REPOSITOR, Vol. 3, No. 1, Januari 2021, Pp. 87-96
ISSN : 2714-7975
E-ISSN : 2716-1382

Rancang Bangun Over the Air Update Firmware Pada Perangkat Iot Dengan Protokol MQTT

Supriyanto*[1], Mahar Faiqurahman[2], Wahyu Andhyka Kusuma[3]
[1,2,3]Universitas Muhammadiyah Malang
supriyanto4id@gmail.com*[1], maharf@gmail.com[2], kusuma.wahyu.a@gmail.com[3]

Abstrak

Perangkat IoT yang diimlementasi pada banyak tempat dapat mengalami perubahan berupa update firmware. Update firmware pada perangkat IoT biasanya dilakukan dengan mengambil perangkat IoT, lalu menghubungkan ke komputer mengunakan komunikasi serial melalui kabel usb to micro usb, selanjutnya melakukan update firmware pada perangkat IoT dan mengembalikan perangkat IoT ke tempat. Jika sistem pada perangkat IoT sudah dapat berkomunikasi melalui antar muka jaringan, tidak perlu lagi mengunakan kabel usb to micro usb, karena bisa dimanfaatkan over the air update firmware mengunaka antar muka jaringan pada perangkat IoT. Over the air update firmware adalah memuat firmware hasil build dari arduino ide pada perangkat IoT mengunakan antar muka jaringan Wi-Fi, pada penelitian ini perangkat IoT mengunakan mikrokontroler esp8266 12E. Untuk melakukan update firmware perangkat IoT digunakan protokol MQTT untuk menjembatani antara aplikasi berbasis website sebagai interface penguna untuk PUBLISH file firmware ke perangkat IoT. Hasil dari implementasi Aplikasi berbasis website untuk over the air update firmware pada perangkat IoT dengan protokol MQTT, dalam 10 kali pengujian pengiriman file firmware perangkat IoT mengunakan masing-masing QoS 0, QoS 1, dan QoS 2, didapatkan hasil QoS 2 lebih direkomendasikan untuk digunakan mengirim file firmware dengan keberhasilan update firmware QoS 0 = 50 %, QoS 1 = 70% dan QoS 2 = 80% dari 10 kali percobaan pengirman file firmware pada perangkat IoT.

Kata Kunci: *OTA, Firmware, MQTT, Perangkat IOT*

Abstract

IoT devices that are implemented in many places can experience changes in the form of firmware updates. Firmware update on an IoT device is usually done by taking an IoT device, then connecting to the computer using serial communication via a usb to micro usb cable, then updating the firmware on the IoT device and returning the IoT device to its place. If the system on the IoT device is able to communicate through a network interface, there is no need to use a USB to Micro USB cable, because it can be utilized over the air firmware update using the network interface on the IoT device. Over the air firmware update is to load the firmware build from Arduino Idea on an IoT device using the Wi-Fi network interface, in this study the IoT device uses the ES8266 12E microcontroller. To update the firmware of the IoT device the MQTT protocol is used to bridge the website-based application as a user interface for PUBLISH firmware files to the IoT device. The results of the implementation of a website-based application for over the air firmware update on IoT devices with the MQTT protocol, in 10 times testing the sending of IoT device firmware files using each of QoS 0, QoS 1, and QoS 2, the results obtained QoS 2 are more recommended for use sending firmware files with successful firmware update QoS 0 = 50%, QoS 1 = 70% and QoS 2 = 80% of 10 attempts to test the firmware file on an IoT device.

Keywords: *OTA, Update Firmware, MQTT, IoT Device*

1. Pendahuluan

Pada akhir 2013, ada 9,1 miliar unit perangkat *Internet of Things* (IoT) dengan konektifitas *Internet Protocol* dan berkomunikasi tanpa interaksi dengan manusia, *Internasional Data Corporation (IDC*) memperkirakan pertumbuhan IoT yang diterapkann tiap tahun mencapai 17.5% diperkirakan menjadi 28,1 miliar di tahun 2020 [1]. Bahkan *Cisco* mempunyai predeksi dua kali lipat lebih besar yaitu 50 miliar pada tahun 2020 [2]. Perangkat IoT yang digunakan diberbagai tempat sering dianggap sebagai sistem yang tidak perlu mengubah *requirements* dan fungsinya, namun, pada kenyataanya dimana perangkat IoT ini berjalan pasti akan berubah(3).

Perubahan ini meliputi perubahan *behavior*, parameter yang terkait komunikasi dengan sistem lain atau penguna, memperbaiki kesalahan, bisa masalah keamanan, yang dilaporkan penguna setelah perangkat IoT digunakan [3].

Berbagai perubahan perangkat IoT dapat dilakukan dengan menganti *firmware*, untuk menganti *firmware* pada perangkat IoT harus keluar mengambil perangkat IoT, menghubungkan ke komputer, melakukan *update* dan mengembalikan perangkat IoT ke tempat. Namun, hal ini tidak dapat terus dilakukan bagi perusahaan yang memiliki perangkat IoT di berbagai tempat, seperti yang dilakukan *Chrysler* "merek mobil" pada tahun 2015 mereka dikritik karena mengirim perangkat *flashdisk* kepelangan untuk melakukan *update firmware* karena sangat rentan, *flashdisk* dapat diambil, di modifikasi dan dikirim kembali [4].

