Purwarupa Kit *Remote Auto Engine Cutoff* Anti Kredit Macet untuk Kendaraan Bermotor Menggunakan NodeMCU ESP8266 dan Protokol MQTT Berbasis Internet Of Things

Satria Wijayandaru[1] dan Mohamad Yamin[2]

[1]Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Gunadarma
[2]Jurusan Teknik Mesin, Fakultas Teknologi Industri, Universitas Gunadarma
Jl. Margonda Raya No. 100, Pondok Cina, Beji, Depok 16424
E-mail : satriawijayandaru@gmail.com, mohay@staff.gunadarma.ac.id*)

Abstrak

Mudahnya pengajuan cicilan kendaraan bermotor membuat banyak orang melakukan kredit kendaraan bermotor yang tidak sesuai dengan kemapuan ekonominya. Akibatnya, tidak jarang orang mengalami kredit macet, atau terlambat membayar cicilan kendaraan bermotornya. Sehingga pada akhirnya dapat merugikan pihak kreditur. Saat ini, pihak kreditur mempekerjakan *debt collector* untuk melakukan penagihan secara paksa kepada debitur yang tidak jarang menggunakan cara yang kasar bahkan kekerasan yang menimbulkan keresahan. Penggunaan kit purwarupa *remote auto engine cutoff* dapat mengurangi pengihan kredit secara manual dengan cara mematikan kontak kendaraan apabila debitur belum melunasi kewajibannya membayar biaya kredit sesuai dengan tanggal yang sudah disepakati. Penelitian ini menggunakan protokol pengiriman pesan jarak jauh MQTT. MQTT dipilih karena *open source* dan tidak memerlukan *bandwidth* yang besar dalam proses pengiriman dan penerimaan datanya. Hal ini memungkinkan perangkat digunakan pada daerah dengan cakupan sinyal yang rendah. Dengan menggunakan protokol MQTT ini, pengiriman pesan dapat dilakukan dengan cepat. Hasil penelitian menunjukkan bahwa rata rata waktu yang diperlukan dalam percobaan pengiriman pesan menggunakan protokol MQTT sebanyak sepuluh kali percobaan dengan jarak antara perangkat dan akses poin satu meter tanpa penghalang adalah 0,956 milidetik pada topik "user1" dan 1,908 milidetik pada topik "report", kemudian saat perangkat dan akses poin berjarak sejauh lima meter tanpa penghalang memerlukan waktu rata – rata 1,206 milidetik pada topik "user1" dan 14,256 milidetik pada topik "report". Kemudian dilakukan percobaan dengan jarak antara perangkat dan akses poin sejauh lima meter dengan satu buah tembok penghalang, rata – rata waktu yang ditempuh dalam sepuluh percobaan adalah 1,319 milidetik pada topik "user1" dan 74,744 milidetik pada topik "report".

Kata Kunci: Internet of Things, Arduino, ESP8266, MQTT

Pendahuluan

Mudahnya pengajuan cicilan kendaraan bermotor membuat banyak orang melakukan kredit kendaraan bermotor yang tidak sesuai dengan kemampuan ekonominya. Akibatnya, tidak jarang orang mengalami kredit macet, atau terlambat membayar cicilan kendaraan bermotornya. Sehingga pada akhirnya dapat merugikan pihak kreditur. Saat ini, pihak kreditur mempekerjakan *debt collector* untuk melakukan penagihan secara paksa kepada debitur yang tidak jarang menggunakan cara kasar bahkan kekerasan yang menimbulkan keresahan. Untuk mengatasi masalah kredit macet kendaraan roda dua, perlu adanya pengendalian engine secara remot, tanpa sepengetahuan debitur. Sehingga pada saatnya pelunasan cicilan, *engine* akan otomatis berhenti dan kreditur dapat memberi pesan ke debitur. Hal ini dapat dilakukan dengan menggunakan perangkat NodeMCU v3 ESP8266 sebagai objek penelitian dan pengiriman data berbasis *Internet of Things* dengan protokol pengiriman data menggunakan MQTT (Mosquitto) [1].

