020
pp 35-40

Komparasi Protokol Komunikasi pada Sistem Produksi Siber-Fisik berbasis IEC 61499

Rico D. Aryandaru[1], Awang N.I. Wardana[2], dan Agus Arif[3]
[1,2,3] Departemen Teknik Nuklir dan Teknik Fisika, Fakultas Teknik, Universitas Gadjah Mada
E-mail: awang.wardana@ugm.ac.id

***Abstract*— The change in the concept of an automation pyramid into an automation cloud in a cyber-physical production system makes data communication no longer stratified but can be done directly between devices. Based on IEC 61499, which defines the function blocks for building such communications, communication protocols can be run on various devices. Several communication protocols that can fulfill these requirements are OPC-UA, FBDK / IP, and MQTT. The research was conducted by comparing the three communication protocols for latency parameters and their jitters. The test method used to compare latency parameters is the variance analysis test and the Tukey test. The jitter value of the protocols are compared to the standard deviation parameter. The test results showed that the MQTT communication protocol had a faster latency value, with a 95% confidence level. The standard deviation of the variation value for OPC-UA, FBDK / IP, and MQTT showed the jitter value of 0.72 seconds, 0.35 seconds, and 0.64 seconds. Comparing the three communication protocols' standard deviation values showed that the FBDK / IP communication protocol has significantly less jitter than the OPC-UA and MQTT communication protocols.**

***Index Terms*—Cyber-Physical Production System, IEC 61499, communication protocol, latency, jitter.**

***Abstrak*–Perubahan konsep piramida otomasi menjadi awan otomasi di dalam sistem produksi siber-fisik membuat komunikasi data tidak lagi bertingkat, melainkan dapat dilakukan langsung antar perangkat. Berbasis pada IEC 61499 yang mendefinisikan blok fungsi untuk membangun komunikasi tersebut, protokol komunikasi yang dapat dijalankan pada berbagai macam perangkat. Beberapa protokol komunikasi yang dapat memenuhi hal tersebut adalah OPC-UA, FBDK/IP, dan MQTT. Penelitian ini dilakukan dengan membandingkan ketiga protokol komunikasi tersebut untuk parameter latensi dan variasi keterlambatannya. Metode uji yang digunakan untuk membandingkan parameter latensi adalah uji analisis variansi dan uji kontras. Parameter variasi keterlambatan dibandingkan parameter standar deviasinya. Hasil uji menunjukan bahwa protokol komunikasi MQTT yang lebih cepat pada nilai latensinya dengan tingkat kepercayaan 95%. Untuk perbandingan nilai variasi keterlambatan, standar deviasi nilai variasi keterlambatan untuk OPC-UA, FBDK/IP dan MQTT menunjukan nilai 0,72 detik, 0,35 detik dan 0,64 detik. Perbandingan nilai standar deviasi dari ketiga protokol komunikasi tersebut menunjukkan bahwa protokol komunikasi FBDK/IP memiliki nilai variasi keterlambatan yang lebih kecil dibandingkan dengan protokol komunikasi OPC-UA dan MQTT.**

***Kata Kunci*—Sistem produksi siber-fisik, IEC 61499, protokol komunikasi, latensi, variasi keterlambatan.**

I. PENDAHULUAN

Revolusi industri keempat atau yang dikenal dengan istilah industri 4.0 muncul karena adanya pergerakan dari pasar yang menuntut industri untuk memiliki fleksibilitas yang tinggi [1]. Salah satu komponen penyusun utama dari industri 4.0 adalah sistem produksi siber-fisik/ *cyber-physical production system*. Karakteristik dari sistem tersebut adalah komunikasi data antar perangkat tidak lagi dilakukan secara bertingkat, melainkan setiap perangkat mempunyai kesempatan untuk berkomunikasi secara langsung dengan perangkat lainnya [2]. Oleh karena itu, Sistem produksi siber-fisik tidak lagi menggunakan konsep pada industri yang terdahulu yaitu piramida otomasi (*automation pyramid*) melainkan awan otomasi (*automation cloud*).

