**Jurnal Transistor ElektrodanInformatika(TRANSISTOR EI)**
Vol. 3, No. 1, Mei 2018, pp. 9~20



# Sistem IoT Terintegrasi Menggunakan Flow Based Programming dengan Protokol MQTT dan Time Series DB

**Sri Mulyono, Moch Taufik , Mohammad Taufiqurrohman**
Jurusan Teknik Informatika, Universitas Islam Sultan Agung

*Correspondence Author: sri.m@unissula.ac.id*

## Abstrak

*Internet of Thing (IoT) dewasa ini semakin populer dan menjadi perhatian berbagai pihak, pada tahun 2010 Ericsson memperkirakan akan ada 50 juta perangkat yang terhubung dengan internet.[1]. Penurunan biaya komputasi sebesar 60 kali lipat, penurunan biaya bandwidth 40 kali lipat dan penurunan biaya sensor dan alat cerdas lainnya yang hampir 2 kali lipat dalam dekade terakhir ini [2]juga turut mempopulerkan Internet of Thing.Sehingga sistem IoT terintegrasi dibutuhkan untuk mengakomodasi dan memaksimalkan perangkat IoT, protokol MQTT menjadi protokol yang dapat diandalkan menangani massiveupdate dalam pengiriman data IoT[3] dan TSDB menjadi platform data yang cocok untuk data IoT, dari uraian tersebut penelitian ini akan mencoba membuat sistem IoT yang mengintegrasikan MQTT dengan TSDB dan dari TSDB akan diolah menjadi visualisasi grafik untuk memudahkan memonitor dan menganalisis. Penelitian ini diterapkan di server Raspberry Pi3 (RPI) dan AWS, menyesuaikan dari kebutuhan bisnis. Hasilnya sistem IoTmampu digunakan sebagai sistem untuk aplikasi monitor suhu dan kelembaban yang berjalan 10 jam non-stop dan mampu menangani beban hingga 1000 clients (untuk AWS dengan 1GB RAM dan 993 clients untuk RPI)*

*Keyword: Internet of Thing* (*IoT*), MQTT, TSDB

## 1. PENDAHULUAN

*Internet of Thing* sendiri secara sederhana bukanlah hanya *internet connected thing* namun juga secara berkala mengirim data/nilai dari sensor maupun perangkat pintar lainnya ke *IoTserver.*Nilai dari sensor atau perangkat pintar nantinya akan dikirimkan terus menerus secara berkala namun hal tersebut dapat memberatkan terhadap *server* (*server* yang digunakan untuk menampung data dari *sensor*) karena *server* harus melayani *request* dalam jumlah yang banyak dan membutuhkan *resource* memori yang besar, hal ini jika terjadi secara terus menerus akan membuat *server* menjadi *down*untuk itulah digunakan protokol MQTT (*Message Queuing Telemetry Transport*) untuk menggantikan *http request* yang ada dikarenakan lebih *lightweight* atau ringan dalam hal *resource* yang dibutuhkan dan dapat menangani *post request* (dalam MQTT *post request* disebut dengan istilah *publish*) lalu dari sisi klien nya juga lebih hemat *resource* karena tidak perlu melakukan *polling* secara terus menerus digantikan menggunakan metode *subscribe*, di mana ketika ada data baru *server* akan mengirimkan data tersebut terhadap klienyang melakukan *subscribe* sehingga ketika tidak terdapat data baru maka klien tidak perlu melakukan *polling* secara berkala.[4]

Di sisi lainnya data yang telah dikirimkan melalui protokol MQTT perlu untuk disimpan ke dalam *database* untuk nantinya dapat digunakan untuk keperluan analisis dan pengambilan keputusan berdasarkan data historis, untuk dapat mencapai hal tersebut dibutuhkan sistem *database* khusus yang mampu menangani proses *write* secara *continue* berkala terus menerus, karena tingkat *write* yang tinggi tersebut basis data relasional biasa seperti mysql kurang cocok digunakan dan *Time Series Database* (TSDB) menjadi solusi yang lebih rasional untuk data sensor yang merupakan data berdasarkan waktu (*time series*/historis),[5] dari alasan tersebut dipilihlah *time series database* InfluxDB yang secara teknis mengungguli Cassandra DB sebanyak 4.5 kali lipat dalam kecepatan *write throughput*, dan menggunakan *resource* 10.8 kali lebih hemat penggunaan *storage* berkat *compression* nya[6]. InfluxDB juga mengungguli beberapa *timeseries database* lainnya dalam berbagai aspek.[7]

Dari beberapa alasan di atas, maka penelitian ini membahas bagaimana membuat sistem terintegrasi untuk *internet of thing* yang ringan, efisien dan maksimal untuk sistem *internet of thing* dengan menggunakan protokol MQTT dan TSDB, sedangkan untuk mengintegrasikan MQTT dengan InfluxDB

menggunakan *flow based programming* dari Node-Red kemudian untuk mempermudah menganalisis data dari sensor tadi dapat divisualisasikan dalam bentuk grafik, tabel maupun *chart* dengan menggunakan Grafana Server maupun Node-Red *Dashboard*.