Mosquitto atau MQTT merupakan protokol pengiriman data antar mesin menggunakan mode transportasi *publish/subscr*ibe. MQTT dapat digunakan pada alat dengan spesifikasi yang rendah, juga tidak memerlukan *bandwidth* yang besar untuk

DOI : http://dx.doi.org/10.32409/jikstik.20.1.402,
*)Penulis Korespondensi

keperluan pengiriman datanya, sehingga cocok digunakan di proyek yang tidak memiliki *bandwidth* besar. MQTT sendiri memerlukan sebuah broker yang terinstall di perangkat lain bertindak sebagai *server* MQTT agar MQTT *client* dapat berinteraksi satu sama lain.

Penggunaan protokol MQTT dan perangkat NodeMCU banyak digunakan terutama pada industri *Smart Ho*me dan teknologi perangkat *Internet of Things*. Salah satu implementasi protokol MQTT dan perangkat NodeMCU termuat dalam jurnal Implementasi Modul Wifi NodeMCU ESP8266 untuk Smart Home [2].

Pada penelitian ini perancangan purwarupa kit *remote auto engine cutoff* pada kendaraan bermotor roda dua anti kredit macet digunakan NodeMCU ESP8266 dan protokol MQTT berbasis *Internet Of Thing*s.

Metode Penelitian

Pada penelitian ini, metode yang digunakan untuk menyelesaikan masalah pembuatan purwarupa kit *remote auto engine cutoff* pada kendaraan roda dua adalah menggunakan protokol kendali jarak jauh Mosquitto (MQTT) yang dijalankan pada perangkat NodeMCU v3 ESP826 dengan MQTT Broker yang berjalan pada sistem operasi Linux Ubuntu Server 18.04. Adapun perangkat lunak yang digunakan dalam penelitian ini yaitu Oracle VM VirtualBox 6.0.8 r130520, Linux Ubuntu Server 18.04, Arduino IDE 1.8.12, *mosquitto* versi 1.4.15, dan *putty* 0.71.

Penelitian ini dilaksanakan dengan mengukur waktu yang dibutuhkan oleh protokol MQTT untuk mengirim dan menerima pesan dari *broker* ke *client*. Lamanya waktu yang dibutuhkan diukur menggunakan networking tools Wireshark.

Spesifikasi Perangkat Keras

Perangkat keras yang digunakan dalam pembuatan purwarupa kit *remote auto engine cutoff* kendaraan bermotor menggunakan NodeMCU ESP8266 dan protokol kendali jarak jauh MQTT adalah sebagai berikut:

Tabel 1: Tabel Perangkat Keras

Perangkat	Jumlah	Spesifikasi	Keterangan
Laptop	1	CPU Intel Core i7 7700HQ @ 3.8 GHz, RAM 8 GB, GPU Intel HD Graphics 630 dan Nvidia GTX 1050	Sebagai sarana penelitian
NodeMCU ESP8266	2	80 MHz CPU, 4 MB Flash Memory, 50 KB usable RAM, 802.11n Wifi	Sebagai sarana penelitian
Relay	1	5V DC input, 10A maximum current	Pemutus arus *ignition* kendaraan

Analisis Kebutuhan Perangkat Lunak

Perangkat lunak atau *software* yang digunakan untuk membantu proses pembuatan purwarupa kit *remote auto engine cutoff* kendaraan bermotor menggunakan NodeMCU ESP8266 diantaranya adalah sistem operasi Linux Ubuntu Server versi 18.04 LTS yang digunakan sebagai sistem operasi di sisi server yang dijalankan sebagai *virtual machine* di dalam *software* Oracle VM VirtualBox versi 6.0.8 r130520 [3]. Arduino IDE versi 1.8.12 [4] digunakan sebagai tempat di mana koding dilakukan. Daftar perangkat lunak dapat dilihat pada tabel 2.

Tabel 2: Daftar Perangkat Lunak

Software	Versi	Keterangan
Linux	Ubuntu Server 18.04 LTS 64bit	Sistem operasi pada *server* dan *client*
Oracle VM VirtualBox	Oracle VM VirtualBox 6.0.8 r130520	Sebagai media pembuatan purwarupa
Arduino IDE	Arduino 1.8.12	Sebagai IDE di mana koding dilakukan
Mosquitto	Mosquitto version 1.4.15	Sebagai pusat kendali protokol MQTT