Salah satu cara untuk membangun komunikasi awan otomasi adalah dengan menggunakan konsep yang menyertakan fungsi untuk melakukan komunikasi. IEC 61499 merupakan standar yang dapat digunakan untuk mendefinisikan fungsi tersebut dengan menggunakan bahasa berupa blok fungsi [3]. Selain itu, dibutuhkan juga protokol komunikasi yang mendukung *interoperability* sehingga dapat digunakan pada berbagai macam perangkat. Pada IEC 61499, persyaratan terhadap protokol komunikasi dalam mendukung *interoperability* sudah disertakan sampai lapisan paling atas yaitu *application layer* [4]. Beberapa protokol yang dapat mematuhi persyaratan tersebut adalah OPC-UA (*Open Platform Communication – Unified Architecture*) [5], FBDK/IP (*Function Block Development Kit/Internet Protocol*) [3] dan MQTT (MQ *Telemetry Transport*) [6].

Dalam membangun komunikasi OPC-UA, lapisan aplikasi dapat dirancang dengan menggunakan skema *server-client.* Skema tersebut memiliki kelebihan dimana *client* dapat mengirimkan argument keluaran dan menerima argument masukan pada waktu yang bersamaan [7]. Selanjutnya protokol komunikasi FBDK/IP memiliki fleksibilitas yang tinggi karena menggunakan skema *publish-subscribe* dan alamat *multicast* [8]. Di sisi lain, MQTT memiliki kelebihan yaitu sifatnya yang berupa data agnostik. Oleh karena itu, data yang dikirim tidak lagi memerlukan konversi khusus untuk dapat diterima [9]. Implementasi protokol komunikasi ke dalam sistem produksi siber-fisik dilakukan dengan perangkat lunak berbasis IEC 61499 [10].

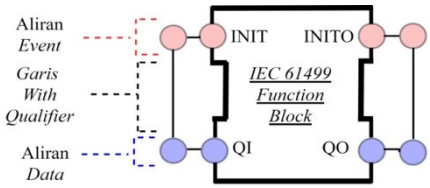
Manuscript submitted on March 2020, accepted and published on April 2020

Pada penelitian yang dituangkan pada makalah ini, ketiga protokol komunikasi tersebut dikomparasikan untuk mengetahui pengaruhnya terhadap sistem produksi siber-fisik [11].

II. LANDASAN TEORI

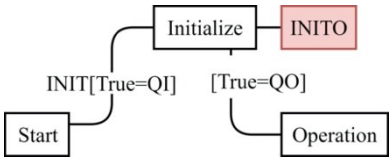
A. Konsep Pemrograman IEC 61499

IEC 61499 mendefinisikan 3 (tiga) tipe blok fungsi (*function block*) antara lain blok fungsi dasar, blok fungsi komposit dan blok fungsi antarmuka servis [3]. Karakteristik blok fungsi pada IEC 61499 (Gambar 1) adalah pemisahan antara aliran data dengan aliran kejadian (*event*). *Event* masukan (INIT) dan keluaran (INITO) diletakkan pada bagian atas dari blok fungsi, sedangkan data masukan (QI) dan keluaran (QO) diletakkan pada bagian bawah. Aliran data dan *event* dihubungkan melalui sebuah garis bernama *with-qualifier*. Hubungan tersebut menunjukkan kapan aliran data akan dibaca. Pada blok fungsi yang ditunjukan pada Gambar. 1, aliran data QI akan terbaca saat aliran *event* INIT menerima sebuah masukan *event*.



Gambar 1. Struktur diagram untuk *blok fungsi* pada IEC 61499 [3].

Perilaku dari blok fungsi (Gambar. 1) dapat dilihat melalui suatu *execution control chart* (Gambar. 2). Saat INIT menerima sebuah masukan *event*, maka aliran data QI akan terbaca. Kala data yang terbaca berupa nilai *true,* maka *blok fungsi* akan berubah keadaan menuju *Initialize* dan mengeluarkan event pada keluaran INITO serta nilai *true* pada keluaran QO.



Gambar. 2 *Execution control chart* pada *blok fungsi* di IEC-61499 [3].

B. Protokol Komunikasi

Protokol komunikasi merupakan sekumpulan aturan yang digunakan untuk membentuk komunikasi data yang dapat dipahami oleh pengirim dan penerima data [2]. Protokol komunikasi dapat digambarkan melalui kerangka *Open System Interconnection* (*OSI*). Kerangka tersebut menjelaskan bahwa komunikasi data dibangun dalam 7 lapisan dimana setiap lapisan memiliki tugas yang spesifik.