Penelitian ini akan mencoba merancang dan membangun sistem *IoT*terintegrasi pada Raspberry Pi 3 model B dan VPS (*Virtual Private Server*) dari Amazon AWS EC2 T2.*micro*yang mana masing masing infrastruktur tersebut mempunyai kelebihan dan kekurangan masing masing, seperti Raspberry Pi 3 model B yang lebih ringkas dan murah yang memiliki kemampuan komputasi yang cukup untuk penerapan sistem *IoT* porsi kecil sampai sedang dan bersifat lokal (diimplementasikan di *edge network*) sedangkan AWS EC2 memiliki komputasi yang lebih baik dibandingkan dengan Raspberry Pi model B sehingga cocok untuk sistem *IoT* yang lebih besar porsi nya dari Raspberry Pi, serta jaminan *uptime, dedicated IP* dan beberapa fitur lainnya yang membuat AWS EC2 untuk kebutuhan yang bersifat publik (diimplementasikan di *cloud*) sehingga dari mana saja dapat mengakses dan menggunakan sistem *IoT* dengan mudah asalkan terdapat jaringan internet.

## 2. METODE PENELITIAN

Metodologi penelitian dalam tugas akhir ini diklasifikasikan kedalam metode pengumpulan data dan metode pengembangan sistem.

a. Pengumpulan Data
   1. Kajian Pustaka: mempelajari berbagai macam pustaka daring, buku elekronik, jurnal-jurnal dan pustaka lainnya yang terkait dengan topik tugas akhir ini.
   2. Observasi: mengamati, mempelajari berbagai macam *project, software tools,* maupun penelitian yang sudah ada sebelumnya terkait topik tugas akhir ini.
   3. Sensor Feeding: percobaan pengumpulan *measurement data* dari sensor dengan cara menjalankan sistem secara kontinyu selama selang waktu tertentu ketika sistem sudah masuk tahapan *testing,* untuk kemudian menganalisis kapabilitas dari sistem untuk menjadi sistem*IoT*terintegrasi.

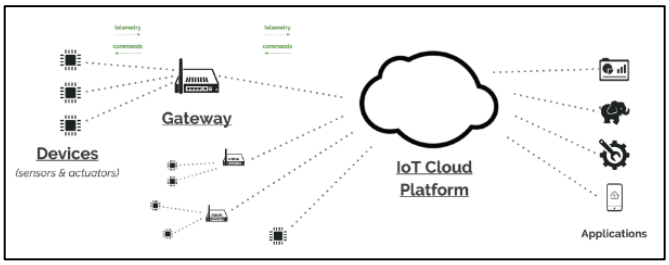
b. Pengembangan Sistem

Metode pengembangan sistem dalam penelitian ini yaitu *waterfall* dengan urutan tahapan berikut:

1. Studi Literatur dan Requirement Analysis
2. Desain Sistem
3. Pemrograman dan Implementasi
4. Pengujian dan Verifikasi
5. Perawatan Perbaikan dan Pembaharuan

## 3. HASIL DAN ANALISA

### 3.1. IoT *Architecture*




Gambar 1. IoT *Architecture* [8]

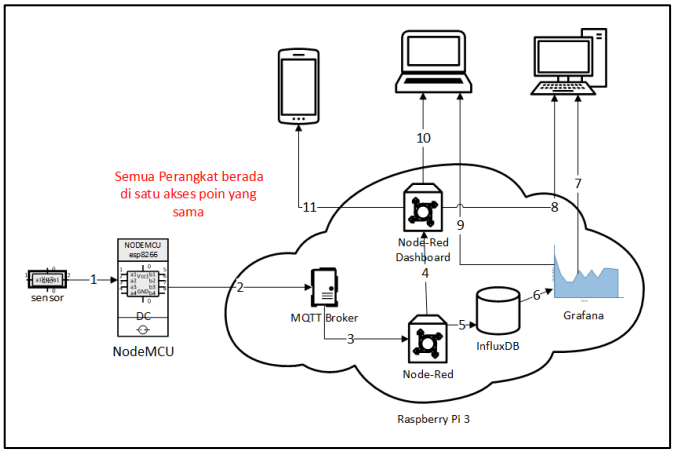
Menurut Peter Weher dalam bukunya yang berjudul *Learning Internet of Things* berpendapat "*The IoT is what we get when we connect Things*, *which are not operated by humans, to the Internet*". *Internet of Thing* adalah sesuatu yang didapatkan ketika menghubungkan *Things* atau benda-benda(*sensor, actuaror, constrained device etc*) yang mana tidak dioperasikan oleh manusia dan dihubungkan dengan internet.[9]

