RANCANG BANGUN INTERNET OF THINGS SERVER DENGAN MENGGUNAKAN ACTIVEMQ ARTEMIS UNTUK MITIGASI BENCANA DI RASPBERRY PI 3

Sadr Lufti Mufreni, and Esi Putri Silmina[*)]

Jurusan Teknologi Informasi, Universitas 'Aisyiyah Yogyakarta
Jalan Siliwangi (Ring Road Barat) 63, Mlangi, Nogotirto, Gamping, Sleman,
D.I. Yogyakarta, 55292, Indonesia

[*)]*E-mail: sadr@unisayogya.ac.id*

Abstrak

Indonesia merupakan negara kepulauan yang mempunyai lebih dari 13.000 pulau. Wilayahnya terletak di antara Samudera Hindia dan Samudera Pasifik dan dilewati oleh Pacific Ring of Fire sehingga banyak gunung berapi aktif. Berdasarkan letak geografis mempunyai potensi tsunami dan gempa bumi cukup tinggi. Diperlukan rencana penanggulangan bencana yang baik untuk menekan risiko yang bisa terjadi, salah satunya dengan mitigasi bencana. Mitigasi bencana adalah serangkaian upaya untuk mengurangi risiko bencana, baik melalui pembangunan fisik maupun penyadaran dan peningkatan kemampuan menghadapi ancaman bencana. Mitigasi bencana diperlukan untuk mengurangi dampak yang ditimbulkan terutama korban jiwa. Salah satunya dengan menggunakan sistem peringatan dini. Sistem peringatan dini terdiri dari 3 komponen utama yaitu sensor untuk mendapatkan nilai dari suatu lingkungan, controller untuk mengolah nilai yang diterima, dan aksi yang dilakukan berdasarkan hasil dari pengolahan. Untuk membuat sistem yang efektif diperlukan komunikasi yang memadai. Messaging queue digunakan oleh industri untuk komunikasi antar perangkat lunak, perangkat keras, dan embedded system. Penelitian berfokus pada penggunaan ActiveMQ Artemis sebagai messaging queue sebagai server untuk komunikasi dengan internet of things (IoT). Keunggulan ActiveMQ Artemis dapat dijalankan di Raspberry Pi 3 dengan sedikit modifikasi. Hasil penelitian membuktikan bahwa ActiveMQ Artemis dapat digunakan untuk komunikasi IoT pada simulasi sistem mitigasi bencana.

Kata kunci: internet of things (iot); messaging queue; sistem peringatan dini; monitoring;

Abstract

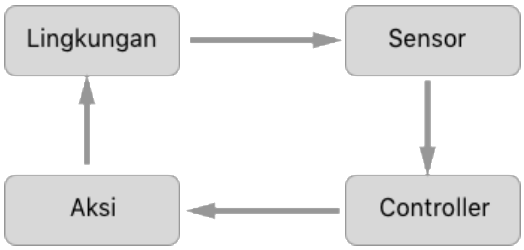
Indonesia is an archipelago that has more than 13,000 islands. Its territory is located between the Indian Ocean and the Pacific Ocean and traversed by the Pacific Ring of Fire having many volcanoes are active and prone to earthquakes. Based on its geographical location it has a high potential for tsunamis and earthquakes. A good disaster management plan is needed to reduce the risks that can occur, one of which is disaster mitigation. Disaster mitigation is a series of efforts to reduce disaster risk, both through physical development and awareness raising and capacity to face the threat of disaster. Disaster mitigation is needed to reduce the impact caused especially fatalities. One of them is by using an early warning system. The early warning system consists of 3 main components, namely the sensor to get the value of an environment, the controller / server to process the value received, and the actions carried out based on the results of processing. To make an effective system requires adequate communication. Messaging queues are used by industry for communication between software, hardware, and embedded systems. This research uses ActiveMQ Artemis as a messaging queue server for communication with the internet of things (IoT). The benefits of Artemis ActiveMQ can be run on Raspberry Pi 3 with a few modifications. The research results prove that Artemis ActiveMQ can be used for IoT communication on disaster mitigation system simulation.