Pada konsep pemrograman IEC 61499, semua lapisan tersebut harus memenuhi aturan dasar yang sudah tertuang dalam IEC 61499 *Compliance Profile for Feasibility Demonstrations* [4]. Peraturan tersebut mendefinisikan bahwa setiap protokol komunikasi harus mampu dibangun di atas lapisan *ethernet* dan *internet protocol*. Saat berada di atas kedua lapisan tersebut, maka protokol komunikasi mendukung *interoperability* dan dapat digunakan sebagai basis protokol pada konsep *automation cloud.* Beberapa protokol yang memenuhi peraturan tersebut adalah OPC-UA, FBDK/IP dan MQTT. Ketiga protokol tersebut mempunyai karakteristik lapisan (Tabel I) tersendiri di bagian *transport*, *session*, *presentation* dan *application*.

TABEL I
LAPISAN PROTOKOL KOMUNIKASI OPC-UA, FBDK/IP DAN MQTT.

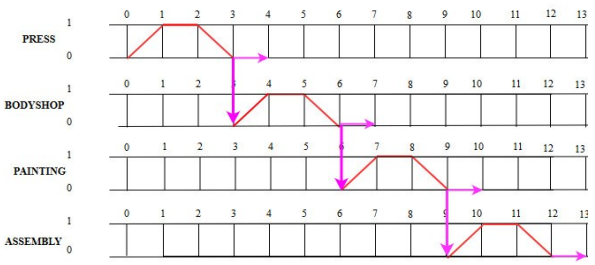
Lapisan	OPC-UA	FBDK/IP	MQTT
Physical	*Wireless*	*Wireless*	*Wireless*
Data Link	*Ethernet*	*Ethernet*	*Ethernet*
Network	*Internet Protocol*	*Internet Protocol*	*Internet Protocol*
Transport	*Transmission Control Protocol*	*User Datagram Protocol*	*Transmission Control Protocol*
Session	*opc_ua*		*mqtt*
Presentation	*Binary*	*Binary*	*Binary*
Application	*Server-Client*	*Publish-Subscribe*	*Publish-Subscribe*

III. METODOLOGI PENELITIAN

A. Implementasi sistem produksi siber-fisik

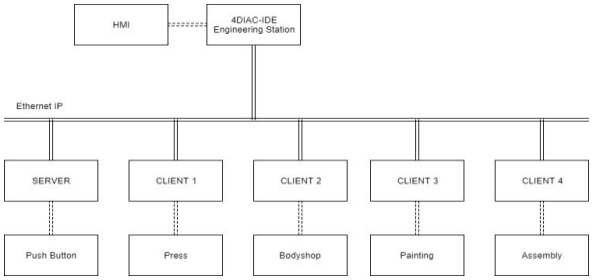
Studi kasus yang diimplementasikan merupakan proses yang disesuaikan dengan skenario awan otomasi pada sistem produksi siber-fisik [11]. Studi kasus yang digunakan terdiri dari 4 buah proses produksi yang dilakukan secara berurutan dan memiliki waktu kerja selama 2 detik. Untuk membangun komunikasi antar perangkat dalam studi kasus tersebut, digunakan protokol komunikasi OPC-UA, FBDK/IP dan MQTT. Pada penelitian ini, ketiga protokol komunikasi OPC-UA, FBDK/IP dan MQTT bertindak sebagai faktor. Selanjutnya, parameter berupa latensi (*latency*) dan variasi keterlambatan (*jitter).* Implementasi dilakukan dengan menggunakan perangkat lunak berbasis IEC 61499 yaitu 4DIAC-IDE [10] dan perangkat keras Raspberry Pi. Perangkat Raspberry Pi dipilih karena kapabilitasnya untuk ditanam berbagai macam *library*.

Studi kasus yang digunakan dalam penelitian adalah proses pembuatan mobil dengan jenis *Sport Utility Vehicle* (SUV) dan *Multi Purpose Vehicle* (MPV) yang terdiri dari proses *press, bodyshop, painting* dan *assembly*.



Gambar. 3. Studi kasus dalam *displacement step diagram* [11].