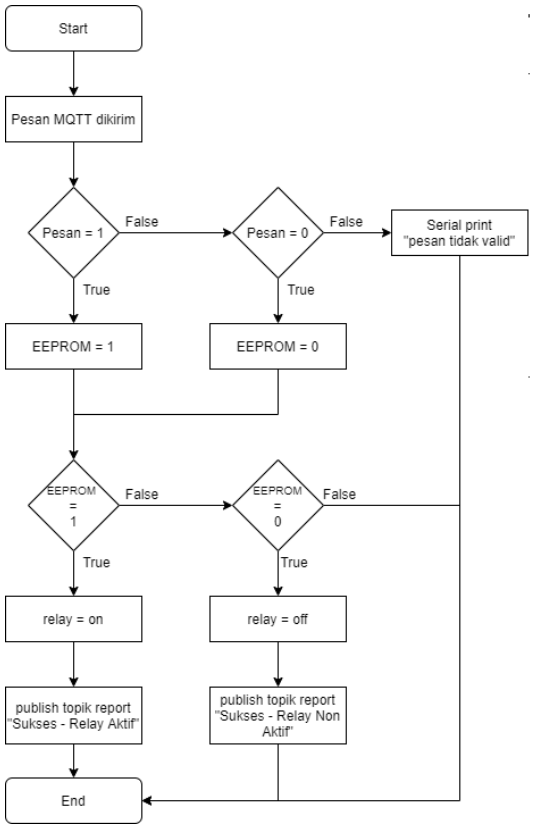
Implementasi

Dalam pembuatan purwarupa kit *remote auto engine cutoff* kendaraan bermotor menerapkan semua yang telah direncanakan pada tahap desain. Perangkat keras dan perangkat lunak yang digunakan pada tahap implementasi ini juga sesuai dengan analisis kebutuhan yang telah dijabarkan. Instalasi perangkat lunak yang perlu disiapkan antara lain adalah Oracle VM VirtualBox, Ubuntu Server 18.04 LTS, Arduino IDE, dan MQTT Dash Android [5]. Penambahan MQTT *broker* di MQTT Dash, dan pengkodean di sisi *client* dengan menggunakan Arduino IDE dilakukan agar koneksi antara *publisher*, *subscriber*, dan *broker* dapat terjalin dengan baik.

Tes dilakukan di sisi *client* menggunakan serial monitor untuk memastikan *client* terhubung ke *broker*. Dalam pembuatan script Arduino menggunakan Arduino IDE untuk diimplementasi ke NodeMCU ESP8266 [6], digunakan *library* tambahan yang disediakan oleh MQTT, yaitu *pubsubclient.h* dan *esp8266wifi.h* agar *client* dapat terkoneksi dengan jaringan wifi dan MQTT *broker*.

Selanjutnya, dilakukan uji coba koneksi kendali jarak jauh menggunakan aplikasi MQTT Dash di android dengan mengirimkan string “1” dan “0” ke topik “user1”. Apabila koneksi sudah terbangun dengan baik, akan muncul *string* yang diterima pada *serial monitor* yang terhubung ke perangkat NodeMCU v3 ESP8266. Saat pesan pada topik “user1” sudah diterima pada prangkat *kill switch*, perangkat tersebut akan melakukan *publish* ke topik

"report" yang berisi pesan "Sukses! - Relay Aktif" apabila pesan yang diterima pada topik "user1" adalah 1 atau "Sukses! - Relay Non Aktif" apabila pesan yang diterima apada topik "user1" adalah 0. *Flowchart* implementasi dan ujicoba serta alur *subscribe* dan *publish report* dapat dilihat pada Gambar 1 dan 2.

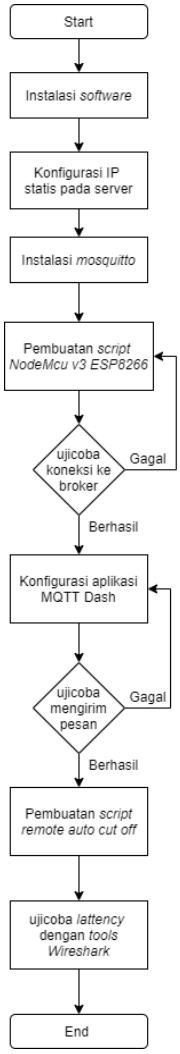


Gambar 1: Alur Subscribe dan Publish Report

Perangkat NodeMCU ESP8266 yang pertama akan digunakan sebagai *client* yang terpasang pada kendaraan bermotor. Perangkat ini akan menggunakan sebuah relay yang berfungsi memutus arus dari kunci kontak ke CDI (*Capacitor Discharge Ignition*). CDI merupakan sistem perapian dalam kendaraan bermotor. Apabila CDI ini tidak mendapat suplai daya, maka kendaraan bermotor tidak dapat dinyalakan. Diagram kelistrikan pada client dapat dilihat pada Gambar 3 berikut.