Sebuah kolaborasi dari Kelompok Kerja *IoT* Eclipse menjalaskan "Solusi *IoT* umumnya dikarakteristikkan oleh banyaknya perangkat (*things*) yang memanfaatkan *gateways* untuk berkomunikasi melalui suatu jaringan ke *back-end server* milik perusahaan penyedia yang di dalamnya beroperasikan *IoT platform* yang mengintegrasikan informasi yang diperoleh dari *IoT* sehingga dapat dimanfaatkan perusahaan tersebut. Peran penting *device, gateways and IoT Cloud Platform* didefinisikan dengan baik dan masing masingnya memiliki fitur dan fungsi yang dibutuhkan untuk solusi *IoT* yang handal"[8]. Pada gambar 1 terdapat gambaran umum tentang arsitektur dari solusi *IoT* yang handal, umumnya terdapat 3 atau bahkan 4 bagian penting dalam arsitektur *IoT* [8] yaitu:

1. *Devices: Sensors, Actuators etc*
2. *Gateway: Device* yang menghubungkan/menjembatani sensor dengan *IoT Cloud Platform e.g:* ESP8266, Raspberry Pi *etc.*
3. *IoT Cloud Platform:* Integrasi beberapa teknologi dan software yang menjadi pusat kontrol dan menejemen.
4. *Application: Specifically developed software* yang disajikan kepada *end-user e.g: Graph, Matrix etc.* Terkadang beberapa aplikasiini juga sudah termasuk kedalam *IoT Cloud Platform* namun terdapat juga yang terpisah berupa *mobile apps, desktop client apps* dan lainnya.

### 3.2. Model yang Diusulkan

Sistem *IoT* ini memiliki gambaran seperti gambar 2, untuk RPI semua komponen terhubung melalui jaringan lokal , sedangkan untuk aws komponen klien akan terhubung dengan cloud yang berada di aws melalui internet.

