PROPOSAL PENELITIAN

PENGEMBANGAN ARSITEKTUR PENGIRIMAN DATA IOT BERBASIS MQTT DAN CoAP

KURNIAWAN GIGIH LUTFI UMAM

SEKOLAH PASCASARJANA
INSTITUT PERTANIAN BOGOR
BOGOR
2019

RINGKASAN

KURNIAWAN GIGIH LUTFI UMAM. Pengembangan Arsitektur Pengiriman Data IoT Berbasis MQTT dan CoAP. Dibimbing oleh SRI WAHJUNI dan HENDRA RAHMAWAN.

Industry 4.0 merupakan *trend* integrasi, pertukaran data serta peningkatan efisiensi dalam proses industri. Dalam rangka menghadapi era *industry* 4.0 salah satu pendukung untuk mancapai *industry* 4.0 adalah dengan *internet of things* (IoT). Penerapan *internet of things* sudah banyak dilakukan dalam berbagai bidang yaitu *smart city, smart home, smart agricultural, smart vehicle, smart health, smart campus, smart security* dan bidang perikanan. Permasalahan utama dalam IoT adalah harus memperhatikan protokol yang baik untuk komunikasi agar *latency* komunikasi dapat diminimalisir.

Dalam penelitian ini akan dibangun sebuah arsitektur IoT yang menggabungkan antara protokol *message queue telemetry transport* (MQTT) dan *constrained application protocol* (CoAP). MQTT adalah protokol dengan konsumsi daya yang rendah serta mempunyai nilai *delay, bandwidth* dan *packet loss* data yang rendah serta akurasi tinggi. CoAP adalah protokol ringan berbasis *web transfer* yang digunakan sebagai protokol pada *layer* aplikasi.

Dalam arsitektur yang dibangun MQTT digunakan dalam komunikasi data antara *IoT board* dengan IoT *Gateway*, sementara protokol CoAP digunakan dalam komunikasi data antara IoT *Gateway* dengan Server aplikasi. Protokol REST digunakan untuk menangani pengiriman data dari server aplikasi dengan server database dan ke *end device*. Tujuan yang ingin dicapai dalam penelitan ini adalah membandingkan pengiriman data pada protokol MQTT dan REST, CoAP dan REST serta meningkatkan kestabilan pengiriman data dengan menerapkan redis.

Pengujian yang akan dilakukan mempunyai empat skema yaitu REST dan REST, MQTT dan REST, MQTT dan CoAP, MQTT-Redis dan CoAP. Nilai yang diukur pada pengujian adalah protokol *overhead*, *packet loss* dan jumlah *byte* per paket. Pengujian tersebut dilakukan dengan menggunakan data nyata dan data sintetik. Data nyata digunakan untuk mempresentasikan keadaan normal, Sementara data sintetik digunakan untuk mempresentasikan keadaan tidak normal *(stress test).* Penelitian ini diharapkan dapat menghasilkan arsitektur IoT yang efisien dan handal dalam pengiriman data.

Kata kunci: *industry* 4.0, *internet of things*, protokol, *smart*, arsitektur, MQTT, CoAP.

PENGEMBANGAN ARSITEKTUR PENGIRIMAN DATA IOT BERBASIS MQTT DAN CoAP

KURNIAWAN GIGIH LUTFI UMAM

Usulan Penelitian
sebagai salah satu syarat untuk melakukan penelitian dan
penulisan tesis
pada
Program Studi Ilmu Komputer

SEKOLAH PASCASARJANA
INSTITUT PERTANIAN BOGOR
BOGOR
2019

Judul Tesis : Pengembangan Arsitektur Pengiriman Data IoT Berbasis MQTT dan CoAP
Nama : Kurniawan Gigih Lutfi Umam
NIM : G651180011

Disetujui oleh

Komisi Pembimbing

Dr Ir Sri Wahjuni, MT	Dr Hendra Rahmawan, S.Kom, MT
Ketua	Anggota

Diketahui oleh

Ketua Program Studi Ilmu Komputer	a.n Dekan Sekolah Pascasarjana Sekretaris Program Magister
Dr Ir Sri Wahjuni, MT	Prof Dr Ir Nahrowi, M.Sc

Tanggal Kolokium : 09 Mei 2019

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

1 PENDAHULUAN

Latar Belakang

Industry 4.0 merupakan *trend* integrasi, pertukaran data serta peningkatan efisiensi dalam proses *industry* yang mencakup *internet of things* (IoT), *internet industry*, pabrik cerdas dan pabrik berbasis *cloud* (Vaidya *et al.* 2018). Dalam rangka menghadapi era *industry* 4.0 salah satu pendukung untuk mencapai *industry* 4.0 adalah dengan *internet of things*, *Internet of things* adalah *object* fisik yang terhubung melalui internet atau sistem tertanam yang terdiri dari perangkat lunak elektronik, sensor dan modul konektivitas seperti modem atau *Wi-Fi*. Hal ini memungkinkan perangkat keras dapat bertukar data dan dapat mengontrol atau memonitoring perangkat keras lainnya (Wukkadada *et al.* 2018). Untuk penerapan *internet of things* sudah banyak dilakukan dalam berbagai bidang yaitu *smart city* (Zabasta *et al.* 2018), *smart home* (*Coelho et al.* 2015), *smart agricultural* (Takelar *et al.* 2017; Wahjuni *et al.* 2017), *smart vehicle* (Wang *et al.* 2016), *smart health* (Kaur *et al.* 2017), *smart campus* (Zhamanov *et al.* 2017), *security* (Saifuzzaman *et al.* 2017) dan bidang perikanan (Kim *et al*. 2018; Wahjuni *et al.* 2016).