Skenario tersebut digambarkan dalam bentuk *displacement step diagram* yang ditunjukan pada Gambar. 3. Sumbu horizontal menggambarkan langkah kerja mesin dan sumbu vertikal menggambarkan keadaan mesin. Nilai 0 menunjukkan bahwa mesin sedang tidak berjalan, sedangkan nilai 1 menunjukkan bahwa mesin sedang berjalan. Setiap proses memiliki 3 langkah kerja antara lain transisi menuju keadaan bekerja, keadaan bekerja dan transisi menuju keadaan tidak bekerja. Garis panah berwarna ungu menunjukkan bahwa proses yang sebelumnya akan mempengaruhi proses selanjutnya. Sebagai contoh, proses *bodyshop* akan dimulai ketika proses *press* sudah selesai. Keempat proses masing-masing dipetakan pada perangkat keras Raspberry Pi (Gambar 4). Proses akan diinisiasi melalui sebuah masukan dari *push button* dan *human machine interface* (HMI).



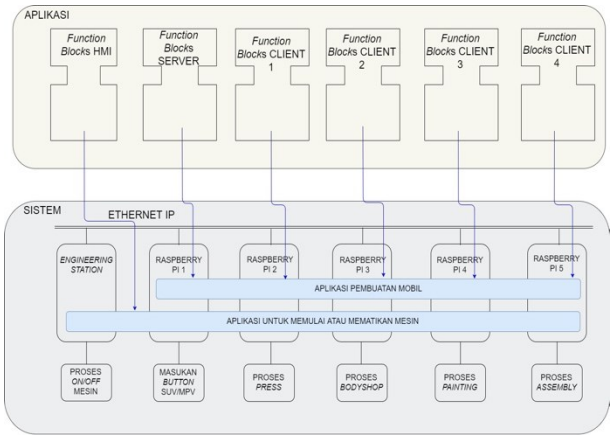
Gambar 4. Pemetaan proses terhadap perangkat keras.

B. Pengambilan dan Analisis Data

Data latensi diperoleh dari pengurangan waktu total yang terjadi saat proses hendak diinisiasi hingga proses terakhir mulai bekerja terhadap waktu ideal proses. Pengambilan data latensi dilakukan dengan menggunakan osiloskop. Setelah diperoleh data latensi, maka digunakan uji analisis variansi untuk mengetahui pengaruh dari perbedaan protokol komunikasi terhadap parameter latensi. Langkah selanjutnya adalah melakukan uji Tukey. Hasil dari uji tersebut akan menunjukkan protokol komunikasi yang berbeda signifikan dengan protokol komunikasi lainnya. Selanjutnya, data variasi keterlambatan diperoleh melalui nilai standar deviasi dari statistik deskriptif terhadap data latensi. Setelah diperoleh data variasi keterlambatan, maka digunakan analisis perbandingan untuk mengetahui protokol komunikasi dengan variasi keterlambatan yang berbeda signifikan. Seluruh uji statistik dilakukan dengan perangkat lunak Minitab [12] dan dengan tingkat signifikansi sebesar 0.05.

IV. HASIL DAN PEMBAHASAN

Garis besar sistem produksi siber-fisik yang menjadi studi kasus digambarkan pada Gambar 5 terdiri dari enam aplikasi blok-blok fungsi yaitu *HMI, SERVER, CLIENT 1, CLIENT 2, CLIENT 3* dan *CLIENT 4*. Aplikasi blok-blok fungsi *HMI* memiliki fungsi untuk menyalakan mesin (*ON*) dan mematikan mesin (OFF) secara keseluruhan. Aplikasi blok-blok fungsi *SERVER, CLIENT 1, CLIENT 2, CLIENT 3* dan *CLIENT 4* akan berintegrasi membentuk suatu fungsi gabungan berupa proses pembuatan mobil. Kelima aplikasi tersebut akan dipetakan masing-masing ke dalam sebuah Raspberry Pi yang bertindak sebagai *controller.* Secara spesifik, Aplikasi blok-blok fungsi *SERVER* berfungsi untuk menerima masukan yang akan menentukan apakah kerangka mobil yang dibuat merupakan tipe *Multi Purposes Vehicle* (MPV) atau *Sport Utility Vehicle* (SUV). Hasil dari *Blok-blok fungsi SERVER* akan dikirim menuju ke keempat *CLIENT 1, CLIENT 2, CLIENT 3* dan *CLIENT 4* dengan tujuan untuk mengetahui apakah proses menginginkan mobil dengan jenis MPV atau SUV.