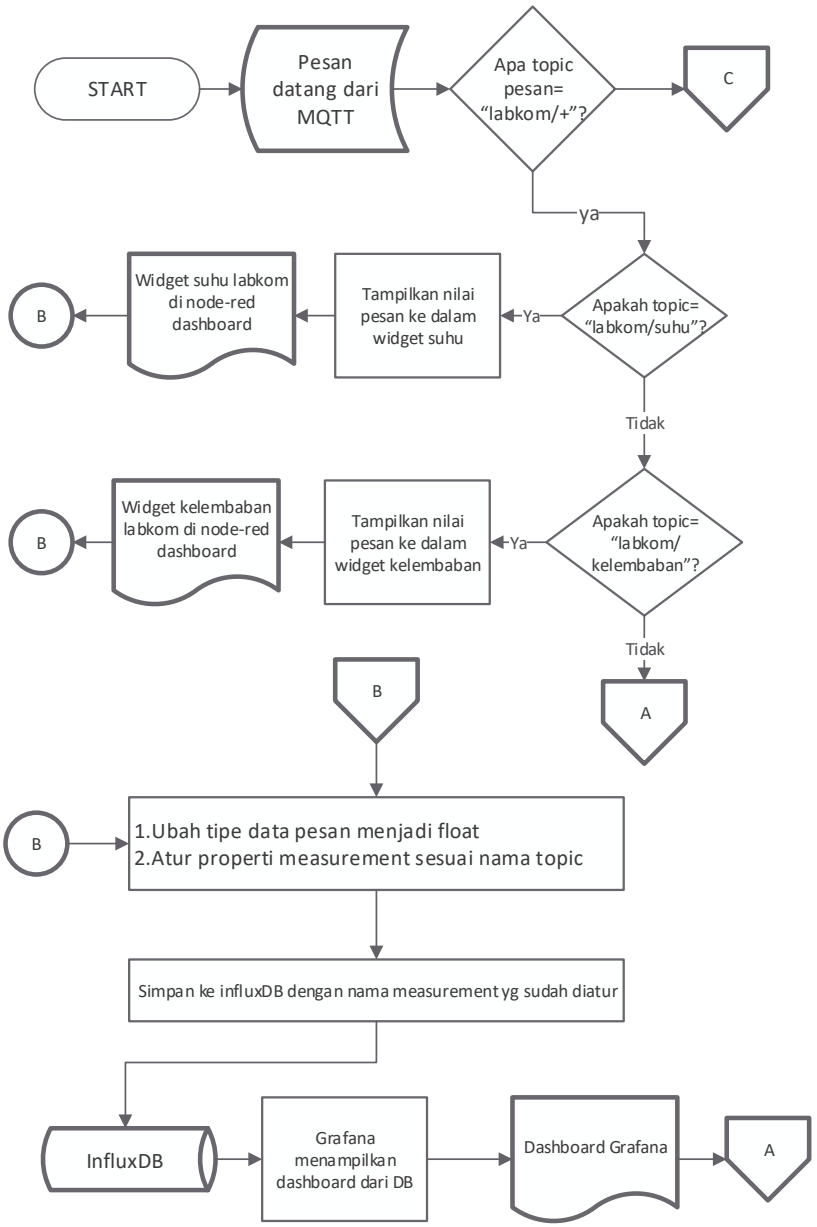



Gambar 2. Model yang diusulkan

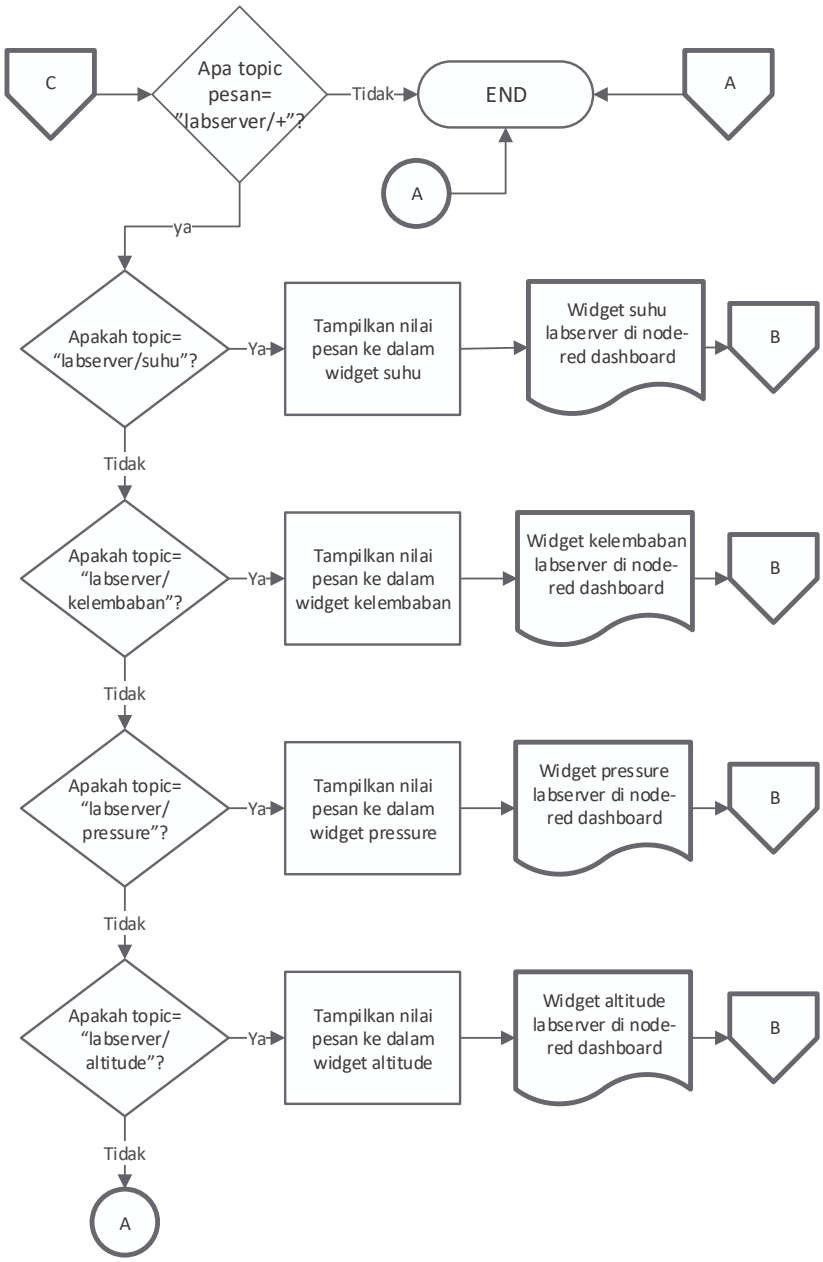
### 3.3. Analisa Kebutuhan

1. Kebutuhan Fungsional
   a. Sistem IoT harus bisa menerima data yang dikirimkan melalui protokol MQTT.
   b. Data yang diterima dari protokol MQTT harus bisa diteruskan ke Node-RED *Dashboard* secara *realtime.*
   c. Data yang diterima dari protokol MQTT harus bisa disimpan ke dalam *time series database* InfluxDB.
   d. Data *measurements* yang terdapat di InfluxDB harus bisa diolah (dicari nilai minimum, maksimum, rata rata, dan disederhanakan diambil *sample* baik itu harian, mingguan atau satuan waktu lainnya) dan divisualisasikan kedalam *graph/chart* maupun *table.*
2. Kebutuhan Non-Fungsional
   a. Perangkat Keras
      AWS EC2, raspberry pi 3, nodeMCU 1.0, sensor dht11 & bme280, komponen elektronika lainnya dan klien(*pc/mobile*).
   b. Perangkat Lunak
      SSH, EMQ, Node-RED, InfluxDB, Grafana, Arduino IDE.