Implementasi IoT untuk *smart farming* telah dilakukan oleh Departemen Ilmu Komputer FMIPA IPB dengan menggunakan protokol *Representasional state transfer* (REST) yang diberi nama KOMIoT. Dimana REST adalah gaya arsitektur *software* untuk merancang sistem yang terdistribusi dan digunakan untuk *Word Wide Web* (Livari 2016). Pada implementasi KOMIoT menghasilkan nilai *overhead* protokol yang besar pada pengiriman data yang kecil dengan nilai rata-rata 80%. *Overhead* adalah persentase *non–data* atau *header* dari total data *respon* sebuah protokol (Falconer 2013). Penelitian ini menyarankan menggunakan protokol yang rendah *overhead* yaitu protokol MQTT (Wahjuni *et al.* 2017).

Dalam penerapan IoT hal yang sangat penting adalah memperhatikan protokol yang baik untuk komunikasi agar *latency* komunikasi dapat diminimalisir (Wukkadada *et al.* 2018). Saat ini beberapa protokol yang ada untuk *internet of things* diantaranya MQTT (*message queue telemetry transport*), CoAP (*constrained application protocol*), HTTP (*hypertext transfer protocol*), AMQP (*advanced message queuing protocol*) dan XMPP (*extensible messaging and presence protocol*). Salah satu protokol yang sering digunakan adalah MQTT. MQTT merupakan protokol dengan komunikasi ringan yang diciptakan oleh IBM yang dikhususkan untuk komunikasi M2M (*machine to machine*) yang dirancang dengan meminimalkan bandwidth jaringan dan sumber daya serta *lacenty* yang rendah (Arron. 2016).

Penelitian Wukkadada *et al*. (2018) membandingkan antara dua protokol yang sering digunakan yaitu HTTP (*hypertext transfer protocol*) dan MQTT, dengan parameter pengujian adalah *packet loss* dan konsumsi listrik. Penelitian ini menyatakan protokol MQTT menghasilkan *throughput* 90 kali lebih cepat dan konsumsi daya lebih rendah dibandingkan protokol HTTP.

Luzuriaga *et al*. (2018) melakukan penelitian tentang toleransi gangguan pada MQTT yang mempunyai kelemahan dalam ketahanan sehingga koneksi dapat mengalami gangguan. Dalam hal ini pendekatan yang digunakan adalah *disrupt tolerant network* (DTN) dengan melakukan pengujian kestabilan jaringan. Parameter yang diuji adalah *round trip time* (RTT) dan *packet loss*, dengan lingkungan pengujian WSN (*wireless sensor network*) dan *backbone*. Hasil pengujian pada WSN memberikan nilai

RTT di bawah 75 ms dan *packet loss* mencapai 5% untuk *publish* dan 3% untuk *subscribe* dengan masing-masing 6 kali percobaan. Dalam jaringan *backbone* waktu yang diperlukan untuk menangani sebuah *bundle* adalah antara 40 dan 65 ms, sedangkan untuk penundaan pesan antar-pengiriman dengan waktu beberapa puluh milidetik, waktu perlahan-lahan akan meningkat ketika degradasi saluran komunikasi bertambah. Dalam hal ini Luzuriaga *et al.* menyatakan bahwa dengan nilai yang dihasilkan protokol MQTT dapat diterima untuk aplikasi IoT secara umum.

Beberapa penerapan protokol MQTT diantaranya adalah penelitian Takelar *et al*. (2017) yang melakukan penelitian tentang protokol MQTT untuk *weather monitoring* dan *precision farming* secara *real-time* dengan tujuan mengurangi biaya keseluruhan, meningkatkan kualitas dan jumlah panen dengan parameter suhu tanaman, kelembapan tanah dan intensitas cahaya. Penelitian ini menghasilkan visualisasi grafis dari hasil pemantauan menggunakan protokol MQTT.

Naik (2017) melakukan penelitian tentang evaluasi perbandingan relatif dari empat protokol pengiriman pesan dalam IoT yaitu MQTT, CoAP, AMQP (*Advanced Message Queuing Protocol*) dan HTTP. Parameter perbandingan yang dilakukan adalah *Message Size* vs *Message Overhead*, *Power Consumption* vs *Resource Requirement*, *Bandwidth* vs *Latency*, *QoS* vs *Interoperability*, Security vs *Provisioning*, *M2M Usage* vs *Standardisation*. Penelitian ini menyatakan pengujian yang dilakukan berdasarkan karakteristik dari setiap protokol secara komparatif dan dari berbagai literatur. Hasil yang didapatkan dari pengujian dengan parameter tersebut adalah protokol MQTT dan CoAP dapat dikatakan baik untuk pengiriman pesan dalam IoT tetapi tetap harus memperhatikan kebutuhan dan kesesuaian dengan perangkat yang akan dibuat.

Penelitian Westhuizen *et al.* (2018) melakukan penelitian tentang perbandingan antara CoAP dan MQTT. Penelitian ini menyatakan bahwa tidak ada arsitektur standar dan sempurna untuk IoT, masing-masing mempunyai kelebihan dan kekurangan. Hasil dari penelitian ini menyarankan menggabungkan kedua protokol atau menjalankannya secara paralel. Broker MQTT akan bertanggung jawab atas tindakan sensor dan protokol CoAP akan bertanggung jawab untuk pengiriman data dari broker ke server dan pengelolaan sistem IoT.