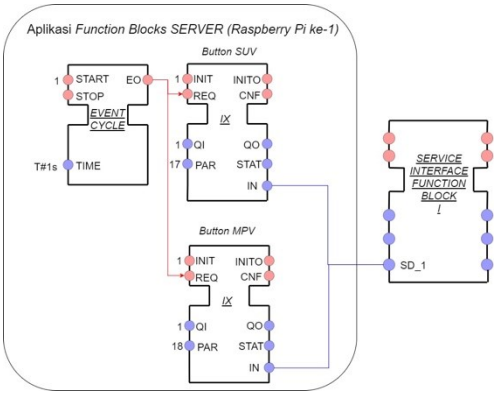
Gambar 5. Garis besar proses kontrol yang diimplementasikan pada IEC 61499.

Aplikasi di dalam Raspberry Pi ke-1 (Gambar 6) terdiri dari aplikasi blok-blok fungsi SERVER dan sebuah blok fungsi antar muka servis. Aplikasi blok-blok fungsi *SERVER* terdiri dari *EVENT CYCLE* dan dua *Basic Blok-blok fungsi* IX (SUV dan MPV). *EVENT CYCLE* memiliki tugas untuk mengeluarkan *event* agar blok fungsi dasar IX melakukan pembacaan nilai IN setiap satu detik. blok fungsi dasar IX (SUV dan MPV) merupakan blok fungsi dasar yang dipetakan khusus ke dalam I/O Raspberry Pi 1 dan dihubungkan ke suatu *push button*. Proses pembuatan mobil dimulai saat keluaran blok fungsi dasar *IX* SUV atau MPV mempunyai nilai *TRUE* pada *output IN*. Sebagai contoh dalam pembuatan kerangka SUV, maka SUV akan mengirimkan nilai *TRUE* tersebut ke dalam *AND*. Hasil dari pengolahan logika di dalam *AND* tersebut akan menghasilkan nilai *TRUE*. Untuk mengubah proses menjadi mobil MPV, maka MPV harus mempunyai nilai *TRUE* pada *output IN.* Pada jalur MPV terdapat F_NOT yang akan mengubah nilai *TRUE* menjadi *FALSE.* Selanjutnya hasil dari pengolahan logika di dalam *AND* tersebut akan menghasilkan nilai *FALSE.* Kesimpulannya adalah hasil *TRUE* pada *AND* akan menghasilkan kerangka SUV, sedangkan hasil *FALSE* pada *AND* akan menghasilkan kerangka MPV.

Antarmuka servis *I* pada Raspberry Pi ke-1 akan mengirimkan data berupa SUV atau MPV ke *controller* yang bertindak sebagai *client* yaitu Raspberry Pi ke-2

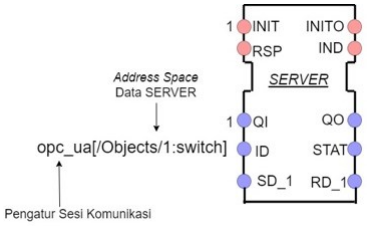
(*Press*), Raspberry Pi ke-3 (*Bodyshop*), Raspberry Pi ke-4 (*Painting*) dan Raspberry Pi ke-5 (*Assembly*).

Protokol yang dibangun dalam sistem memiliki kewajiban untuk memenuhi aturan yang tertuang dalam IEC 61499. Untuk menghubungkan *server* dengan semua *client*, digunakan protokol komunikasi yang berbeda antara lain FBDK/IP, OPC-UA dan MQTT. Dalam penelitian ini, seluruh *message header* dan *payload* data direpresentasikan dalam bentuk *binary*.



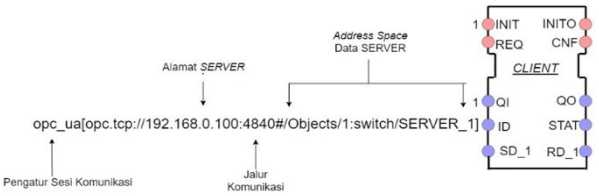
Gambar 6. Aplikasi di dalam Raspberry Pi ke-1 pada studi kasus.