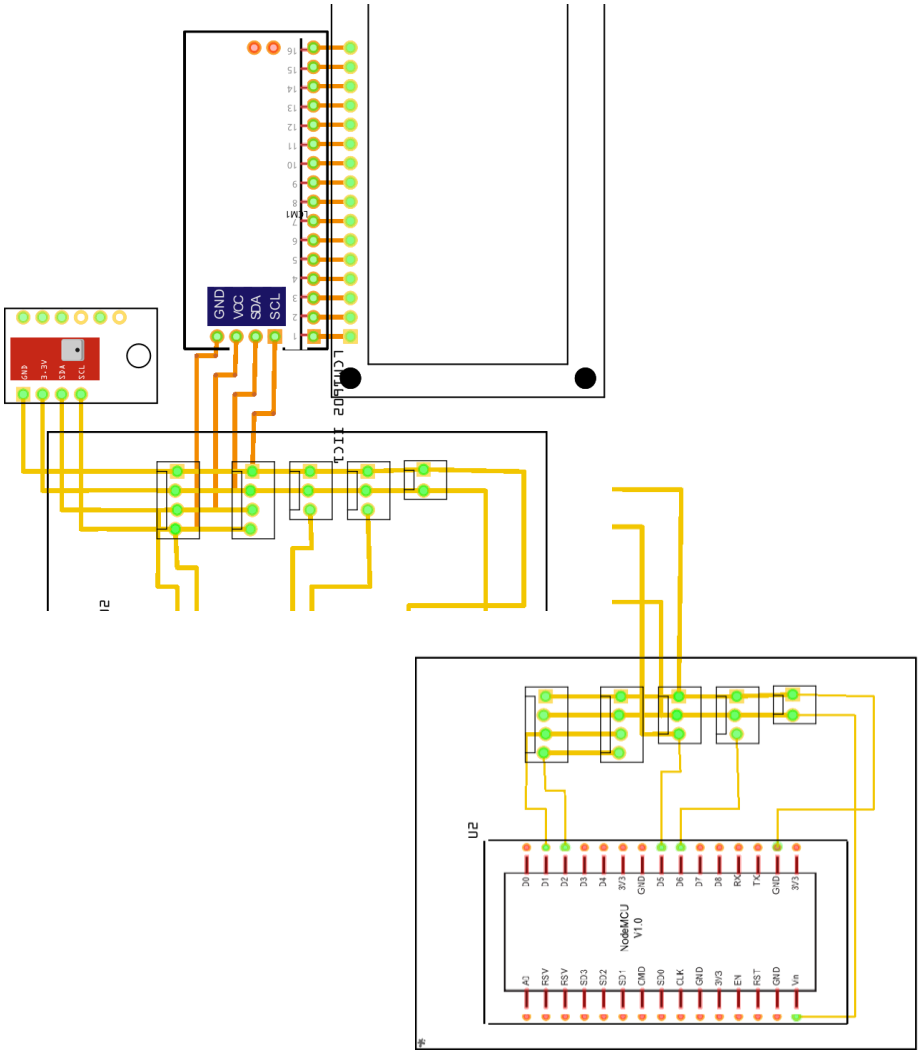
### 3.4. Desain



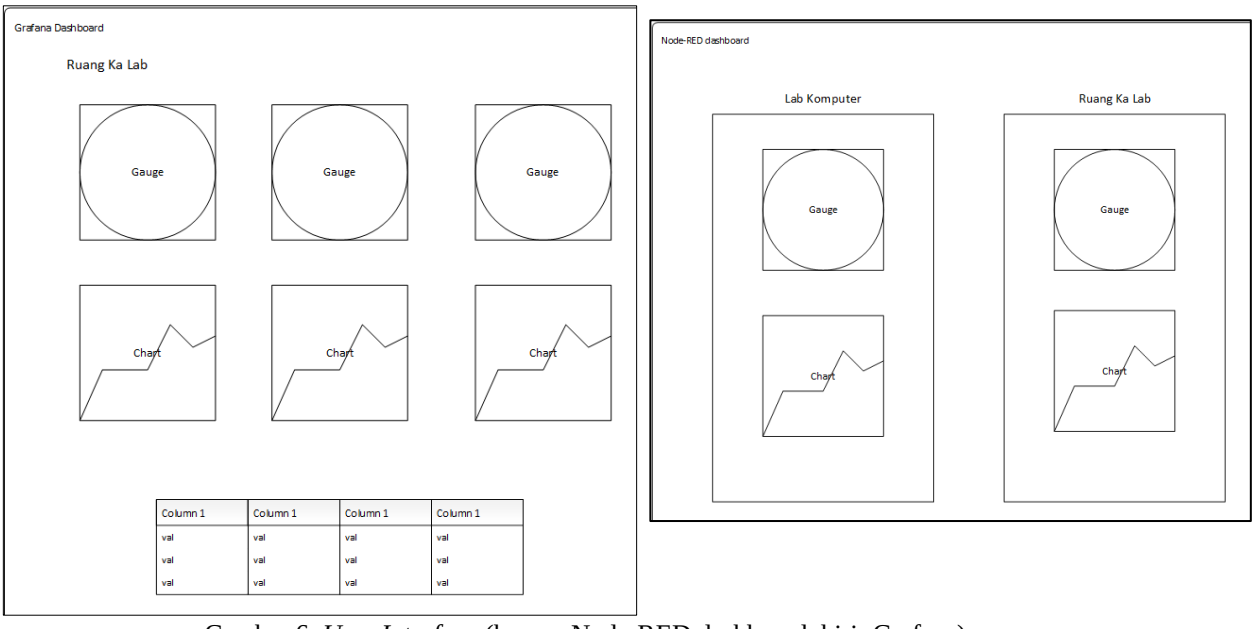

Gambar 3. *Flowchart* sistem *IoT* bagian 1