Keywords internet of things (iot); messaging queue; early warning system; monitoring;

1. Pendahuluan

Indonesia merupakan negara kepulauan yang mempunyai lebih dari 13.000 pulau. Wilayahnya terletak di antara Samudera Hindia dan Samudera Pasifik dan dilewati oleh Pacific Ring of Fire sehingga banyak gunung berapi aktif dan rawan terhadap gempa bumi dan tsunami. Mitigasi adalah serangkaian upaya untuk mengurangi risiko bencana, baik melalui pembangunan fisik maupun penyadaran dan peningkatan kemampuan menghadapi ancaman bencana [1]. Mitigasi bencana diperlukan untuk mengurangi dampak yang ditimbulkan terutama korban jiwa. Salah satunya dengan menggunakan sistem peringatan dini.

Sistem peringatan dini terdiri dari tiga komponen utama yaitu sensor untuk mendapatkan nilai dari suatu lingkungan, controller untuk mengolah nilai yang diterima, dan aksi yang dilakukan berdasarkan hasil dari pengolahan (Gambar 1). Untuk membuat sistem yang efektif diperlukan komunikasi yang memadai.



Gambar 1. Interaksi Tiga Komponen Sistem dan Lingkungan

Penelitian embedded system sebelumnya menggunakan web service untuk komunikasi antar komponen [2] dan penggunaan ActiveMQ Classic sebagai printer server [3]. Web service menerima nilai dari sensor, mengolah, dan langsung memberikan respons. Proses dilakukan secara sinkron. Proses ini digunakan karena sensor dan aksi berada pada perangkat yang sama dan sistem memerlukan interaksi secara langsung dengan controller.

Sistem peringatan dini umumnya mempunyai karakteristik sebagai berikut

a. Perangkat sensor dan aksi pada sistem peringatan dini pada umumnya terpisah.
b. Proses pengolahan data dilakukan secara asynchronous.
c. Perangkat sensor merupakan embedded system dengan sumber daya yang sangat terbatas.

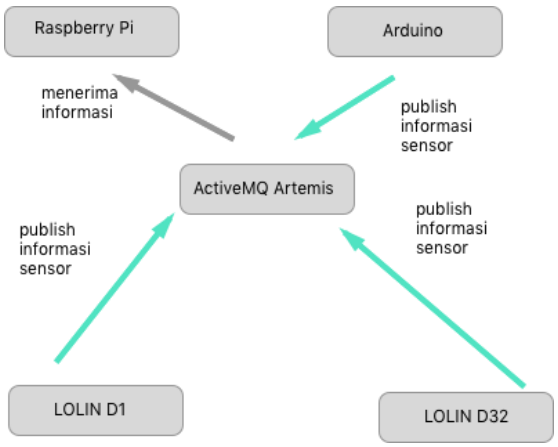
Dengan tiga karakter diatas, peneliti memutuskan untuk menggunakan messaging queue sebagai standar komunikasi. Messaging queue digunakan oleh industri untuk komunikasi antar perangkat lunak, perangkat keras, dan embedded system. Penggunaan messaging queue memungkinkan proses penerimaan nilai, dan pengolahan nilai dilakukan oleh controller yang berbeda.

Protokol yang sering digunakan adalah MQTT (Messaging Queuing Telemetry Transport), AMQP (Advanced Message Queueing Protocol), dan ZMTP (Zero Message Trasport Protocol). Penelitian ini akan menggunakan MQTT karena mempunyai data overhead yang kecil dan mudah dimodifikasi [4], digunakan oleh industri [5], cocok dengan kebutuhan IoT [6], dan mendukung komunikasi secara asynchronous [7]. MQTT server yang digunakan adalah ActiveMQ Artemis.