Dalam upaya meningkatkan kestabilan pengiriman data IoT dapat menerapkan *cache memory* dengan redis, selain digunakan untuk *cache memory*, redis dapat digunakan sebagai broker IoT tetapi dalam penerapannya terdapat masalah *syntactical interoperability* jika redis digunakan untuk broker. *Syntactical interoperability* adalah kemampuan dua atau lebih komponen untuk bertukar informasi dan menggunakan informasi yang tersedia. Wulandari *et al.* (2018) melakukan penelitian tentang implementasi *Cluster Message Broker* untuk skalabilitas pada *minddleware* dengan tujuan untuk mengatasi masalah pada *syntactical interoperability* dikarenakan terbatasnya jumlah data yang mampu ditampung oleh *memory* RAM. Penelitian ini membangun *cluster message broker* dengan 6 redis yang terdapat di 3 Raspberry pi dengan bantuan *Ioredis* sebagai pengintegrasi *middleware* dan *cluster*. Hasil dari penelitian ini menyatakan penggunaan *cluster message broker* dapat diimplemtasikan pada *middleware* serta dapat berkomunikasi dengan protokol MQTT dan CoAP.

Untuk memperbaiki kelemahan penelitian sebelumnya (KOMIoT) dalam hal menyediakan layanan pengiriman data pada IoT yang lebih efisien, penelitian ini akan dikembangkan sebuah arsitektur IoT yang menggabungkan protokol MQTT dan CoAP. Sedangkan untuk menangani masalah kestabilan untuk pengiriman data dalam penelitian ini akan menerapakan *cache memory* redis dengan skema penugasan pada

saat tidak tersedianya koneksi internet pada broker maka data yang didapatkan dari input sensor akan disimpan sementara kedalam redis, setelah mendapatkan koneksi internet maka data akan dikirimkan ke server kemudian data yang berada diredis akan di hapus. Jika tidak ada permasalahan dalam koneksi internet maka data sensor akan dikirimkan langsung ke server tanpa melewati *cache memory* redis. Arsitektur IoT yang akan dibangun juga dilengkapi dengan *middleware* yang sesuai untuk mengelola pertukaran data dan interkonektifitas antar komponen IoT.

Perumusan Masalah

Perumusan masalah pada penelitian ini adalah bagaimana menggabungkan dua protokol CoAP dan MQTT pada sebuah arsitektur yang mempunyai perbedaan dalam segi karakteristik diantaranya *header size*, *message size*, metode pengiriman data, abstraksi, *transfer* protokol (Naik. 2017) serta menangani masalah kestabilan pengiriman data.

Tujuan Penelitian

Berdasarkan permasalahan yang ada, penelitian ini bertujuan untuk:

1. Membandingkan pengiriman data IoT di internet antara protokol MQTT+REST dengan REST+REST.
2. Membandingkan pengiriman data IoT di internet antara protokol MQTT+CoAP dengan MQTT+REST.
3. Meningkatkan kestabilan pengiriman data.

Manfaat Penelitian

Manfaat penelitian ini diharapkan dapat menghasilkan arsitektur IoT baru yang dapat meningkatkan kinerja IoT dalam hal efisiensi protokol dan kehandalan pengiriman data.

Ruang Lingkup Penelitian

Untuk menerapkan arsitektur yang diusulkan dalam penelitian ini akan dibangun sebuah *prototype* arsitektur IoT dengan kasus *monitoring* dibidang pertanian hidroponik.

2 TINJAUAN PUSTAKA

Arsitektur Internet of Things (IoT)

Arsitektur IoT terdiri dari berbagai lapisan teknologi yang mendukung IoT yaitu *smart device/sensor layer, gateway and network, management service layer, application layer*. Berfungsi untuk menggambarkan bagaimana berbagai teknologi saling berhubungan satu sama lain dan untuk mengkomunikasikan skalabilitas, modularitas,

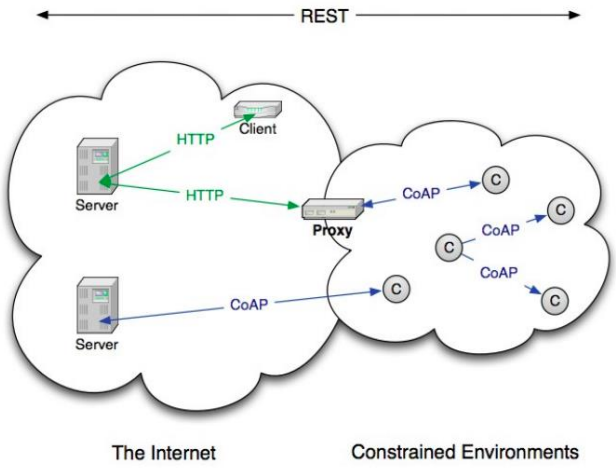
dan konfigurasi penyebaran IoT dalam skenario yang berbeda (Patel *et al.* 2016). Arsitektur sistem IoT didasarkan pada konteks dan operasi sesuai aplikasi yang ingin dibuat, sehingga arsitektur IoT akan bervariasi tergantung konteks aplikasi (Suresh *et al.* 2014).