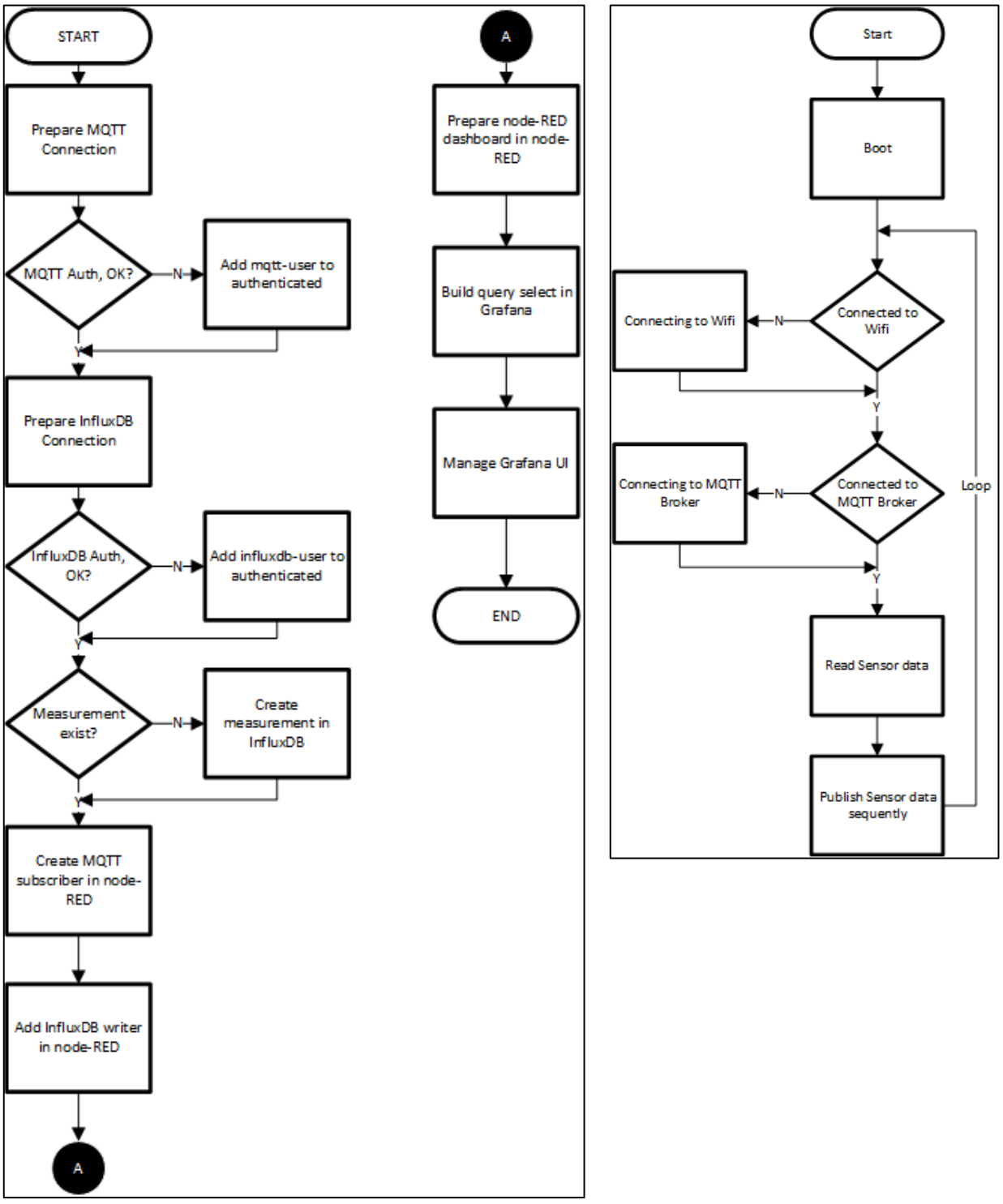

Gambar 4. *Flowchart* sistem *IoT* bagian 2




Gambar 5. Rangkaian Node MCU( kanan:NodeRobot, kiri:NodeAmica)