Jika sistem pada perangkat IoT sudah dapat berkomunikasi melalui antarmuka jaringan, hal ini bisa dimanfaatkan untuk menerapkan pembaruan firmware pada sistem IoT yang disebut dengan *Over The Air* (OTA)(3). OTA di lakukan oleh *Tesla* pada tahun tahun 2016 mengirimkan pembaruan *firmware* pada mobil mereka dan konsumen dapat mengatur akan melakukan pembaruan pada saat mobil di parkir [4].

Over The Air update adalah proses memuat *firmware* pada modul ESP "perangkat *IoT*" mengunakan koneksi jaringan Wi-Fi dari pada mengunakan kabel *port serial* [5]. Secara umum isitilah OTA adalah mekanisme pengunaan *wireless* untuk mengirim data, memperbarui paket untuk pembaruan *firmware* atau perangkat lunak ke perangkat *mobile*, sehingga penguna tidak perlu pergi mengakses fisik perangkat untuk mengubah aplikasi, parameter*, firmware*, atau memperbarui *software* [6].

Over The Air Update pada perangkat IoT sudah ada di pasaran dalam produk merek *Libelium*, namun *OTA* hanya bisa dilakukan pada perangkat IoT *Libelium* melalui *File Tranfer Protocol* (FTP). Selain itu ada *particle.io* sama seperti *Libelium* hanya bisa digunakan untuk perangkat IoT yang mereka sediakan tidak bisa untuk perangkat lain. Selain dua perangkat berbayar *libelium* dan *particle.io*, OTA di sediakan oleh *espresif* melalui produk bernama *esp8266 12-E*, dapat melakukan update *firmware* melalui *Arduino ide,* web browser , dan *HTTP Server*(5*).*

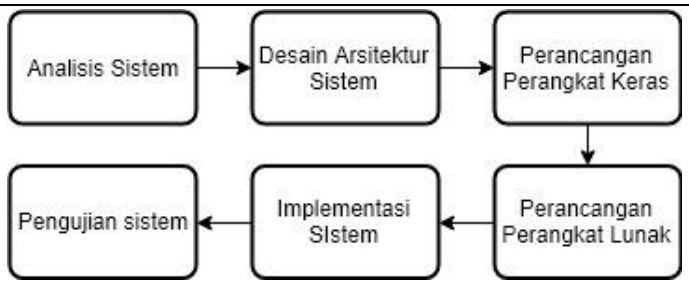
Over The Air Update dengan *esp8266* 12-E yang telah tersedia melalui *arduino ide* dan web browser memilik keterbatasan hanya bisa dilakukan dalam satu jaringan yang sama, dan HTTP Server bisa dilakukan dengan jaringan berbeda [5]. Namun pada pada penelitian pengujian sensing data suhu dan kelembapan pada penelitian sebelumnya menunjukan protokol HTTP 6 kali lebih lambat melakukan transfer data dari pada *protokol MQTT* dalam 60 detik dalam 5 kali percobaan di dapatkan rata-rata *HTTP* 934.4 data terkirim dan *MQTT* 6520.2 data terkirim [7).

Protokol MQTT adalah protokol pesan *publish/subcribe,* sangat sederhana, dan ringan, dirancang untuk perangkat yang terbatas oleh jaringan dengan *bandwidth* rendah, latensi tinggi atau tidak dapat diandalkan. Prinsip desain *protokol MQTT* adalah untuk meminimalkan *bandwidth* jaringan dan kebutuhan sumber daya perangkat, sambil berusaha memastikan kehandalan dan beberapa tingkat kepastian pengiriman data benar-benar terkirim *Quality of Service (QoS)* [8]. Contoh pengunaan protokol MQTT adalah Facebook Messenger [9] pada awal peluncuranya tahun 2011.

Berdasarkan latar belakang tersebut akan diimplementasikan *protokol MQTT* yang dapat melakukan *Over The Air update* pada perangkat *IoT. Over The Air update* dengan *protokol MQTT* akan digunakan untuk melakukan *update firmware*, *software* pada perangkat IoT, dari jaringan lokal atau jaringan internet, melalui media aplikasi berbasis website. Aplikasi berbasis website sebagai media *interface* penguna untuk melakukan *update file firmware* dan melakukan monitoring hasil *upload* apakah berhasil atau tidak. *Protocol* MQTT digunakan sebagai media pengiriman *file firmware* hasil *build* dari *Arduino IDE* berupa *file tipe bin*, yang di *publish* ke perangkat IoT yang telah melakukan s*ubscribe* pada suatu *topic*.

2. Metode Penelitian

Penerapan over the air update firmware dengan protocol MQTT dengan aplikasi berbasis website sebagai interface penguna, dilakukan dengan beberapa tahapan, yaitu analisis Sistem, desain arsitekture sistem, perancangan perangkat keras, perancangan perangkat lunak, implementasi sistem dan pengujian sistem, seperti pada Gambar 1 berikut.