Pada penelitian ini, protokol komunikasi OPC-UA dibangun dengan menggunakan skema *client-server*. Oleh karena itu, objek yang akan dipanggil adalah *method*. Selain itu, protokol komunikasi OPC-UA menggunakan *binary encoding* terhadap semua *message header*. Dalam penulisan di *Service Interface SERVER* dan *CLIENT*, aturan ASN.1 *encoding* dapat dituliskan dengan "opc_ua" pada ID dari . Selain itu penulisan ID juga mencakup alamat *server* berikut dengan *address space*. Selanjutnya protokol komunikasi OPC-UA sudah melakukan standardisasi dengan menetapkan *port* 4840 sebagai jalur komunikasinya. Untuk menggunakan *port* yang lain, maka *port* baru tersebut harus dicantumkan dalam *source code* OPC-UA.



Gambar 7. *Service interface SERVER* OPC-UA.

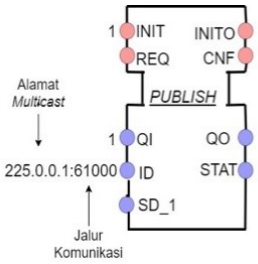
SERVER OPC-UA (Gambar 7) akan membuat sebuah *address space* berupa "*/Objects/1:switch*" ketika ada *event* RSP yang masuk. Di dalam *address space* tersebut, terdapat *method* yang berisi *input argument* (RD_1) dan *output argument* (SD_1). Setelah *address space* terbentuk, maka data yang berada di SD_1 dan RD_1 sudah siap untuk diproses oleh *client*.



Gambar 8. *Service interface CLIENT* OPC-UA.

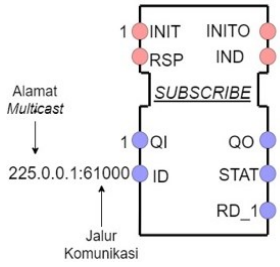
CLIENT OPC-UA (Gambar. 8) akan memulai melakukan pemanggilan *method* yang berada di dalam *address space SERVER*. *Event* IND mengindikasikan bahwa data yang berasal dari alamat 192.168.0.100 dan port 4840 telah berhasil diterima. Ada satu hal menarik dari implementasi protokol komunikasi OPC-UA di dalam sistem dengan basis IEC 61499. Protokol komunikasi OPC-UA hanya dapat menggunakan satu *port* atau jalur data untuk setiap komunikasi yang dibangun.

Selanjutnya, protokol komunikasi FBDK/IP dapat dibangun di atas lapisan User Datagram Protocol. Oleh karena itu, skema yang dapat digunakan adalah *publish-subscribe.* Saat berdiri di atas lapisan U*ser Datagram Protocol*, transmisi data akan dikirim ke alamat *multicast*. Protokol komunikasi FBDK/IP merupakan protokol *default* dari IEC 61499 sehingga penulisan publish dan subscribe hanya mencakup *alamat* dan *port* yang akan dituju. Alamat 225.0.0.1 merupakan salah satu alamat yang dapat digunakan sebagai alamat *multicast*. Selanjutnya *port* 61002 merupakan salah satu *port* dalam Raspberry Pi yang masih tersedia, sehingga dapat diimplementasikan untuk komunikasi data.



Gambar 9. *Service interface PUBLISH* FBDK/IP.

PUBLISH FBDK/IP (Gambar 9) hanya akan mengirim data ketika ada *event* REQ yang masuk. Saat event tersebut datang, maka otomatis data yang berada di *input* SD_1 akan ditransmisi ke alamat 225.0.0.1 dan *port* 61002.



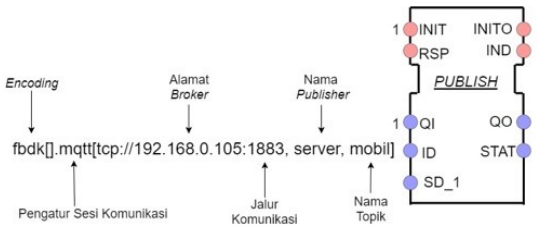
Gambar 10. *Service interface SUBSCRIBE* FBDK/IP.

SUBSCRIBE FBDK/IP (Gambar 10) akan berhasil menerima data ketika ada *event* IND yang keluar. Event

IND mengindikasikan bahwa data yang berasal dari alamat 225.0.0.1 dan *port* 61002 telah berhasil diterima.