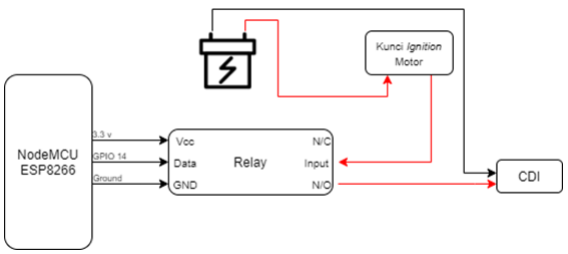
Perangkat *client remote auto engine cutoff* ini akan menerima pesan MQTT pada topik "user1" berupa string "1" dan "0". Apabila pesan MQTT yang diterima adalah "1", maka perangkat akan mengirimkan sinyal "HIGH" ke relay yang terhubung ke GPIO pin 14. Saat NodeMCU mendapat pesan 1 atau 0 dari topik "user1", NodeMCU akan mem-*publish* pesan melalui topik "report". Topik "report" ini akan diterima perangkat NodeMCU yang bertindak sebagai monitor. Perangkat *client remote auto engine cutoff* ini juga akan mengirimkan pesan melalui topik "reportonline" setiap 2000 milidetik untuk menandakan keaktifan perangkat client.

Pada sisi *relay*, terdapat tiga buah *pinout*, yaitu *input*, N/O (*normaly open*), dan N/C (*normaly closed*). Dalam penelitian ini, *pinout* yang digunakan adalah *input* dan *normaly open*.



Gambar 2: Flowchart Implementasi dan Uji Coba

Pinoutnormaly open dipilih karena sirkuit dari *ignition* ke CDI akan selalu terputus (*open circuit*) apabila *relay* tidak mendapatkan daya. CDI hanya akan mendapatkan daya apabila relay memiliki suplai daya dan mendapat sinyal dari NodeMCU.



Gambar 3: Diagram kelistrikan

Perangkat Monitor

Perangkat NodeMCU ini akan berfungsi sebagai monitor yang akan memantau aktifitas *client* melalui protokol komunikasi *mosquitto*. Perangkat ini akan men-*subscribe* dua buah topik, yaitu "*report*" dan "*reportonline*". Pada topik "*report*", perangkat monitor akan menerima data dari *client* berupa *string* sesuai dengan hal yang dilakukan oleh client.

Data yang dikirimkan oleh perangkat *client* pada topik "*report*" akan ditampilkan dalam serial monitor. Kemudian perangkat monitor juga menerima pesan dari topik "*reportonline*" setiap dua detik atau 2000 milidetik yang menandakan perangkat client masih dalam keadaan *online*. Pesan ini kemudian akan ditampilkan secara visual dengan menyalakan satu buah LED yang terintegrasi dalam modul NodeMCU ESP8266 dan terhubung secara langsung dengan GPIO pin 2.

Auto remote cutoff engine dengan integrasi database MySQL

Scriptremote auto cutoff engine dalam penelitian ini dibuat menggunakan bahasa pemrograman *python* yang akan berjalan pada sistem operasi Linux Ubuntu Server 18.04 LTS [7]. Program ini akan melakukan koneksi ke database "kredit" dan akan memanggil tabel "motor". *Field* yang dipanggil oleh program ini yaitu *field* ID, tanggaltempo, dan tanggalbayar. Untuk melakukan koneksi ke *database*, bahasa pemrograman *python* memerlukan module bernama *mysql.connector* [8].