Constrained Application Protocol (CoAP)

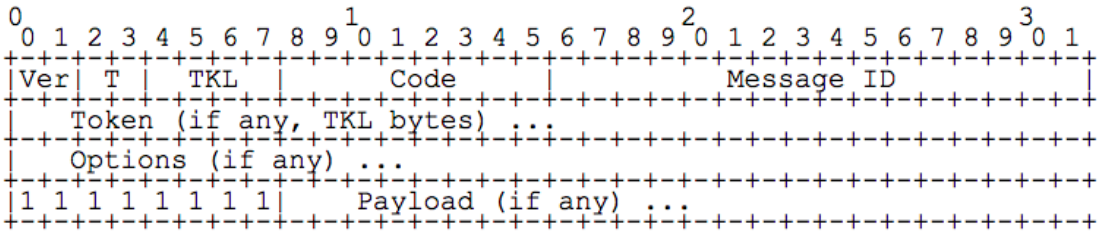
Constrained Application Protocol adalah suatu *web transfer* protokol khusus untuk pengunaan dengan *node* terbatas dan jaringan yang dibatasi (Shelby *et al.* 2014). CoAP dikembangkan oleh *international engineering task force* (IETF) merupakan suatu protokol layer aplikasi dan termasuk ke dalam standar RFC 7252. CoAP yang juga merupakan suatu *lightweight* protokol dimaksudkan untuk digunakan sebagai pengganti HTTP untuk menjadi protokol pada layer aplikasi di dalam IoT (*internet of things*). CoAP menyediakan model interaksi *request/response* antara aplikasi dan *endpoint*, dikarenakan CoAP memiliki beberapa fitur yang menyerupai HTTP. Tidak seperti HTTP yang beroperasi pada TCP, CoAP beroperasi pada UDP untuk menghindari *congestion control* yang kompleks. CoAP menerapkan arsitektur *Representational State Transfer* (REST), sehingga menyediakan URI dan metode seperti GET, POST, PUT, dan DELETE. Untuk menutupi kelemahan UDP, CoAP memiliki sebuah mekanisme retransmisi. Untuk mengatasi kelemahan dalam keterbatasan sumber daya, CoAP perlu mengoptimalkan panjang datagram untuk menyediakan komunikasi yang handal. Terdapat empat tipe pesan yang digunakan pada CoAP untuk melakukan pertukaran data antara *client* dan *server*, yaitu :

1. *Confirmable* (CON), merupakan pesan yang berisi request dan memerlukan *Acknowledgment*.
2. *Non-Confirmable* (NON), merupakan pesan yang digunakan berulang secara teratur tanpa memerlukan *Acknowledgment*.
3. *Acknowledgment* (ACK), merupakan pesan yang berisi *response*.
4. *Reset* (RST), merupakan pesan yang diguanakan ketika pesan CON tidak diterima dengan benar atau terdapan konteks yang hilang.

Untuk arsitektur dari CoAP dan format pengiriman pesan, seperti Gambar 1 dan Gambar 2.



Gambar 1 Arsitektur CoAP (Sherby et al. 2014)



Gambar 2 Format Pesan CoAP (Sherby et al. 2014)

Dapat dilihat pada Gambar 2 terdapat lima bagian yaitu :

1. Ver : *version* dari protokol CoAP.
2. T : tipe pesan yang dikirimkan seperti (*Confirmable*, *Non-Confirmable, Acknowledgement, Reset*)
3. TKL : *Token length*, jika dibutuhkan *token* untuk pesan.
4. Code : *Request method* (1-10) atau *response code* (40-255)
5. Message ID : untuk mengidentifikasi pesan yang terdiri dari 16-bit.

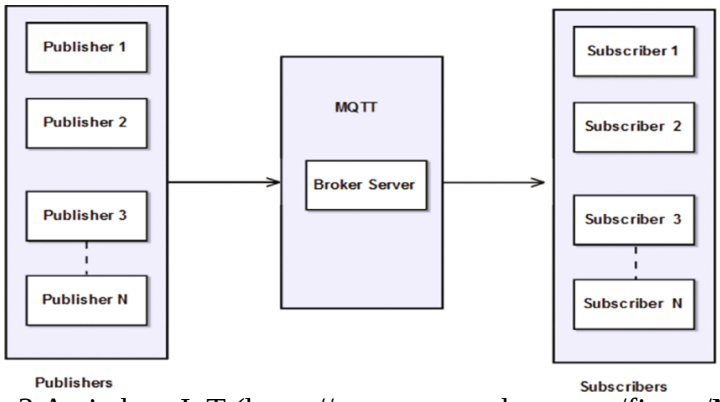
Message Queue Telemetry Transport (MQTT)

Message Queue Telemetry Transport (MQTT) adalah sebuah protokol komunikasi data *machine to machine* (M2M) yang berada pada layer aplikasi, MQTT bersifat *lightweight* message artinya MQTT berkomunikasi dengan mengirimkan data pesan yang memiliki header berukuran kecil yaitu hanya sebesar 2 *bytes* untuk setiap jenis data, sehingga dapat bekerja di dalam lingkungan yang terbatas sumber dayanya seperti kecilnya *bandwidth* dan terbatasnya sumber daya listrik, selain itu protokol ini juga menjamin terkiriminya semua pesan walaupun koneksi terputus sementara, protokol MQTT menggunakan metode *publish/subscribe* untuk metode komunikasinya.

Publish/subscribe sendiri adalah sebuah pola pertukaran pesan di dalam komuunikasi jaringan dimana pengirim data disebut *publisher* dan penerima data disebut dengan *subscriber*, metode *publish/subscribe* memiliki beberapa kelebihan salah satunya yaitu *loose coupling* atau *decouple* dimana berarti antara *publisher* dan *subscriber* tidak saling mengetahui keberadaannya, terdapat tiga buah *decoupling* yaitu *time decoupling*, *space decoupling* dan *synchronization decoupling*, *time decoupling* adalah sebuah kondisi dimana *publisher* dan *subscriber* tidak harus saling aktif pada waktu yang sama, *space decoupling* adalah dimana *publisher* dan *subscriber* aktif di waktu yang sama akan tetepi antara *publisher* dan *subscriber* tidak saling mengetahui keberadaan dan identitas satu sama lain, dan yang terakhir adalah *synchronization decoupling* kondisi dimana pengaturan *event* baik itu penerimaan atau pengiriman pesan di sebuah *node* hingga tidak saling mengganggu satu sama lain (Adi *et al*. 2016).