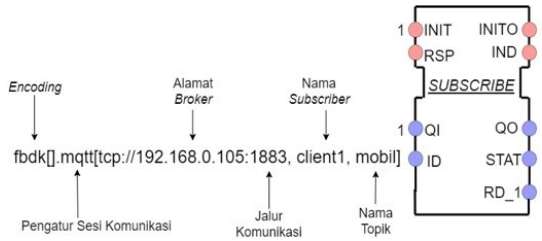
Selanjutnya, protokol komunikasi MQTT dapat dibangun dengan menggunakan skema *publish-subscribe.* Meskipun menggunakan skema tersebut, protokol komunikasi MQTT berdiri di atas lapisan *Transmission Control Protocol*. Tingkat QoS MQTT diatur pada angka satu sehingga menyaratkan bahwa pesan diterima setidaknya satu kali. Protokol komunikasi MQTT tidak mendefinisikan jenis *encoding* yang digunakan untuk data *payload*. Untuk mendukung IEC 61499, maka *encoding* tersebut dapat menggunakan aturan *binary encoding* yang tertera di dalam ASN.1 *encoding*. Dalam penulisan di antarmuka servis, aturan ASN.1 *encoding* dapat dituliskan dengan "fbdk[]" pada ID dari . Selain itu penulisan ID juga mencakup alamat *broker* yang dituju berikut dengan nama dari *publisher* atau *subscriber* serta topik yang akan di *publish* atau *subscribe*. Selanjutnya protokol komunikasi MQTT sudah melakukan standardisasi dengan menetapkan port 1883 sebagai jalur komunikasinya.

PUBLISH MQTT (Gambar 11) hanya akan mengirim data ketika ada *event REQ* yang masuk. Saat *event* tersebut datang, maka otomatis data yang berada di *input* SD_1 akan ditransmisi ke alamat 192.168.0.105 dan *port* 1883. Setiap *publisher* atau *subscriber* dalam komunikasi MQTT harus mempunyai sebuah nama dan topik. Sebagai contoh, *PUBLISH* (Gambar 7) memiliki nama sebagai *server* dan topik bernama mobil.



Gambar 11. *Service interface PUBLISH* MQTT.

SUBSCRIBE MQTT (Gambar 12) akan berhasil menerima data ketika ada *event* IND yang keluar. Event IND mengindikasikan bahwa data yang berasal dari alamat 192.168.0.105 dan port 1883 telah berhasil diterima.



Gambar 12. *Service interface SUBSCRIBE* MQTT.

Pada pengambilan data latensi, kondisi yang akan dipantau adalah kondisi pada saat *push button* ditekan dan kondisi pada saat *client* keempat mulai bekerja. Setelah dilakukan pengambilan data dengan menggunakan osiloskop, diperoleh data latensi seperti Tabel II.

TABEL II
DATA LATENSI UNTUK PROTOKOL OPC-UA, FBDK/IP DAN MQTT.

Percobaan ke-	*OPC-UA (detik)*	*FBDK/IP (detik)*	*MQTT (detik)*
1	3,6	2,2	1,0
2	4,2	2,6	1,4
3	2,6	3,0	0,4
4	3,8	3,0	0,4
5	2,0	2,6	1,0
6	3,8	2,6	1,8
7	2,6	3,0	1,8
8	2,4	2,2	1,8
9	3,4	2,2	2,2
10	3,0	3,0	2,0

TABEL III
TABEL HASIL UJI ANALISIS VARIANSI.

Source	*DF*	*Adj SS*	*Adj MS*	*F-Value*	*P-Value*
Protokol	2	16.451	8.2253	23.08	0.000
Error	27	9.624	0.3564		
Total	29	26.075			

Hasil uji analisis variansi (Tabel III) untuk data latensi menunjukkan nilai *p-value* yang lebih kecil dari tingkat signifikansi 0,05, sehingga dapat dikatakan bahwa perbedaan protokol komunikasi akan mempengaruhi latensi dari sistem. Akan tetapi, hasil uji tersebut belum memberikan indikasi protokol mana yang paling berpengaruh atau berbeda. Oleh karena itu, dilakukan uji kontras untuk mengetahui protokol komunikasi yang mempunyai latensi yang paling cepat.