Gambar 1. Alur penelitian

2.1 Analisis Sistem

Dari analisis masalah, maka di buat Server yang telah di konfigurasi, untuk menjadi web server, menyimpan aplikasi berbasis web sebagai interface penguna dan protocol MQTT berfungsi untuk jembatan komunikasi antara node perangkat IoT. Protocol MQTT adalah protocol komunikasi SUBSCRIBE dan PUBLISH, di mana sebelum mengirim dan menerima pesan client harus terlebih dahulu terkoneksi dengan koneksi TCP ke server broker MQTT yang telah di konfigurasi, selanjutnya publisher dan subscriber harus memiliki topik yang sama untuk saling komunikasi.

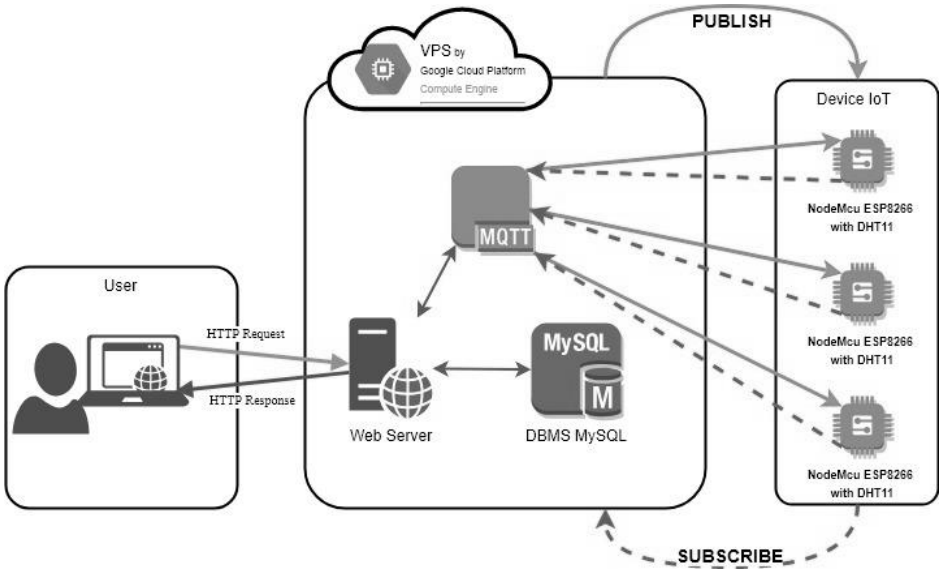
Publisher akan mengirim data ke server broker MQTT lalu broker akan mengirim ke subscriber, untuk identifikasi antara publisher dan subscriber harus memiliki topik yang sama agar pesan tersampaikan contoh topik “rumah/dapur/suhu” jadi setiap publisher dan subscriber akan memiliki topik yang sama.

Melakukan Over The Air update firmware kepada perangkat IoT di lakukan dengan cara mengambil file firmware hasil compile dari Arduino ide dengan ekstensi .bin, selanjutnya melakukan upload ke aplikasi berbasis web dan PUBLISH ke broker MQTT, lalu perangkat IoT melakukan SUBSCRIBE dan menerima file ekstensi .bin selajuntnya melakukan update.

Aplikasi berbasis web dibuat dengan PHP Framework Codeigniter, penguna harus login terlebih dulu, setelah login akan tampil pilihan untuk upload file firmware ekstensi .bin, form publish topik tujuan, dan button publish.

2.2 Desain Arsitekture Sistem

Arsitektur sistem penerapan protokol MQTT untuk *over the air update firmware*, mengunakan aplikasi berbasis website sebagai *interface* penguna dengan rancangan topologi sebagai berikut.



Gambar 2. Arsitektur system protokol MQTT untuk OTA update Firmware perangkat IoT

Dari rancangan topologi Gambar 2, terdiri dari Vitual Private Server (VPS) yang dikonfigurasi dengan protokol MQTT, Web Server Apache2, dan DBMS MySQL. Dalam penelitian

ini perangkat IoT yang di gunakan adalah modul NodeMCU ESP8266-12E sebagai sensor node untuk sensing data suhu dan kelembapan dengan DHT 11, sensor node tersebut akan menerima pembaruan firmware, dan penguna yang akan mengakses aplikasi berbasis web melalui browser.

Agar perangkat IoT dapat terhubung ke VPS maka harus di hubungkan dengan Wi-Fi yang sudah terhubung ke internet, seperti itu juga dengan aplikasi berbasis website yang akan di akses oleh penguna melalui browser internet, harus di pastikan memiliki nama domain atau ip publik yang bisa diakses melalui browser internet.