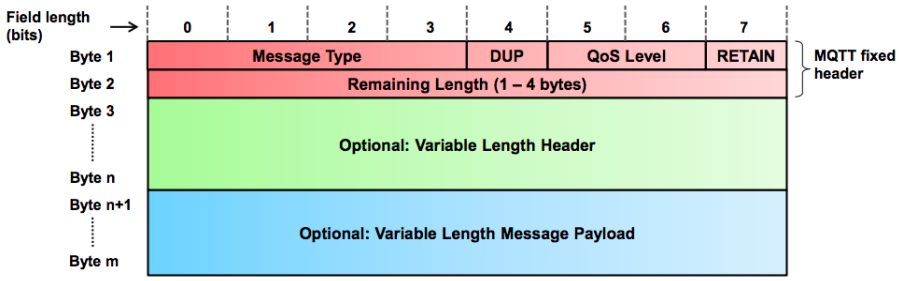
Pengiriman data pada MQTT didasari oleh topik, topik ini nantinya yang akan menentukan pesan dari *publisher* harus dikirim pada *subscirber* yang mana, topik ini dapat bersifat hirarki, MQTT *topic* memiliki tipe data string dan untuk perbedaan hirarki atau level dari topik digunakan tanda baca “/” (Solace. 2019.). MQTT memiliki 3 *level quality of service* (QOS) dalam pengiriman pesannya yaitu 0,1,2 (Lampkin *et al.* 2012). Untuk mendukung penerapan MQTT pada IoT dibutuhkan sebuah broker. Beberapa contoh broker yaitu Mosquitto, EMQ, Mosca, VerneMQ, Hive MQ. Dalam penelitian ini menggunakan mosquitto. Dimana mosquitto merupakan salah satu

opensource broker pesan yang mengimplementasikan protokol MQTT versi 3.1 dan 3.1.1. Broker mosquitto juga mendukung implementasi *server lightweight* dari MQTT maupun MQTT-SN. Dari hasil pengujian yang telah dilakukan sebelumnya broker mosquitto dapat mendukung 100.000 koneksi secara bersamaan (Eclipse. 2013). Untuk arsitektur dari MQTT, dapat dilihat pada Gambar 3 :



Gambar 3 Arsitektur IoT (https://www.researchgate.net/figure/MQTT-architecture_fig2_319404271)

Penjelasan untuk Gambar arsitektur MQTT adalah MQTT *client* akan mempublish atau mengirimkan pesan pada MQTT broker lalu akan di *subscriber* oleh MQTT server. Dan untuk format pengiriman pesan pada MQTT, seperti Gambar 4 :



Gambar 4 Format Pesan MQTT (Egli. 2017)

Pada Gambar diatas terdapat 4 bagian yaitu *Message Type*, DUP, *Quality of Service (QoS) Level*, Retain. Dimana untuk *message type* adalah adalah tipe pesan pada MQTT. Kemudian DUP adalah duplikat pesan yang dikirimkan, jika terjadi kegagalan pada pengiriman pesan, maka DUP akan dikirimkan kembali oleh *publisher*. Selanjutnya terdapat 3 *QoS level* pada MQTT yaitu *at most once devlivery* atau *QoS level 0* dimana pada *QoS level 0* ini respon dari klien atau jawaban tidak diminta dan tidak ada pengiriman pesan ulang. *at least once delivery* atau *QoS level 1* pada *QoS level 1* ini jika terjadi kegagalan pengiriman pesan maka *publisher* akan mengirimkan pesan kembali dengan menyertakan bit DUP. *exactly once delivery* atau *QoS level 2* digunakan ketika duplikat pesan yang dikirimkan oleh *QoS level 1* tidak diterima oleh *server* dan retain digunakan untuk memerintahkan *server* menyimpan pesan untuk topik yang di*publish* sehingga jika pada topik dan nilai data yang sama maka pesan tidak akan dikirimkan jadi klien akan mempublish kembali jika terjadi perubahan data, untuk lebih lengkap tentang penjelasan format pesan MQTT dapat dilihat Gambar 5.

Message fixed header field	Description / Values	
Message Type	0: Reserved	8: SUBSCRIBE
	1: CONNECT	9: SUBACK
	2: CONNACK	10: UNSUBSCRIBE
	3: PUBLISH	11: UNSUBACK
	4: PUBACK	12: PINGREQ
	5: PUBREC	13: PINGRESP
	6: PUBREL	14: DISCONNECT
	7: PUBCOMP	15: Reserved
DUP	Duplicate message flag. Indicates to the receiver that this message may have already been received. 1: Client or server (broker) re-delivers a PUBLISH, PUBREL, SUBSCRIBE or UNSUBSCRIBE message (duplicate message).	
QoS Level	Indicates the level of delivery assurance of a PUBLISH message. 0: At-most-once delivery, no guarantees, «Fire and Forget». 1: At-least-once delivery, acknowledged delivery. 2: Exactly-once delivery. Further details see MQTT QoS.	
RETAIN	1: Instructs the server to retain the last received PUBLISH message and deliver it as a first message to new subscriptions. Further details see RETAIN (keep last message).	
Remaining Length	Indicates the number of remaining bytes in the message, i.e. the length of the (optional) variable length header and (optional) payload. Further details see Remaining length (RL).	