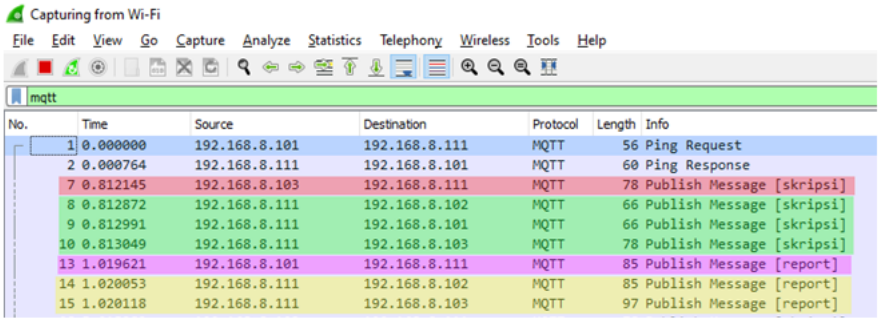
Setelah data berhasil dipanggil dari *database*, selanjutnya program akan memilah data berdasarkan *field* tanggaltempo dan tanggalbayar. Kedua *field* ini akan melakukan pencocokan dengan tanggal hari ini. Agar *python* dapat mengetahui tanggal hari ini, diperlukan module bernama datetime. Tanggal hari ini kemudian disimpan dalam variable "*today*" yang nantinya akan melalui fungsi percabangan.

Terdapat dua buah kondisi pada program ini, yang pertama adalah apabila tanggal hari ini sama dengan *field* tanggaltempo, maka program akan melakukan *publish* pada topik yang sesuai dengan *field* ID dan akan mengirimkan pesan "0" yang akan diinterpretasikan oleh MQTT *client* untuk mematikan *relay*. Kemudian kondisi kedua adalah apabila tanggal hari ini sama dengan *field* tanggaltempo, maka program akan melakukan *publish* pada topik yang sesuai dengan *field* ID dan akan mengirimkan pesan "1" yang akan diinterpretasikan oleh MQTT *client* untuk mematikan *relay*. Agar program ini dapat berjalan secara otomatis, diperlukan pengaturan cronjob pada sistem operasi Linux Ubuntu Server 18.04 LTS. Dengan menggunakan kode *cronjob* "0 4 * * *" berarti sistem akan menjalankan *script* secara otomatis setiap pukul 4 dini hari.

Hasil dan Pembahasan

Ujicoba dengan Aplikasi Wireshark

Ujicoba *latency* dilakukan dengan menggunakan *networking tools* yang bernama Wireshark [9]. Wireshark dapat melakukan analisa pada setiap paket data yang melewati *network interface card* yang dalam penelitian ini menggunakan wifi card Realtek 8821 AE yang terhubung ke akses poin. Filter yang tersedia dalam tools Wireshark dapat digunakan untuk memisahkan jenis paket data yang akan dianalisa. Hal ini bertujuan agar data yang akan dianalisa dapat dengan mudah terlihat dan tidak tercampur dengan data lain yang terdapat dalam jaringan.



Gambar 4: Tampilan analisa paket data Wireshark

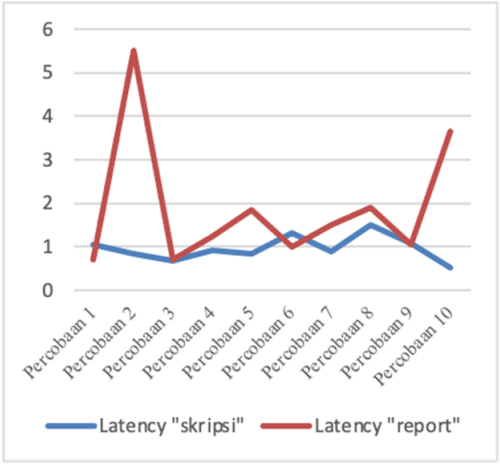
Gambar 4 adalah informasi dari paket data yang berhasil dianalisa oleh *tools* Wireshark, dapat dilihat pada waktu 0,812145 dengan tanda kotak berwarna merah, perangkat android dengan alamat IP 192.168.8.103 melakukan *publish* ke topik "user1" yang kemudian diterima oleh MQTT *broker* dengan alamat IP 192.168.8.111. Kemudian pada waktu 0,812872 sampai waktu 0,813049 yang diberi tanda dengan warna hijau, MQTT *broker* meneruskan pesan ke masing – masing *client* yang

men-*subscribe* topik yang sama, yaitu "user1". Kemudian MQTT *client* yang berlaku sebagai *kill switch* yang memilik IP 192.168.8.101 dan ditandai dengan warna ungu melakukan *publish* ke topik "*report*" pada waktu 1,019621 yang selanjutnya diterima oleh MQTT *broker* dan diteruskan kembali ke *client* yang men-*subscribe* topik "*report*". Dalam kasus ini perangkat monitor memiliki alamat IP 192.168.8.102 menerima pesan dalam topik "*report*" pada waktu 1,020053 yang ditandai dengan warna kuning.