Agar rancangan arsitektur sistem gambar 2 dapat laksanakan sesuai dengan rancangan yang telah dibuat maka dibutuhkan hardware dan software sebagai berikut;

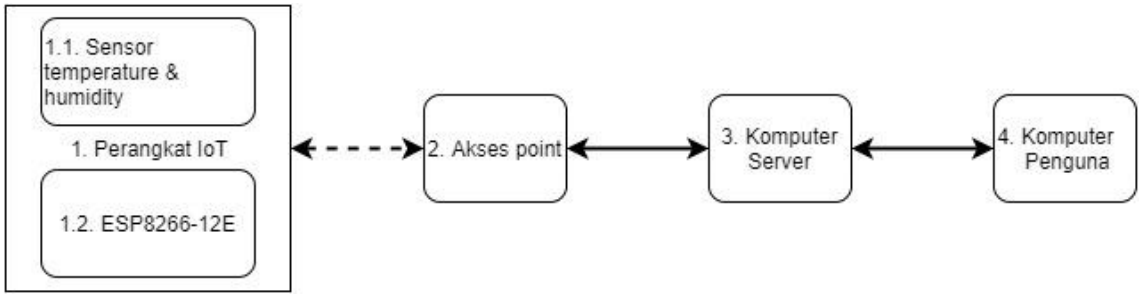
Tabel 1. Software dan Hardware sistem OTA dengan MQTT

Software	**Hardware**
Penguna	Laptop
Windows 10	Kabel USB 2.0 to Micro USB
Browser	
PuTTy	
Atom IDE	
Server VPS	
LAMP (Linux, Apache, MySQL, PHP)	CPU 1 Ghz
Linux Ubuntu Server 16.04	Ram 1 GB
Broker EMQTT 2.0	Hardisk 10 GB
Perangkat IoT	
Arduino IDE	Nodemcu ESP8266 12E
	Sensor DHT 1

Dari Tabel 1 hardware dan software terbagi menjadi 3 jenis yaitu untuk Penguna, Virtual Private Server, dan perangkat IoT.

2.3 Rancangan Perangkat Keras

Dari rancangan arsitektur *sistem over the air update firmware* dengan protokol MQTT mengunakan aplikasi berbasis website sebagai interface penguna. Rancangan hardware terdiri dari perangkat IoT, akses point, komputer server, dan komputer penguna seperti yang terlihat pada Gambar 3 dibawah ini.



Gambar 3. Rancangan hardware protocol MQTT untuk OTA update firmware perangkat IoT

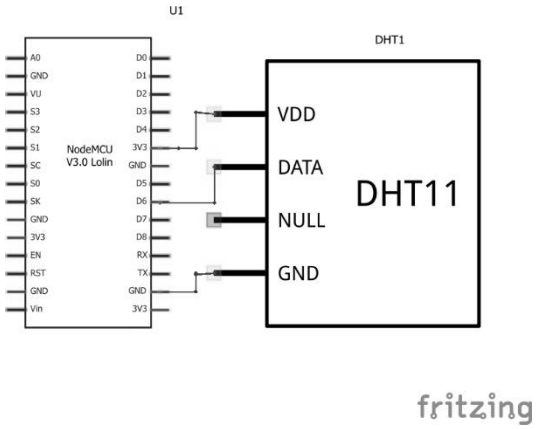
Keterangan:

◄----► : Terhubung dengan jaringan *Wi-Fi*

◄——► : Terhubung ke jaringan internet

1. Perangkat IoT terdariri dari mikrokontroler dan sensor. Sensor temperature dan humadity mengunakan DHT 11 dan Mikrokontroler mengunakan NodeMcu ESP8266-12E.
2. Akses Point yang sudah terhubung ke internet.
3. Komputer Server sebagai tempat menginstall web server Linux, Apache, MYSQL, dan PHP, dan menginstall EMQTT Broker.
4. Komputer Penguna untuk penguna melakukan akses ke aplikasi berbasis website melalui browser untuk melakukan OTA dengan MQTT.

Perangkat IoT terdiri dari NodeMcu ESP8266-12E dan sensor DHT 11. Nodemcu ESP8266-12E digunakan untuk menerima *update firmware* dari aplikasi berbasis website yang kirim oleh penguna melalui protokol MQTT dan Sensor DHT 11 digunakan sebagai monitoring apakah program pada *firmware* yang di kirim berfungsi atau tidak, selain itu juga sensor di gunakan untuk mengirim data temperatur dan humadity di sekitar perangkat IoT dikirim ke aplikasi berbasis website. Jadi akan dibuat dua firmware yang telah terisi program PUBLISH data sensor, dan SUBSCRIBE topik *firmware* yang akan digunakan untuk menerima *update* dari aplikasi berbasis website, dengan rancangan skematik sebagai berikut.

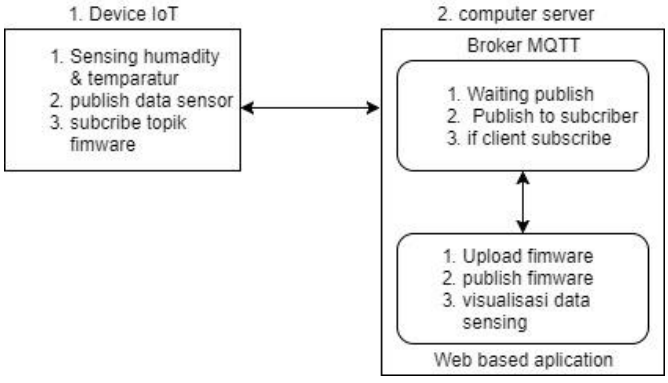


Gambar 4. Rangkaian skematik perangkat IoT

Dari rangkaian skematik Gambar 4 DHT 11 memiliki 4 pin yaitu *VCC*, data, dan *GND*, dihubungkan ke Nodemcu ESP8266 12-E VCC dihubungkan ke VCC dan GND dihubungkan ke GND, data di hubungkan ke D6 pada NodeMcu ESP8266 12-E.