Gambar 5 Deskripsi format pesan MQTT (Egli. 2017)

Redis

Redis merupakan penyimpanan struktur data yang dapat digunakan sebagai database, *cache*, broker pesan dan antrian. Redis mendukung tipe data seperti string, hash, list, set, *sorted set*, bitmap, *hyperloglog*. Adapun contoh penerapan redis yang sering digunakan adalah *caching*, obrolan, perpesanan dan antrian, *leaderboard*, penyimpanan dengan *session*, *streaming*, geospasial, *machine learning*, analisis *real-time*. Adapun struktur penggunaan redis yaitu *string*, *list*, *set*, *hash*, zset (Carlson, 2013). Pada penelitian ini akan menggunakan redis untuk caching dan menggunakan struktur *list*. Untuk list akan melakukan pengimputan data berdasarkan urutan masuk dan berdasarkan index yang diberikan. Perintah yang dapat dilakukan pada *list* adalah RPUSH (digunakan untuk memasukkan nilai), LRANGE (digunakan untuk melihat jarak list atau jumlah data), LINDEX (digunakan untuk mengambil data berdasarkan *index*), LPOP (menghilangkan/menghapus data).

Redis adalah pilihan tepat untuk mengimplementasikan *cache* dalam memori yang tersedia. Sangat baik untuk mengurangi latensi akses data, meningkatkan *throughput*, Redis dapat melayani item yang sering diminta pada waktu respons di bawah satu milidetik, dan memungkinkan untuk secara mudah menskalakan muatan yang lebih tinggi. *Cache* hasil kueri database, *cache* sesi persisten, *cache* halaman web, dan *cache* objek yang sering digunakan seperti gambar, file, dan metadata semuanya merupakan contoh penggunaan *cache* dengan Redis (aws.amazon.com).

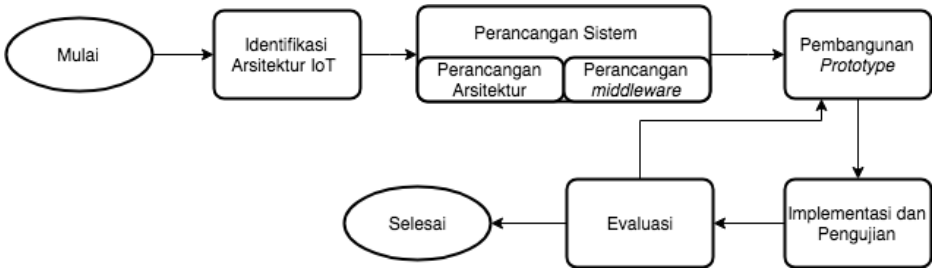
3 METODE

Tempat Penelitian

Penelitian ini dilaksanakan di *green house* Departemen Ilmu Komputer FMIPA Institut Pertanian Bogor.

Tahapan Penelitian

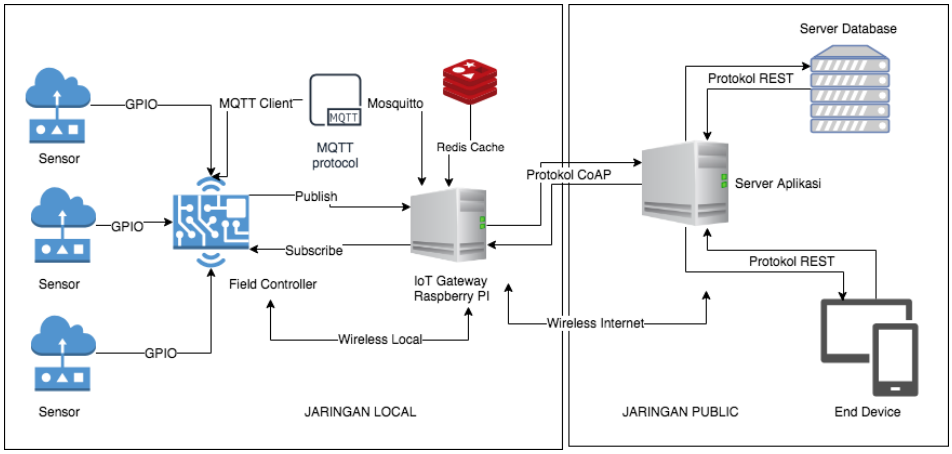
Tahapan penelitian terdiri dari beberapa proses seperti ditunjukan pada pada Gambar 6.



Gambar 6. Tahapan Penelitian.

Tahapan penelitian tersebut memiliki 4 tahapan dengan penjelasan sebagai berikut :