Perkembangan prosesor, dan memori komputer yang semakin kecil dan murah, mendorong kemajuan embedded system. Mendesain dan mengembangkan embedded system menjadi sangat mudah dengan munculnya Arduino. Kemampuan Arduino dapat ditingkatkan dengan memasang alat tambahan yang dinamakan Shield. Didukung oleh komunitas, shield yang ditawarkan sangat bervariasi. Raspberry Pi, embedded system seukuran kartu kredit, yang dapat di-install dengan sistem operasi Linux ditambah komponen input/output yang dapat diprogram menjadikan perangkat yang cocok jika ingin mengkombinasikan kemampuan mikro kontroller dan komputer. Raspberry Pi 3 ini akan digunakan sebagai server dan internet of things (IoT) yang dipakai adalah Arduino. Tipe Arduinonya adalah Mega 2560, LOLIN D1 mini Pro, dan LOLIN D32. Arduno Mega 2560 mempunyai enam input analog, empat belas digital input/output, dan inter integrated circuit (I2C) untuk komunikasi [8] dan memerlukan Ethernet shield untuk berhubungan dengan jaringan. LOLIN D1 mini Pro dan LOLIN D32 berbasis mikrokontroller ESP-8266, selain mempunyai input analog, digital input/output, I2C juga mempunyai kemampuan Wifi untuk komunikasi nirkabel [9].

Tiga komponen utama sistem peringatan dini yang dikembangkan embedded system dan dikombinasikan dengan messaging queue yang berbasis awan terbentuklah Internet of Things (IOT) untuk mitigasi bencana. Penelitian ini bertujuan untuk mewujudkan pembuatan sistem tersebut. Penelitian [10] [11] [12] [13] tidak berfokus kepada mitigasi bencana dan variasi jenis mikro kontroller dan embedded system kurang. Penelitan [14] berfokus kepada sistem banjir saja. Penelitian ini berfokus kepada penggunaan ActiveMQ Artemis di Raspberry Pi sebagai server untuk komunikasi IoT dalam simulasi sistem mitigasi bencana.

2. Analisis Kebutuhan

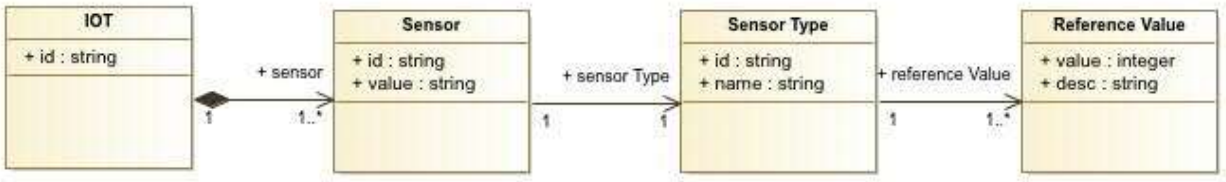


Gambar 2. Interaksi Server dan IoT

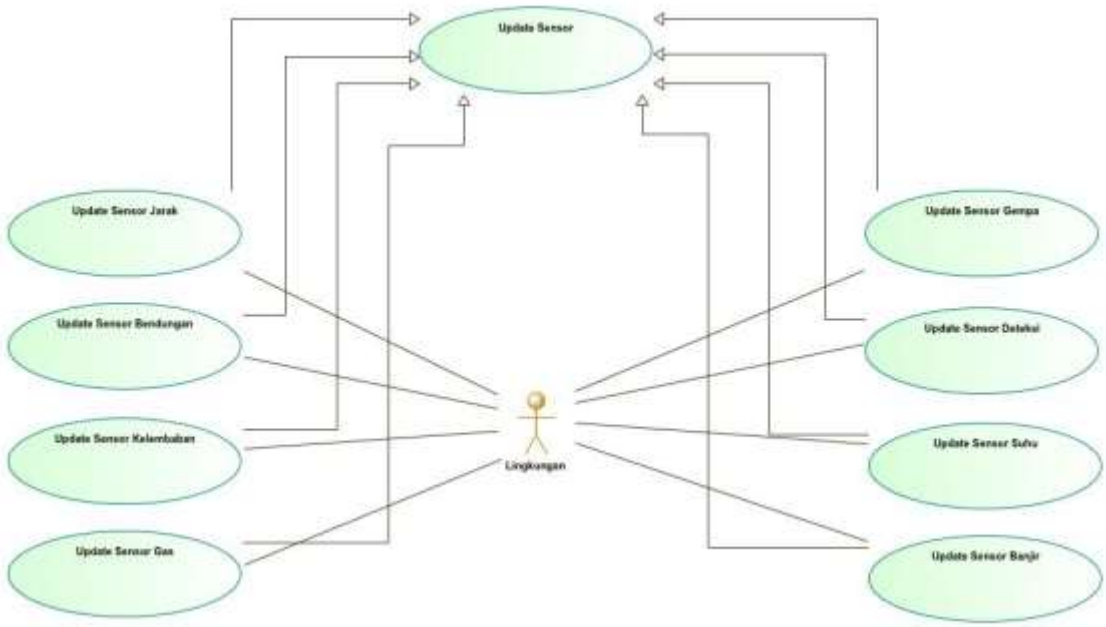
Berikut interaksi beberapa IoT (LOLIN D1, LOLIN D32, Arduino) dengan sistem mitigasi bencana (di Raspberry Pi) menggunakan ActiveMQ Artemis sebagai messaging queue server. IoT akan mengirimkan informasi sensor ke