2.4 Rancangan Perangkat Lunak

Perancangan perangkat lunak untuk setiap komponen pada sistem Over The Air update firmware dengan protokol MQTT mengunakan aplikasi berbasis website, yang akan diaplikasakan pada perangkat IoT, dan server. Berikut adalah rancangan proses perangkat lunak pada Gambar 5.



Gambar 5. Perancangan Proses Perangkat Lunak

Keterangan:
1. Desain proses perangkat lunak pada perangkat IoT
2. Desain proses perangkat lunak pada Komputer Server

◄——► Jaringan Internet

Dari desian perancangan proses perangkat lunak pada Gambar 5 perangkat lunak di bagi menjadi 2 bagian perangkat lunak pada perangat IoT dan pada server. Perangkat lunak pada perangkat IoT melakukan s*ensing* data sensor suhu dan kelembapan lalu melakukan PUBLISH data sensor dan melakukan SUBSCRIBE topik *firmware* untuk menerima data *firmware* dari

PUBLISH yang di lakukan aplikasi berbasis website. Pada *server broker* MQTT menunggu PUBLISH dari penguna jika ada yang melakukan PUBLISH dari penguna maka *broker* akan melakukan PUBLISH kepada penguna yang melakukan SUBSCRIBE dan memiliki topik yang sama, pada aplikasi berbasis website melakukan SUBSCRIBE data sensor dari perangkat IoT dan melakukan *upload firmware* lalu melakukan PUBLISH file bin ke broker MQTT, lalu meneruskan ke perangkat IoT yang memiliki topik sama.

2.5 Implementasi Sistem

Impelementasi sistem dilakukan berdasarkan desain sistem yang telah dirancang. Implementasi sistem terdiri dari implementasi perangkat keras dan implementasi perangkat lunak.

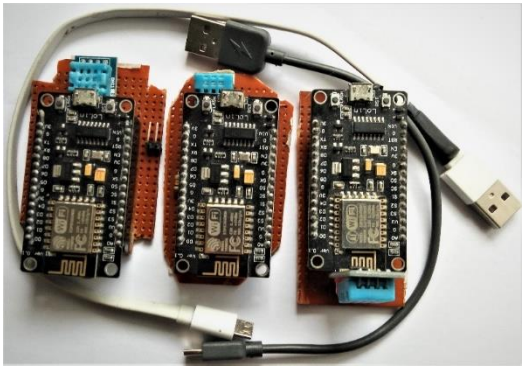
2.5.1 Implementasi perangkat keras IoT

Untuk dapat melakukan implementasi perangkat IoT dibutuhkan alat dan bahan seperti Tabel 2 berikut.

Tabel 2. Alat dan Bahan Perangkat IoT

Nama alat dan bahan	Jumlah
Mikrokontroler Nodemcu ESP8266-12E	3
Sensor DHT 11	3
Kabel usb micro	3

Dari alat dan bahan pada Tabel 2, dibuat hasil perancangan perangkat IoT sebagai berikut.



Gambar 6 rangkaian perangkat keras IoT dan usb micro

Dari hasil implementasi perangkat IoT pada Gambar 6 terdapat 3 buah Nodemcu ESP8266-12E dengan masing-masing Nodemcu terhubung dengan sensor DHT 11. Setiap perangkat IoT yang telah dilakukan implementasi selanjutnya di program melalui kabel usb micro.

2.5.2 Implementasi perangkat Lunak

Implementasi perangkat lunak dibagi menjadi dua bagian implementasi perangkat lunak pada perangkat IoT dan implementasi perangkat lunak pada server.

Perangkat lunak pada perangkat IoT adalah proses melakukan pemprogaman dengan bahasa C pada text editor arduino ide setelah di lakukan pemprogaman selanjutnya di lakukan *compile*, hasil dari *compile* berupa *file.bin* yang akan di *upload* pada aplikasi berbsis website.

Implementasi perangkat lunak pada *server* terbagi menjadi tiga, pertama implementasi aplikasi berbasis website melakuka pemprogaman dengan bahasa program PHP fremwork codeigniter, kedua instalasi, konfigurasi dan setting Linux Aapache Mysql DAN PHP pada server, dan ketiga instalasi, konfigirasi dan *setting* EMQX broker protokol MQTT.

2.6 Pengujian Sistem

Setelah rancangan *hadware* dan *software,* selanjutnya adalah dilakukan pengujian untuk melihat kemampuan broker MQTT untuk mengirim *file firmware* melalui aplikasi berbasis website ke perangkat IoT dengan tahapan pengujian sebagai berikut.