Tabel 3: Rerata *latency* (dalam milidetik) 1 Meter *line of sight*

Software	Versi	Keterangan
Linux	Ubuntu Server 18.04 LTS 64bit	Sistem operasi pada *server* dan *client*
Oracle VM VirtualBox	Oracle VM VirtualBox 6.0.8 r130520	Sebagai media pembuatan purwarupa
Arduino IDE	Arduino 1.8.12	Sebagai IDE di mana koding dilakukan
Mosquitto	Mosquitto version 1.4.15	Sebagai pusat kendali protokol MQTT

Untuk mengetahui rerata *latency* pada saat pengiriman data melalui protokol MQTT, maka dilakukan sepuluh buah percobaan dengan hasil rata – rata latensi pengiriman data pada topik "user1" sebesar 0,956 milidetik dan pada topik "report" sebesar 1,908 milidetik. *Latency* pada topik "*report*" lebih besar dari pada *latency* pada topik "user1". Hal ini disebabkan pada topik "*report*" dikirim oleh perangkat yang sama dengan perangkat yang menerima topik "user1", sehingga menyebabkan penggunaan *resource* yang lebih besar saat melakukan pengiriman data. Hasil rerata data dapat dilihat pada Tabel 3 dan Gambar 5.



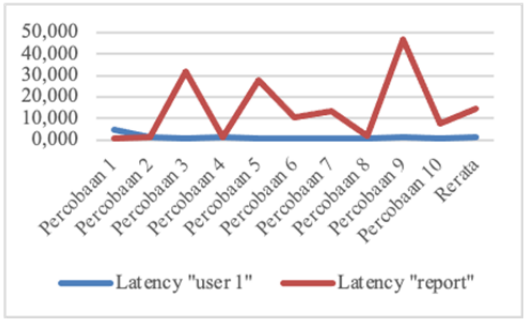
Gambar 5: Rerata Latency 1 Meter line of sight

Dalam menguji keandalan sistem, dilakukan kembali sebuah pengambilan data sebanyak sepuluh percobaan dengan jarak antara perangkat ESP8266 dan akses poin sejauh lima meter tanpa ada penghalang (*line of sight*). Dari sepuluh buah percobaan yang dilakukan, terlihat rerata *latency* mengalami peningkatan yang dikarenakan oleh jarak antara perangkat dan akses poin yang saling berjauhan. Hasil pengambilan data tersebut tertuang dalam Tabel 4 dan Gambar 6.

Tabel 4: Rerata *latency* (dalam milidetik) 5 Meter *line of sight*

Percobaan	Latency "user1"	Latency "report"
1	4.656	0.583
2	0.988	1.266
3	0.834	31.770
4	0.935	1.134
5	0.838	27.672
6	0.641	10.518
7	0.793	13.318
8	0.608	1.736
9	0.970	46.958
10	0.801	7.603
Rerata	1.206	14.256

Pengambilan data kembali dilakukan dengan jarak antar perangkat dan akses poin sebesar lima meter, namun dengan sebuah tembok sebagai penghalang antara perangkat. Dari sepuluh percobaan yang dilakukan, terlihat *lattency* sistem meningkat pada saat perangkat melakukan *publish* pada topik "report".

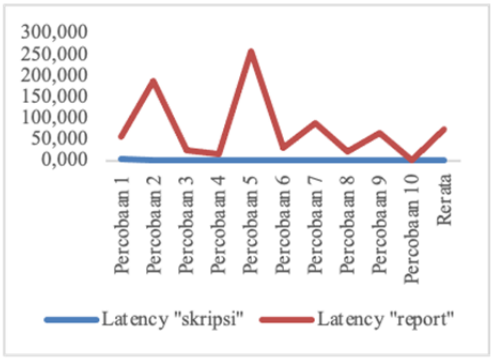


Gambar 6: Rerata *Latency* 5 Meter *line of sight*

Hal ini diakibatkan jauhnya jarak antara perangkat dan akses poin serta adanya tembok penghalang yang mengakibatkan *packet loss*. Detail rerata percobaan dapat dilihat pada Tabel 5. dan Gambar 7.