ActiveMQ Artemis dan sistem mitigasi bencana menerima info sensor kemudian memprosesnya (Gambar 2).

Hasil analisa menggunakan Unified Modelling Language (UML) [15]. Diagram yang digunakan adalah class diagram, use case diagram, dan sequence diagram. Diagram-diagram ini merupakan representasi kinerja sistem dengan menggunakan sensor yang ada. Diagram yang dihasilkan adalah Class Diagram yang menggambarkan model IoT yang digunakan oleh sistem, Use Case Diagram yang menggambarkan interaksi sistem dengan lingkungan, dan Sequence Diagram menggambarkan interaksi di dalam sistem ketika mengirimkan data ke server



Gambar 3. Class Diagram dari IoT dan Sensor



Gambar 4. Use Case Diagram dari Sistem Mitigasi Bencana

2. 1. Class Diagram

Pada sistem ini, setiap IoT dapat memiliki lebih dari satu sensor. Setiap Sensor mempunyai tipe dan nilai referensi. Tipe ada dua yaitu digital dan analog. Tipe digital mempunyai nilai referensi 1 dan 0 dan tipe analog mempunyai nilai referensi yang fleksibel tergantung dari keluaran nilai dari sensornya (Gambar 3). Penelitian ini menggunakan referensi dengan nilai 0-1023 dan 0-4095 untuk tipe analog.

2. 2. Use Case Diagram

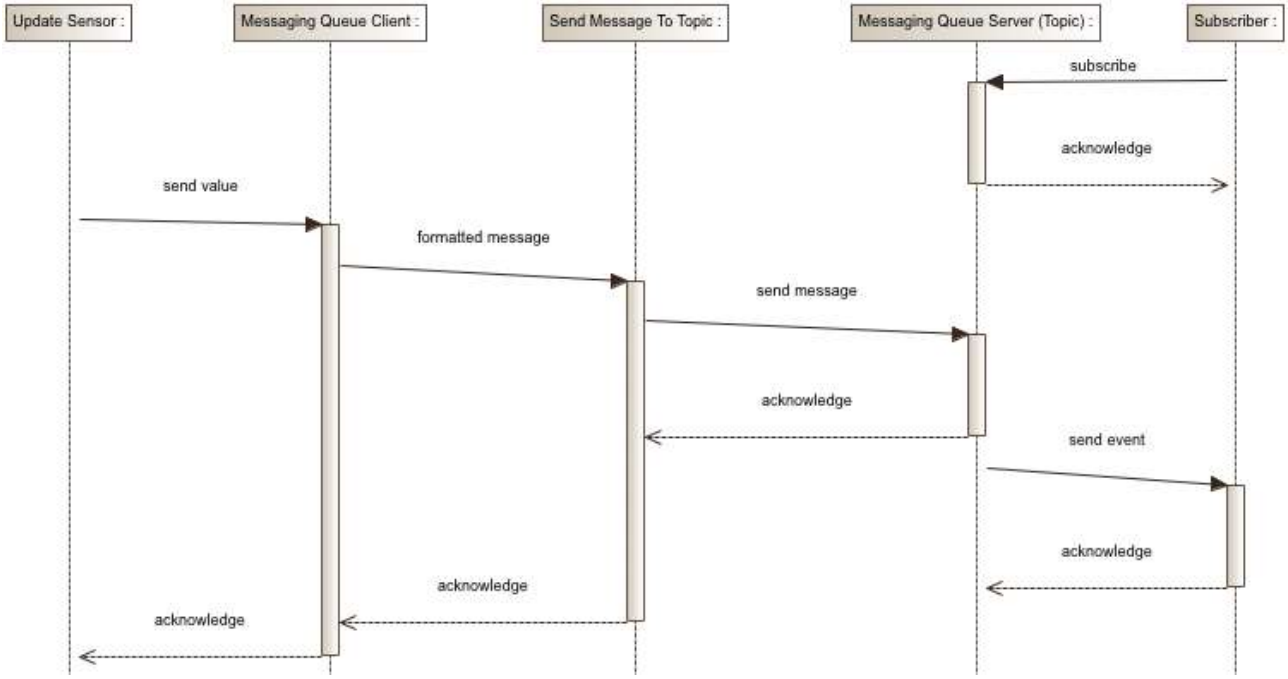
Penelitian ini menggunakan tujuh sensor untuk mensimulasi bencana di dunia nyata. Berikut adalah use case dari sensor-sensor tersebut (Gambar 4):