1. Uji coba pengiriman file firmware konstan 276 KB untuk mendapatkan tingkat kegagalan dan keberhasilan update firmware pada perangkat IoT
2. Uji coba apakah kegagalan dan keberhasilan update firmware pada perangkat IoT berpengaruh terhadap delay

3. Hasil Penelitian dan Pembahasan

Dari hasil pengujian pengiriman *file firmware* konstan 276 KB untuk melakuan *Over The Air Update Firmware* pada perangkat IoT mengunakan aplikasi berbasis website, dibuat analisis untuk digunakan sebagai rekomendasi QoS yang digunakan saat mengirim *file firmware* dengan dua sub bab analisis antara lain. Pengaruh pengunaan QoS terhadap keberhasilan dan kegagalan update firmware pada perangkat IoT dan pengaruh keberhasialn dan kegagalan update *firmwar*e pada perangkat IoT, terhadap *delay*.

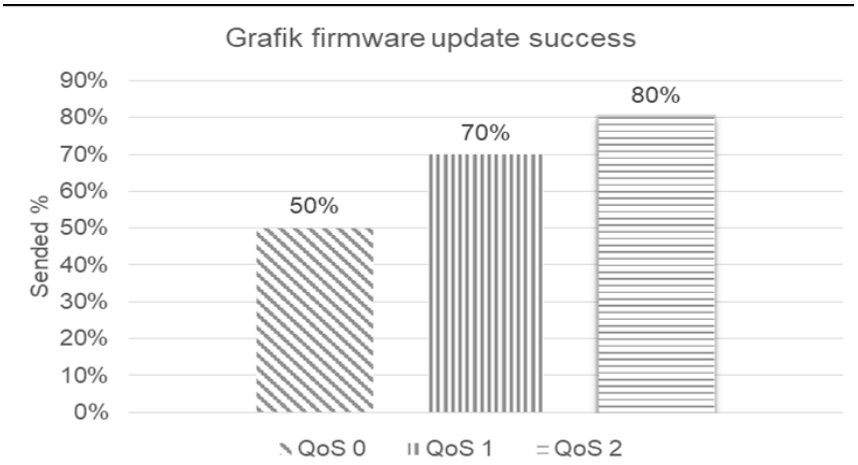
3.1 Pengaruh pengunaan QoS terhadap keberhasilan dan kegagalan update firmware para perangkat IoT.

Dari 10 kali pengiriman *file firmware* ke perangkat IoT dengan mengunakan setiap QoS 0, QoS 1, dan QoS 2, didapatkan hasil perangkat IoT berhasil di *update* dan *restart* sukses dan gagal sebaga berikut.

Tabel 3. Perangkat IoT sukses update firmware dan restart

QoS	Updated successfully
QoS 0	50%
QoS 1	70%
QoS 2	80%

Dari Tabel 3 dibuat grafik pada setiap QoS agar mudah dianalisis sebagai berikut.



Gambar 7. update firmware sukses pada setiap QoS 0, QoS 1 dan QoS 2

Dari Gambar 7 *firmware* yang berhasil di update ke perangkat IoT, pada saat pengirman dengan 10 kali percobaan pada QoS 0 = 50%, QOS 1 = 70% dan QoS 2 = 80%. Hasil *update firmware* ke perangkat IoT tersebut terjadi karena *Quality of Service* setiap QoS 0, QoS 1 dan QoS 2, memiliki paket control MQTT yang berbeda-beda, sehingga jika mengunakan QoS 0 tingkat keberhasilan *update firmware* 50 % hanya melakukan PUBLISH *firmware* tanpa menerima kembali *acknowledge* PUBACK, jika mengunakan QoS 1 dengan tingkat keberhasilan *update firmware* 70 %, PUBLISH *firmware* akan mendapatkan balasan kembali berupa PUBACK, dan jika mengunakan QoS 2 dengan tingkat keberhasilan *update firmare* 80 % paket kontrol MQTT yang digunakan pada saat peroses PUBLISH *firmware* memiliki *acknowledge* sebanyak tiga paket kontrol MQTT yaitu PUBREC, PUBREL, dan PUBCOM, yang memastikan *firmware* benar-benar terkirim.

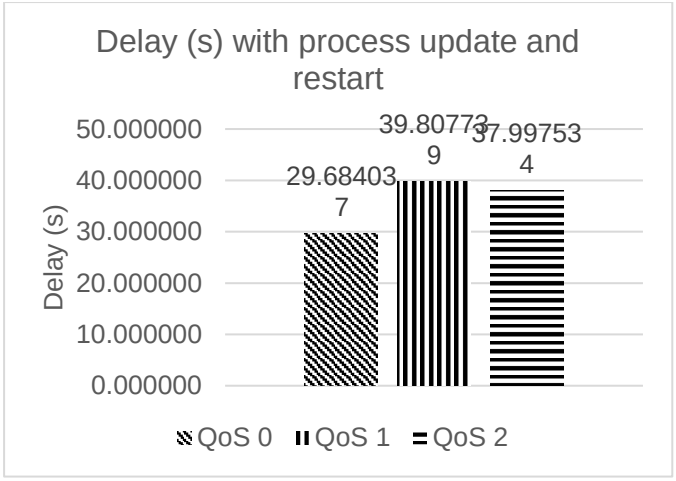
3.2 Pengaruh keberhasialn dan kegagalan *update firmware* pada perangkat IoT terhadap *delay*