Gambar 6. *User Interface* (kanan: Node-RED dashboard, kiri: Grafana)




Gambar 7. *Flowchart* (kanan: Proses pada NodeMCU, kiri: Proses penggunaan sistem *IoT*)

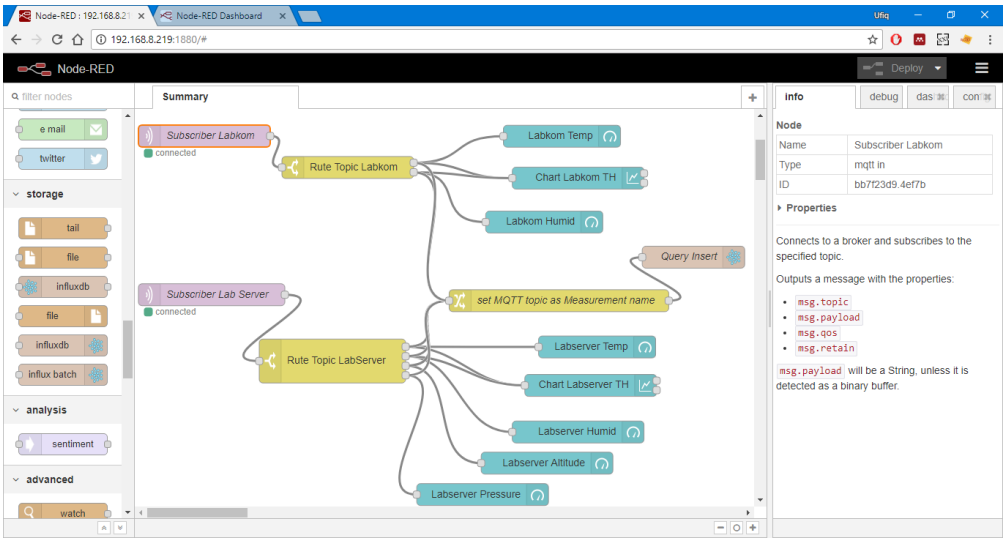
### 3.5. Test *Plan*

Berikut adalah rangkaian test *plan*:

1. Pengujian Jalannya Servis: Memeriksa servis di dalam sistem *IoT* sudah berjalan atau belum
2. Pengujian Fungsional : Menguji fungsi dari sistem *IoT* secara keberhasilan fungsionalitas
3. Penguian Beban: Menguji kapabilitas sistem *IoT* dengan beban *virtual client*

### 3.6. Implementasi Sistem

Implementasi sistem dapat dilihat pada gambar 8.




Gambar 8. *Flow* pada Node-RED

### 3.7. Pengujian Sistem

1. Pengujian Jalannya servis
   a. RPI

```
Bitvise xterm - pi.bscp - pi@192.168.8.219:22 - pi@minipink: ~
root@minipink:/home/pi# systemctl status emqttd
● emqttd.service - Erlang MQTT broker - build by mtrohman
   Loaded: loaded (/lib/systemd/system/emqttd.service; enabled)
   Active: active (exited) since Sun 2017-09-17 16:20:52 WIB; 24h ago
```

Gambar 9. EMQ pada RPI

```
root@minipink:/home/pi# systemctl status influxdb
● influxdb.service - InfluxDB is an open-source, distributed, time series databa
se
   Loaded: loaded (/etc/systemd/system/influxdb.service; enabled)
   Active: active (running) since Sat 2017-09-16 22:04:45 WIB; 1 day 19h ago
```

Gambar 10. Influx DB pada RPI

```
root@minipink:/home/pi# systemctl status nodered
● nodered.service - Node-RED graphical event wiring tool.
   Loaded: loaded (/lib/systemd/system/nodered.service; enabled)
   Active: active (running) since Mon 2017-09-18 10:10:41 WIB; 7h
```