a. Update sensor jarak: mengukur jarak dengan benda yang di depannya. Sensor ini biasanya di kendaraan.
b. Update sensor bendungan: mengukur ketinggian air, biasanya digunakan untuk mengukur ketinggian air di sungai atau di bendungan.
c. Update sensor kelembaban: mengukur kelembaban udara. Untuk penelitian kali ini disimpan di kendaraan.
d. Update sensor gas: mengukur kadar gas yang mudah terbakar seperti LPG, Butane, Propane, Methane, Alkohol, dan Hydrogen.
e. Update sensor gempa: membaca data dari accelerometer untuk 3 koordinat X, Y, dan Z dan melaporkannya ke server.
f. Update sensor deteksi: memdeteksi ada benda bergerak dalam radius tertentu.
g. Update sensor suhu: mengukur suhu sekitar IoT.

h. Update sensor banjir: mengukur banyaknya air di sebuah lokasi.

2. 3. Sequence Diagram

Semua use case mempunyai algoritma yang sama sehingga sequence diagram dari masing-masing use case juga sama (Gambar 5). Update sensor dilakukan oleh IoT. Dengan menggunakan komunikasi messaging queue, IoT bertindak sebagai client mengirimkan pesan ke server (Messaging Queue Server). Pesan diterima oleh server dan disebarkan kepada subscriber. Subscriber untuk menerima pesan dari server harus mendaftar terlebih dahulu. Penelitian ini menggunakan console monitor sebagai subscriber yang menampilkan data dari semua sensor di IoT.



Gambar 5. Sequence Diagram dari Pengiriman Data Dari IoT sampai Subscriber

3. Implementasi

Sistem yang dibangun menggunakan jaringan kabel dan nirkabel. Penelitian menggunakan jaringan wifi untuk jaringan nirkabelnya. Sensor yang digunakan adalah

a. Sensor rintik hujan.
b. Sensor ketinggian air.
c. Sensor HY-SR05 untuk jarak.
d. Sensor ADXL 345 untuk sensor gempa.
e. Sensor MQ 135 untuk gas.
f. Sensor SHT 30 untuk suhu dan kelembaban.
g. Sensor PIR untuk deteksi gerakan objek.

Untuk komunikasi MQTT, Arduino menggunakan library PubSubClient dan LOLIN menggunakan library EspMQTTClient.
Untuk menggunakan ActiveMQ Artemis diperlukan:

a. Pembuatan broker untuk MQTT. Ini dilakukan sekali untuk satu broker. Setelah pembuatan broker berhasil dilakukan, maka akan dibuat direktori kerja.
b. Menjalankan server ActiveMQ Artemis dari direktori kerja.

Penggunaan ActiveMQ Artemis di Raspberry Pi 3 memerlukan modifikasi untuk menjalankannya. Hal ini diperlukan karena Java Runtime Environment di Raspberry Pi 3 tidak mendukung Garbage First (G1) Collector. Caranya dengan menghapus -XX:+UseG1GC di artemis.profile pada direktori /etc di instaliasi ActiveMQ Artemisnya.