Dari pembahasan mengenai *delay* pada sub bab 4.2.3.1.1 *delay* pengujian performa mengirim *file firmware* konstan, hanya menghitung *delay* yang terjadi dalam sistem ketika mengirim *file firmware* antara aplikasi berbasis website saat PUBLISH dan saat perangkat IoT menerima kontrol paket MQTT berisi *file firmware*, namun tidak menghitung *delay* yang terjadi pada saat perangkat IoT menerima *file firmware* lalu melakukan *update* kemudian melakukan *restart* perangkat IoT. Dalam peroses perangkat IoT melakukan *update* lalu *restart,* ini lah terjadi perbedaan waktu *delay*. karena ketika peroses *update firmware* ini terjadi *update firmware* gagal dan *update firmware* berhasil. Jika *update firmware* berhasil peroses akan lebih cepat dari pada ketika peroses *update firmware* gagal, dengan hasil pengujian *delay* sebagai berikut.

Tabel 4 Delay pengiriman firmware, update firmware dan restart perangkat IoT

QoS	Delay(s) sending firmware with process update & restart
QoS 0	29.684037
QoS 1	39.807739
QoS 2	37.997534

Tabel 4 adalah *delay* pada saat PUBLISH yang dilakukan apliasi berbasis website dan diterima oleh perangkat IoT pada Tabel 4, ditambah dengan proses *update firmware* dan *restart* yang dilakukan perangkat IoT.



Gambar 8. Delay pengiriman firmware, update firmware dan restart perangkat IoT

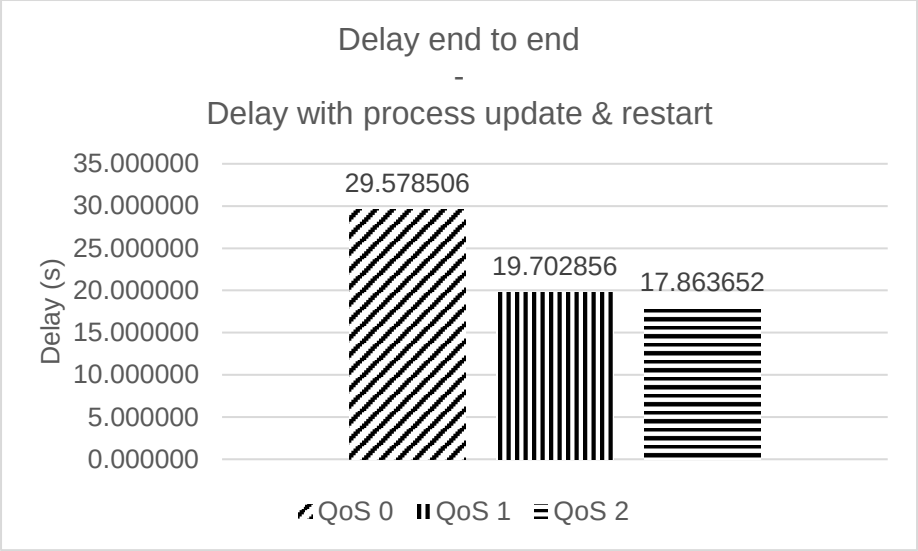
Dari Gambar 8 *delay* ketika mengunakan QoS 0 dengan proses *update firmware* dan *restart* didapatkan *delay* 29.684037/s, jauh berbeda dengan nilai *delay* QoS 0 pada Gambar 8 delay pengiriman firmware, dengan nilai *delay* 0.105521/s, perbedaan ini terjadi karena nilai *delay* QoS 0 pada Gambar 8 ditambah dengan proses *update firmware* dan *restart* perangkat IoT lalu mengirimkan paket PUBLISH berisi *version update firmware*.

Pada *delay* QoS 1 dan QoS 2 memiliki nilai *delay* yang meningkat dari pada *delay* Gambar 8 *delay* pengiriman firmware. hal ini terjadi karena ada peroses *update firmware* dan *restart* perangkat IoT pada saat menerima *file firmware*, peroses *update firmware* dan *restart* perangkat IoT dapat terjadi berhasil dan gagal *update firmware* ke perangkat IoT, jika peroses *update firmware* berhasil maka *delay* akan menjadi lebih cepat, dari pada jika gagal akan menjadi lebih lama. Dari *delay* proses pengiriman *file firmware* dikurang dengan *delay* proses *update firmware* dan restart perangkat IoT didapatkan hasil *delay* tertinggi pada masing-masing QoS 0, QoS 1, dan QoS 2 sebagai berikut.

Tabel 5. Hasil delay pengirman firmware dikurang delay peroses update dan restart

QoS	Delay sending firmware - Delay with process update and restart
QoS 0	29.578506
QoS 1	19.702856
QoS 2	17.863652

Tabel 5 adalah hasil pengurangan *delay* pada Tabel 4 dan *delay* pada Tabel 3, untuk melihat *delay* tertinggi pada saat aplikasi berbasis website PUBLISH *firmware* lalu diterima oleh perangkat IoT, dan melakukan *update firmware* kemudian melakukan *restart* perangkat IoT.