TABEL IV
TABEL HASIL UJI TUKEY DENGAN TINGKAT KEPERCAYAAN 95%.

Protokol	N	*Mean*	*Grouping*
OPC-UA	10	3.140	A
FBDK/IP	10	2.640	A
MQTT	10	1.380	B

Hasil uji Tukey (Tabel IV) menunjukan bahwa protokol komunikasi MQTT merupakan protokol komunikasi yang memiliki nilai latensi signifikan lebih cepat dibandingkan protokol komunikasi OPC-UA dan FBDK/IP dengan tingkat kepercayaan 95%. Selanjutnya hasil analisis statistik deskriptif data latensi secara berturut-turut menunjukkan nilai latensi pada protokol komunikasi OPC-UA, FBDK/IP dan MQTT adalah 3,14 ± 0,72 detik, 2,64 ± 0,35 detik dan 1,38 ± 0,64 detik. Standar deviasi dari masing-masing latensi merupakan nilai variasi keterlambatan untuk masing-masing protokol komunikasi. Perbandingan ketiga nilai tersebut menunjukan protokol komunikasi FBDK/IP memiliki nilai variasi keterlambatan yang lebih kecil dibandingkan dengan protokol komunikasi OPC-UA dan MQTT.

V. KESIMPULAN

Komparasi protokol komunikasi OPC-UA, FBDK/IP dan MQTT yang dilakukan dengan uji statistik menunjukan hasil bahwa perbedaan protokol komunikasi memberikan pengaruh terhadap nilai dari parameter latensi. Melalui uji analisis variansi dan uji Tukey, diketahui bahwa protokol komunikasi MQTT merupakan protokol komunikasi dengan nilai latensi yang lebih cepat secara signifikan dibandingkan dengan OPC-UA dan FBDK/IP dengan tingkat kepercayaan 95%. Untuk nilai variansi keterlambatan pada protokol komunikasi OPC-UA, FBDK/IP dan MQTT menunjukan nilai 0,72 detik, 0,35 detik dan 0,64 detik. Hal tersebut menunjukkan bahwa protokol komunikasi FBDK/IP memiliki nilai variasi keterlambatan yang lebih kecil dibandingkan dengan protokol komunikasi OPC-UA dan MQTT.

REFERENSI

[1] K. Schwab, *The Fourth Industrial Revolution*, London: Penguin, 2017.

[2] J. Jasperneite dan M. Wollschlaeger, "The future of industrial communication," *IEEE Industrial Electronics Magazine*, vol. 11, hal. 17-27, Mar. 2017.

[3] A. Zoitl dan R. Lewis, *Modelling Control Systems Using IEC 61499*, London: Institution of Engineering and Technology, 2014

[4] *IEC 61499 compliance profile for feasibility demonstrations* (2019), Holobloc [Daring]. Tersedia: http://holobloc.com/.

[5] W. Mahnke dan L. Stefan-Helmut, *OPC Unified Architecture*, Berlin: Springer Science & Business Media, 2009.

[6] Organization for the Advancement of Structured Information Standards, *MQTT 3.1.1*, OASIS, 2014.

[7] A. Zoitl dan W. Monika, "Towards an industry 4.0 compliant control software architecture using IEC 61499 & OPC UA," pada *22nd IEEE International Conference on Emerging Technologies and Factory Automation*, 2017.

[8] A. Zoitl dan S. Andreas, "Real-Time communication for IEC 61499 in switched Ethernet network," pada *International Congress on Ultra Modern Telecommunications and Control Systems*, 2010.

[9] L. Valima, "*MQTT client implementation in IEC 61131-3 compatible programming environment*," Tampere University of Technology, Finland, 2017.

[10] T. Strasser, M. Rooker, G. Ebenhofer, A. Zoitl, C. Sunder, A. Valentini dan A. Martel, "Framework for distributed industrial automation and control," pada *IEEE International Conference on Industrial Informatics*, 2008.

[11] R. D. Aryandaru, "*Analisis pengaruh protokol komunikasi OPC-UA, FBDK/IP dan MQTT terhadap nilai Latency dan Jitter pada sistem berbasis IEC 61499*", Tugas Akhir, Universitas Gadjah Mada, 2018.

[12] *Minitab Expert Support* (2020), Minitab, [Daring]. Tersedia: https://support.minitab.com.