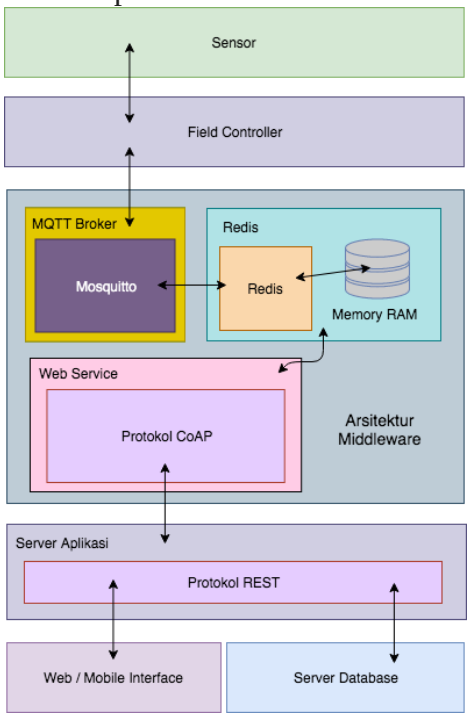
1. Identifikasi Arsitektur IoT
   Untuk mengidentifikasi permasalahan arsitektur pada penerapan IPB *smart farming*, dengan cara membaca literatur dan permasalahan utama dari IoT sebagai acuan untuk perancangan Arsitektur baru.
2. Perancangan Sistem
   Pada proses perancangan sistem dibagi menjadi 2 yaitu perancangan arsitektur dan perancangan *middleware*. Adapun penjelasan tentang perancangan tersebut adalah
   - Perancangan arsitektur
     Penjelasan tentang rancangan arsitektur yang dibuat adalah sensor-sensor yang dibutuhkan pada hidroponik akan terhubung dengan NodeMCU lalu NodeMCU akan mengirimkan data yang didapatkan dari sensor ke IoT *gateway* dengan menggunakan protokol MQTT, IoT *gateway* menggunakan Mosquitto akan melakukan *subscribe* pada *topic* yang di *publish* oleh MQTT *client* pada NodeMCU di jaringan lokal, lalu sistem akan mengecek apakah koneksi internet tersedia atau tidak jika terkoneksi maka data akan dikirimkan langsung ke server aplikasi menggunakan protokol CoAP, jika koneksi internet tidak tersedia maka data akan disimpan ke redis untuk sementara sampai koneksi internet tersedia jika telah tersedia data akan dikirimkan ke server aplikasi menggunakan protokol CoAP. Setalah data tersedia di server aplikasi maka data akan disimpan ke dalam server *database* menggunakan protokol REST. Dalam hal ini server aplikasi juga bertugas untuk melayani penglolaan proses eksternal yaitu permintaan dan pengiriman data untuk *end user* menggunakan protokol REST, seperti ditunjukkan pada Gambar 7.



Gambar 7. Rancangan Arsitektur IoT

- Perancangan *Middleware*

  Rancangan pada arsitektur *middleware* yang akan diterapkan pada arsitektur IoT terdapat MQTT broker menggunakan Mosquitto, *cache memory* menggunakan Redis serta protokol CoAP dan REST, untuk lebih lengkap dapat dilihat pada Gambar 8.



Gambar 8. Arsitektur Middleware.

3. Pembangunan *Prototype*

   Pada proses ini akan dilakukan pembangunan *prototype* pada alat IoT pada studi kasus pertanian hidroponik yaitu menggunakan pH, TDS, DS 18, DHT 22. Kemudian untuk IoT *gateway* menggunakan Raspberry pi 3 dan server *database* menggunakan mongoDB serta pada tahapan ini akan dilakukan penentuan parameter arsitektur IoT serta konfigurasi *cache memory* redis.

4. Implementasi dan Pengujian

Proses ini akan mengimplementasikan arsitektur yang telah dibuat untuk hidroponik pada *green house* Departemen Ilmu Komputer FMIPA Institut Pertanian Bogor serta akan dilakukan pengujian, adapun parameter yang akan diuji adalah mengukur kestabilan (*packet loss*) dan mengukur jumlah *byte* per paket terkirim. Pengujian ini akan dilakukan pada protokol MQTT dan REST, kemudian protokol MQTT dan CoAP baik menggunakan redis maupun tidak menggunakan redis serta menguji protokol *overhead* pada MQTT. Pengujian tersebut dilakukan pada dua lingkungan yang berbeda, yaitu :

a. Lingkungan normal yaitu pengujian mengguanakan data nyata dari sensor.
b. Lingkungan tidak normal dengan menggunakan *stress test* yaitu menggunakan data sintetik sesuai *variable* dan *tipe* data dari *sensor*, dengan cara menambahkan data secara berkala untuk mengetahui batas maksimal protokol MQTT.

5. Evaluasi Parameter

Dalam tahapan ini akan mengevaluasi hasil dari penerapan parameter yang dilakukan untuk implementasi dan pengujian. Proses ini akan kembali ketahap pembangunan *prototype* sesuai dengan ketentuan yang diinginkan untuk mendapatkan pengaturan parameter yang terbaik.

4 JADWAL PENELITIAN

Penelitian dijadwalkan mulai Januari 2019 dan berakhir Desember 2019 dapat dilihat pada Tabel 1.

Tabel 1 Jadwal Penelitian

No	Kegiatan	2019											
		1	2	3	4	5	6	7	8	9	10	11	12
1	Penyusunan Proposal												
2	Sidang Komisi 1												
3	Kolokium												
4	Perancangan Arsitektur												
5	Penelitian												
6	Sidang Komisi 2												
7	Publikasi												
8	Seminar Hasil												
9	Sidang Tesis												
10	Perbaikian Tesis												

DAFTAR PUSTAKA

Amandeep K, Ashish J. 2017. Health Monitoring Based on IoT using Raspberry Pi. *International Conference on Computing, Communication and Automation (ICCCA).* ISBN:978-1-5090-6471-7.

Anatolijs Z, Nadezsa K, Kaspars K, Antons P, Leonids R, Jerker D. 2018. MQTT Service Broker for Enabling the Interoperability of Smart City Systems. *Energy and Sustainability for Small Developing Economies (ES2DE)*. Doi : 10.1109/ES2DE.2018.8494341.

Azamat Z, Zhulduz S, Rassim S, Zhazira K. 2017. IoT smart campus review and implementation of IoT applications into education process of university. *International Conference on Electronics, Computer and Computation (ICECCO).* Doi : 10.1109/ICECCO.2017.8333334.