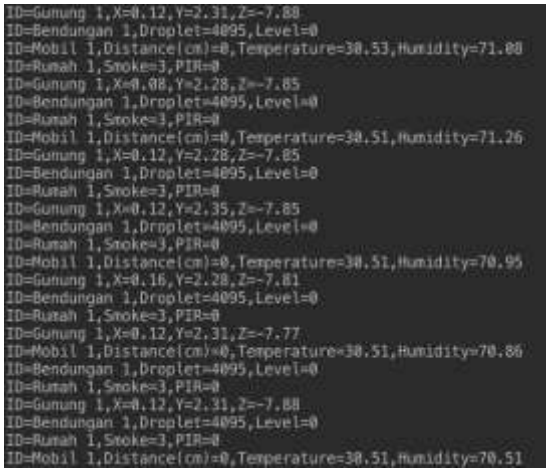
Dibuat program berbasis Java sebagai subscriber. Program ini menggunakan library active-mq untuk berkomunikasi dengan ActiveMQ Artemis dengan menggunakan protokol MQTT. Program ini menampilkan hasil dari bacaan sensor yang dikirim oleh IoT.

4. Hasil dan Analisis

Tes dilakukan dengan cara mengaktifkan empat IoT untuk menjalankan tujuh sensor. Bendungan 1 menggunakan sensor rintik hujan dan ketinggian air, Mobil 1 menggunakan sensor sensor jarak dan gas, Rumah 1 menggunakan sensor suhu, kelembaban, dan deteksi gerakan objek, dan Gunung 1 menggunakan sensor gempa. Data sensor dikirim ke ActiveMQ Artemis di Raspbery Pi 3. Subscriber memonitor data yang diterima oleh ActiveMQ Artemis.

Sebelum menjalankan tes dengan ActiveMQ Artemis, broker harus dibuat terlebih dahulu dengan perintah ./artemis_server/bin/artemis create mqtt, artemis_server adalah direktori ActiveMQ Artemis, create adalah perintah untuk membuat broker, dan mqtt adalah nama broker dan direktori kerja yang akan dibuat. Nama pengguna dan kata kunci akan ditanyakan dalam proses pembuatan broker. Setelah berhasil dibuat modifikasi artemis.profile pada mqtt/etc dan eksekusi perintah ./mqtt/bin/artemis run, argumen run bisa digantikan dengan start jika ingin perintah dieksekusi secara background service. Gambar 6 menampilkan ActiveMQ Artemis berhasil dijalankan. Gambar 7 merupakan data sensor yang dikirimkan oleh IoT. ID=Gunung 1, X=0.12, Y=2.31, Z=-7,88 adalah data dari IoT Gunung 1 dan mengirimkan hasil sensor gempa 0.12 untuk nilai koordinat X, 2.31 untuk nilai koordinat Y, dan -7.88 untuk nilai koordinat Z.

Gambar 6. Log dari ActiveMQ Artemis ketika dijalankan



Gambar 7. Log dari Subscriber yang Berisi Data dari Sensor

Tabel 1. Hasil Pengetesan ActiveMQ Artemis Dengan 4 IoT

ID	Jumlah data yang dikirim	Durasi IoT aktif	Rata-rata jarak antar data
Rumah 1	20851	~5.9 jam	1.026 detik
Gunung 1	8098	~2,39 jam	1.065 detik
Mobil 1	20850	~5.9 jam	1.021 detik
Bendungan 1	20811	~5.8 jam	1.004 detik

Tes dilakukan dengan mengirimkan data sensor kepada server. Data dikirim oleh IoT tiap 1 detik. Subscriber menvalidasi data yang dikirim dengan mencek urutan dan nilai sensor. Nilai sensor disimulasikan dengan angka 0-1023 untuk sensor 1, 1023-2046 untuk sensor 2, dan 2047-3070 untuk sensor 3. Simulasi nilai digunakan untuk mencek urutan data.

Hasil yang diperoleh Tabel 1 membuktikan rata-rata jarak pengiriman adalah 1 detik dengan error margin 7%. Data yang terkirim sesuai dengan urutan. Dengan pembuktian ini maka ActiveMQ Artemis dapat digunakan sebagai IoT server.