Gambar 9. Delay peroses pengiriman firmware, update firmware dan restart perangkat IoT

Dari Gambar 9 pada QoS 0 memiliki nilai *delay* tertinggi 29.578506/s karena mengikuti tingkat kegagalan pada saat proses *update firmware* yang hanya terkirim 50 % seperti terlihat pada Gambar 9, berikutnya QoS 1 *delay* tertinggi ke dua 19.702856/s dengan tingkat kegagalan proses *update* 30 % dan pada QoS 2 memiliki tingkat *delay* 17.663625/s dengan tingkat kegagalan proses u*pdate* 20 %.

Dari proses *update firmware* yang berhasil dan gagal pada saat proses *update firmware* dilanjutkan *restart* perangkat IoT dapat di simpulan bahwa pada saat *update firmware* perangkat IoT, pengunaan QoS 2 lebih disarankan dari pada QoS 1 dan QoS 0, karena tingkat kegagalan pengiriman *update firmware* mengunakan QoS 2 = 20 % atau 2 kali kegagalan dari 10 kali percobaan pengiriman, namun pengunaan QoS 2, juga berpengaruh ke pada pengunaan sumberdaya CPU dan Memory paling tinggi dari pada QoS 0, dan QoS 1, seperti terlihat pada Gambar 8 dan Gambar 9.

4. Kesimpulan

Setelah melakukan implementasi dan pengujian pada bab IV dari tugas akhir dengan judul Rancang Bangun Layanan Over the Air Update Firmware dengan Protokol Message Queue Telemetry Transport (MQTT) pada IoT maka didapatkan hasil dan kesimpulan sebagai berikut.

1. Impelementasi protokol MQTT untuk melakukan over the air update firmware perangkat IoT berhasil dilakukan.
2. Dari hasil pengujian pengiriman file firmware untuk over the air update firmware perangkat IoT didapatkan hasil direkomendasi mengunakan QoS 2 karena tingkat keberhasilan pengiriman file firmware dalam 10 kali percobaan adalah 80 % dari pada QoS 1 = 70 % dan QoS 0 = 50 %. Pengunaan QoS 2 pada saat mengirim file firmware mengakibatkan pengunaan sumber daya CPU dan Memory juga tinggi dari pada mengunakan QoS 1, dan QoS 0.

3. Tingkat kegagalan QoS 0 = 50%, QoS 1 = 70 % dan QoS 2 = 80% berpengaruh terhadap delay pada saat mengirim file firmware. Dengan delay tertinggi QoS 0 = 29.578506/s, diikuti QoS 1 = 19.702856/s dan delay terendah QoS 2 = 17.663625/s

Referensi

[1] Lund D, Morales M. *Worldwide and Regional Internet of Things* (IoT) 2014 – 2020 Forecast : A Virtuous Circle of Proven Value and Demand. IDC Anal Futur. 2014;(May):29.

[2] Evans D. *The Internet of Things How the Next Evolution of the Internet Is Changing Everything* [Internet]. 2011. Available from: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[3] Reißmann S, Pape C. *An Over the Air Update Mechanism for ESP8266 Microcontrollers.* ICSNC 2017 Twelfth Int Conf Syst Networks Commun An [Internet]. 2017;(October):11–7. Available from: https://www.researchgate.net/publication/320335879_An_Over_the_Air_Update_Mechanism_for_ESP8266_Microcontrollers

[4] Lee Jeffrey. *Over-The-Air Firmware: The Critical Driver of IoT* Success - DZone IoT [Internet]. https://dzone.com. 2017 [cited 2018 Mar 30]. Available from: https://dzone.com/articles/over-the-air-firmware-the-critical-driver-of-iot-s

[5] *ESP8266. OTA Update · ESP8266 Arduino Core* [Internet]. [cited 2018 Mar 29]. Available from: http://esp8266.github.io/Arduino/versions/2.0.0/doc/ota_updates/ota_updates.html

[6] Quadri ASA, Sidek B. O. *An Introduction to Over-the-Air Programming in Wireless Sensor Networks*. Int J Comput Sci Netw Solut [Internet]. 2014;2:33–49. Available from: https://www.researchgate.net/publication/262181994%0AAn

[7] R A Atmoko*, R Riantini MKH. *IoT real time data acquisition using MQTT protocol*. Int Conf Phys Instrum Adv Mater. 2016;012003.

[8] Stanford Clark Andy NA. MQTT [Internet]. IBM. 1999 [cited 2017 Oct 9]. Available from: http://mqtt.org/

[9] Zhang Lucy. *Building Facebook Messenger* [Internet]. 2011 [cited 2017 Oct 10]. Available from: https://www.facebook.com/notes/facebook-engineering/building-facebook-messenger/10150259350998920

[10] Lee S, Kim H, Hong DK, Ju H. Correlation analysis of MQTT *loss and delay according to QoS level.* Int Conf Inf Netw. 2013;714–7.