Bharati W, Kirti W, Ramith N, Amala N. 2018. Comparison with HTTP and MQTT In Internet of Things (IoT). *International Conference on Inventive Research in Computing Applications (ICIRCA)*. Doi : 10.1109/ICIRCA.2018.8597401.

Chirtopher C, David C. 2015. An IoT Smart Home Architecture for Long-Term Care of People with Special Needs. *World Forum on Internet of Things (WF-IoT)*. Doi : 10.1109/WF-IoT.2015.7389126.

Daphney, Stavrouda Z. 2016. Sequential Decision–Making in Healthcare IoT: Real–Time Health Monitoring, Treatments and Interventions. *World Forum on Internet of Things (WF-IoT)*. Doi : 10.1109/WF-IoT.2016.7845446.

Eclipse. 2013. Mosquitto. *Eclipse*. [diakses 17 April 2019]. Tersedia pada : https://www.eclipse.org/proposals/technology.mosquitto/.

Falconer G, Mitchell S. 2012. Smart City Framework: A Systematic Process for Enabling Smart+Connected Communities. San Jose (US): *Cisco Internet Business Solutions Group (IBSG)*.

Henri W, Gerhard P. 2018. Comparison between COAP and MQTT - Server to Business System level. *Conference: 2018 Wireless Advanced (WiAd).* Doi : 10.1109.

Jessy RW, Eko SP, Heru N. 2018. Implementasi Cluster Message Broker Sebagai Solusi Skalabilitas Middleware Berbasis Arsitektur Publish-Subscribe pada Internet of Things (IoT). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*. Vol. 2, No. 12, Desember 2018, pp.6861-6867.

Jorge EL, Marco Z, Juan CC, Carlos C, Pietro M. 2017. A Disruption Tolerant Architecture based on MQTT for loT Applications. *Consumer Communications & Networking Conference (CCNC).* Doi : 10.1109, ISSN : 2331 – 9860.

Josiah LC. 2013. *Redis in Action*. Shelter Island(NY): Manning Publications Co.

Keyur KP, Sunil MP. 2016. Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges. *International Journal of Engineering Science and Computing*. ISSN : 2321 3361, Vol. 6 Issue 5.

Livari A, Koivusaari J. 2016. A RESTful Sensor Data Back–end for the Internet of Things. *The Sixth International Conference on Advanced Communications and Computation*; 2016 May 22–26; Valencia, Spain. Hlm 51–55.

Mohd. Saifuzzaman, Ashraf HK, Nazmun N, Fernaz NN. 2017. Smart Security for an Organization based on IoT. *International Journal of Computer Applications*. Volume 165, No.10, May 2017.

Narayan P. 2015. Overview Of Redis Architecture. Tersedia pada : http://qnimate.com/overview-of-redis architecture/#Redis_Single_Instance_Architecture.

Nagesh UB, Uday DV, Shamitha GT, Pooja S. 2017. Application of MQTT Protocol for Real Time Weather Monitoring and Precision Farming. *International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT)*. Doi : 10.1109/ICEECCOT.2017.8284616.

Nitin N, 2017. Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP. *International Systems Engineering Symposium (ISSE)*. Doi : 10.1109/SysEng.2017.8088251.

Nusantara MF, Akbar S, Rachmadi A. 2016. Analisa Metode Publish/subcribe untuk komunikasi data antar perangkat dalam lingkungan smarthome. *Repositori Jurnal Mahasiswa PTIIK UB*. Vol. 8 No. 15.

Peter RE. 2017. An Introduction To MQTT, a Protocol For M2M and IoT Application. *Peteregli.net* [Internet]. [diunduh 17 April 2019]. Tersedia pada : http://peteregli.net/content/iot/MQTT/MQTT.html.

Saurabh V, Prashant A, Santosh B. 2018. Industry 4.0 – A Glimpe. *2nd International Conference on Materials Manufacturing and Design Engineering*. Procedia Manufacturing 20 : 233-238.

Shelby Z, Hartke K, Bormann C. 2014. The Contrained Application Protocol (CoAP). *Internet Engineering Task Force (IETF)*. ISSN : 2070-1721.

Shulong W, Yibin H, Fang G, Xinrong J. 2016. A Novel IoT Access Architecture for Vehicle Monitoring System. *World Forum on Internet of Things (WF-IoT).* Doi : 10.1109/WF-IoT.2016.7845396.

Solace. 2019. MQTT Topic. *Solace*. [diakses 17 April 2019]. Tersedia pada : https://docs.solace.com/Open-APIs-Protocols/MQTT/MQTT-Topics.htm.

Sri W, Akhyar W. 2017. Komiot: Exploring Rest Protocol for IoT Server of the Automatic Control System for Production Lan Irrigation. Proc. The 4th ISS : pp. 71-81.

Sri W, Ardhi M, Tatag B. 2016. The Fuzzy Infrernce System for Intelligent Water Quality Monitoring System to Optimize Eel Fish Farming. *International Symposium on Electronic and Smart Devices(ISESD).*

Suresh J, Vijay D, Parthasarthy V. 2014. A state of the art review on the Internet of Things (IoT). *International Conference on Science, Engineering and Management Research*. Doi : 10.1109/ICSEMR.2014.7043637.

Valerie L, Weng TL, Leonardo O, Sweta R, Nagesh S, Rong X. 2012. Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry [Internet].[diunduh 17 April 2019].1st edition : USA.

YuHwan K, Namgu L, ByeongJun K, KyooJae S. 2018. Realization of IoT based Fish Farm Control Using Mobile App. *2018 International Symposium on Computer, Consumer and Control (IS3C)*. Doi : 10.1109/IS3C.2018.00055.