5. Kesimpulan

Penggunaan ActiveMQ Artemis berhasil dilakukan dengan menggunakan berbagai macam IoT dan sensor yang dapat digunakan untuk mitigasi bencana. Pengembangan lebih lanjut dapat dilakukan dengan mengelola data sensor menjadi big data atau sistem cerdas atau membuat sensor yang mempunyai reliabilitas dan ketepatan data yang baik.

Tidak lupa kami ucapkan terima kasih kepada Kementerian Riset dan Teknologi Republik Indonesia (Ristekdikti) atas kesempatannya melaksanakan Penelitian Dosen Pemula.

Referensi

[1]. Presiden Republik Indonesia, "PP RI No 21 Tahun 2008," Peratur. Pemerintah, 2008, doi: 10.1017/CBO9781107415324.004.

[2]. S. L. Mufreni, "Rancang Bangun Sistem Antrean Multi Fungsi Yang Restful Dengan Menggunakan Credit Card-Sized Computer," J. Transm., vol. 20, no. April, pp. 79–84, 2018.

[3]. S. L. Mufreni, "RANCANG BANGUN BARCODE PRINTER SERVER DENGAN MENGGUNAKAN ACTIVEMQ CLASSIC DI RASPBERRY PI 3," J. Transm., vol. 22, no. April, pp. 62–66, 2020.

[4]. G. C. Hillar, MQTT Essentials - A Lightweight IoT Protocol, 1st ed. Packt Publishing, 2017.

[5]. N. Naik, "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP.," 2017.

[6]. P. Thota and Y. Kim, "Implementation and Comparison of M2M Protocols for Internet of Things," Proc. - 4th Int. Conf. Appl. Comput. Inf. Technol. 3rd Int. Conf. Comput. Sci. Appl. Informatics, 1st Int. Conf. Big Data, Cloud Comput. Data Sci. Eng. ACIT-CSII-BCD 2016, pp. 43–48, 2017, doi: 10.1109/ACIT-CSII-BCD.2016.021.

[7]. T. Yokotani and Y. Sasaki, "Comparison with HTTP and MQTT on required network resources for IoT," ICCEREC 2016 - Int. Conf. Control. Electron. Renew. Energy, Commun. 2016, Conf. Proc., pp. 1–6, 2017, doi: 10.1109/ICCEREC.2016.7814989.

[8]. Arduino-Based Embedded Systems : Interfacing, Simulation, and LabVIEW GUI. .

[9]. M. Schwartz, Internet of Things with ESP8266. Packt Publishing2017, 2016.

[10]. H. A. Rochman, R. Primananda, and H. Nurwasito, "Sistem Kendali Berbasis Mikrokontroler Menggunakan Protokol MQTT pada Smarthome," J. Pengemb. Teknol. Inf. dan Ilmu Komput., vol. 1, no. 6, pp. 445–455, 2017, [Online]. Available: http://j-ptiik.ub.ac.id.

[11]. A. P. Segara, R. Primananda, and S. R. Akbar, "Implementasi MQTT (Message Queuing Telemetry Transport) pada Sistem Monitoring Jaringan berbasis SNMP (Simple Network Management Protocol)," Pengemb. Teknol. Inf. dan Ilmu Komput., vol. 2, no. 2, pp. 695–702, 2018.

[12]. R. P. Pratama, P. K. Malang, R. P. Pratama, and S. Grid, "Aplikasi Wireless Sensor Esp8266 Untuk Smart Home," Semin. Nas. Teknol. dan Rekayasa, vol. IV, pp. 1–10, 2017.

[13]. I. B. P. Widja, "Sistem IoT Berbasis Protokol MQTT Dengan Mikrokontroler ESP8266 dan ESP32," in SNATIF Ke-5, 2018, pp. 329–336, doi: 10.2298/PAN0903301G.

[14]. C. Hasiholan, R. Primananda, and K. Amron, "Implementasi Konsep Internet of Things pada Sistem Monitoring Banjir menggunakan Protokol MQTT," vol. 2, no. 12, pp. 6128–6135, 2018.

[15]. J. Rumbaugh, I. Jacobson, and G. Booch, The Unified Modeling Language Reference Manual, 2nd ed. Pearson Education, Inc., 2005.