Tabel 5: Rerata *latency* (dalam milidetik) 5 Meter 1 tembok

Percobaan	Latency "user1"	Latency "report"
1	2.807	57.492
2	0.829	187.078
3	0.637	24.467
4	1.505	14.198
5	0.797	257.596
6	0.589	30.095
7	0.826	89.081
8	1.819	21.103
9	2.428	65.227
10	0.955	1.104
Rerata	1.319	74.744



Gambar 7: Rerata *Latency* 5 Meter 1 tembok

Dari sebanyak 30 data yang diambil, terdapat *latency spikes* di tiap percobaan yang diakibatkan interferensi sinyal nirkabel di sekitar wahana percobaan dan keterbatasan *processing power* dari perangkat ESP8266 yang digunakan. *Latency spikes* dapat terlihat dari grafik masing – masing percobaan yang menunjukkan perbedaan hasil perhitungan *latency*.

Penutup

Setelah melakukan serangkaian perancangan dan uji coba, purwarupa kit *remote auto engine cutoff* anti kredit macet untuk kendaraan bermotor menggunakan NodeMCU ESP8266 dan protokol MQTT berbasis *Internet of Things* dinyatakan dapat berfungsi dengan baik. Perangkat baik perangkat *client*, *monitor* dan *server* dapat terkoneksi dengan baik dengan koneksi TCP/IP dan protokol MQTT.

Rerata waktu minimum yang dibutuhkan oleh masing – masing perangkat dalam proses pengiriman dan penerimaan data adalah 0,956 milidetik dengan jarak antara perangkat dan akses poin sejauh satu meter tanpa adanya hambatan atau *line of sight*. Sementara untuk waktu maksimum dalam proses pengiriman data pada topik MQTT "user1" adalah 1,319 dengan jarak antara perangkat dan akses poin sejauh 5 meter dan dengan penghalang satu buah tembok.

Antenna pemancar dan penerima sinyal wifi dengan *gain* yang lebih besar dapat digunakan untuk menanggulangi masalah *data loss* apabila diperlukan. Namun dalam praktik penggunaan kit *remote auto engine cutoff* anti kredit macet untuk kendaraan bermotor ini tidak diperlukan *latency* yang minim untuk dapat bekerja dengan baik.

Daftar Pustaka

[1] Bryan Boyd, Joel Gauci, Michael P Robertson, Nguyen Van Duy, Rahul Gupta, Vasfi Gucer, V. K. , "Building Real-time Mobile Solutions with MQTT and IBM MessageSight". IBM Redbooks, 2014.

[2] M. Fajar Wicaksono, "Implementasi Modul Wifi NodeMCU Esp8266 Untuk Smart Home", J. Tek. Komput. Unikom-Komputika 6, pp: 9-14, 2017.

[3] Anonym, "Oracle VMVirtual Box-User Manual Version 6.1.16 ", Oracle Corporation, diakses daring pada https://www. virtualbox.org/, 2020.

[4] Feri Djuandi, "Pengenalan Arduino", www.tokobuku.com, 2011.

[5] Anonym, "Software, R. MQTT Dash (IoT, Smart Home) User Manual", diakses daring pada https://play.google.com/store/apps/details?id=net.routix.mqttdash&hl=in&gl=US, 2020.

[6] Anonym, "ESP8266", diakses daring pada https://www.espressif.com/en/ products/socs/esp8266ex/overview, 2020.

[7] R. A Raharja, A. Yunianto & W. Widyantoro,"Pengenalan Linux", Open Source Campus Agreement Modul Pelatihan, 2011

[8] Lynn Beighley and Michael Morrison, "Head First PHP & MySQL: A Brain-Friendly Guide 1st Edition", O'Reilly Media, ISBN-13 : 978-0596006303, 2009.

[9] Angela Orebaugh, J. B. Gilbert Ramirez, "Wireshark & Ethereal Network Protocol Analyzer Toolkit", Publisher: Elsevier, 2006.