Gambar 11. Node RED pada RPI

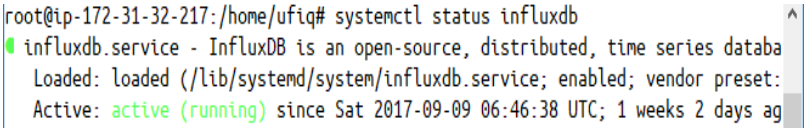
```
root@minipink:/home/pi# systemctl status grafana-server
● grafana-server.service - Grafana instance
   Loaded: loaded (/usr/lib/systemd/system/grafana-server.service; enabled)
   Active: active (running) since Mon 2017-09-18 17:19:02 WIB; 12s ago
```

Gambar 12. Grafana *Server*pada RPI

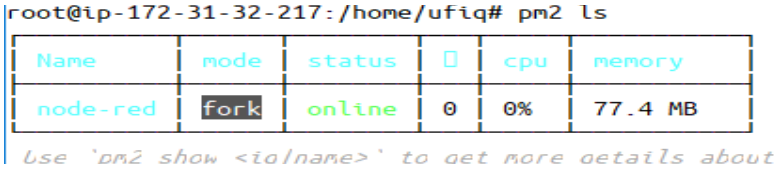
   b. AWS

```
Bitvise xterm - aws.bscp - ufiq@iot.mtrohman.com:22 - root@ip-172-31-32-217: /home/ufiq
root@ip-172-31-32-217:/home/ufiq# systemctl status emqttd
● emqttd.service - LSB: Erlang MQTT Broker
   Loaded: loaded (/etc/init.d/emqttd; bad; vendor preset: enabled)
   Active: active (running) since Sun 2017-09-03 15:04:45 UTC; 2 wee
```

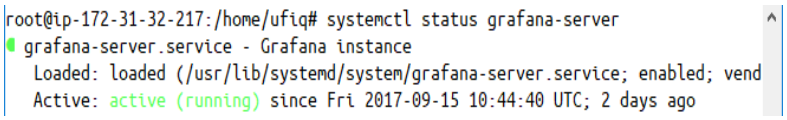
Gambar 13. EMQ pada AWS




Gambar 14. Influx DB pada AWS

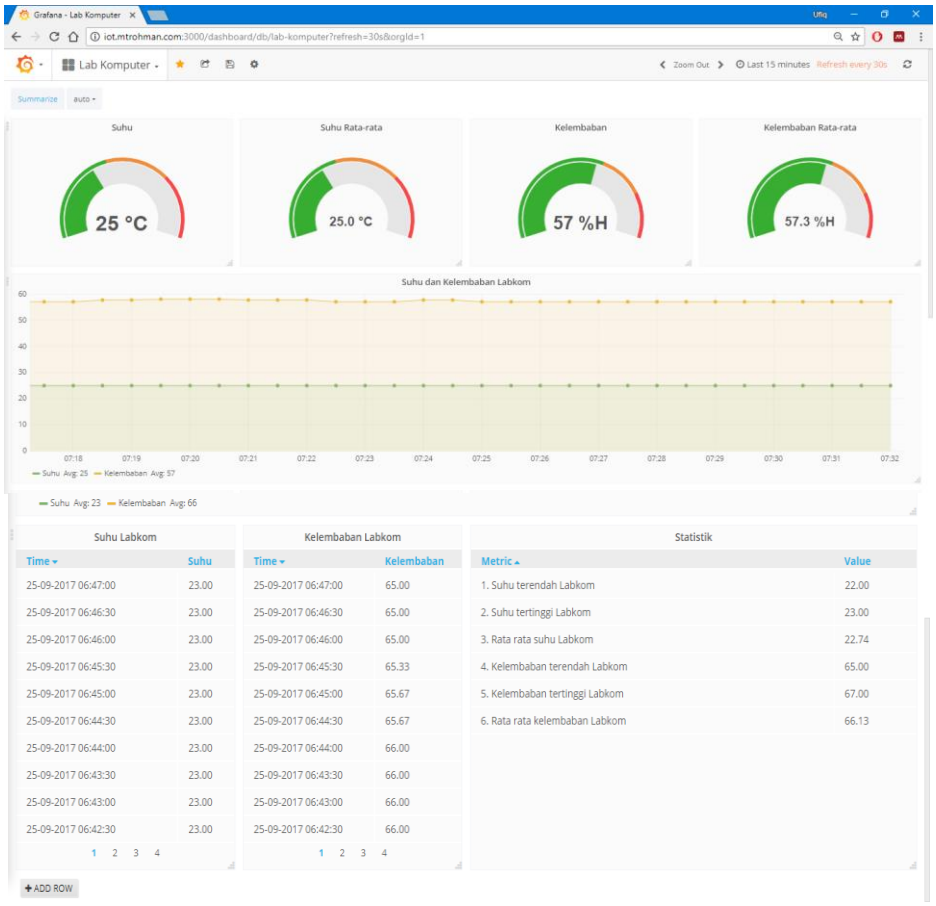



Gambar 15. Node-RED dalam PM2 *list* pada AWS

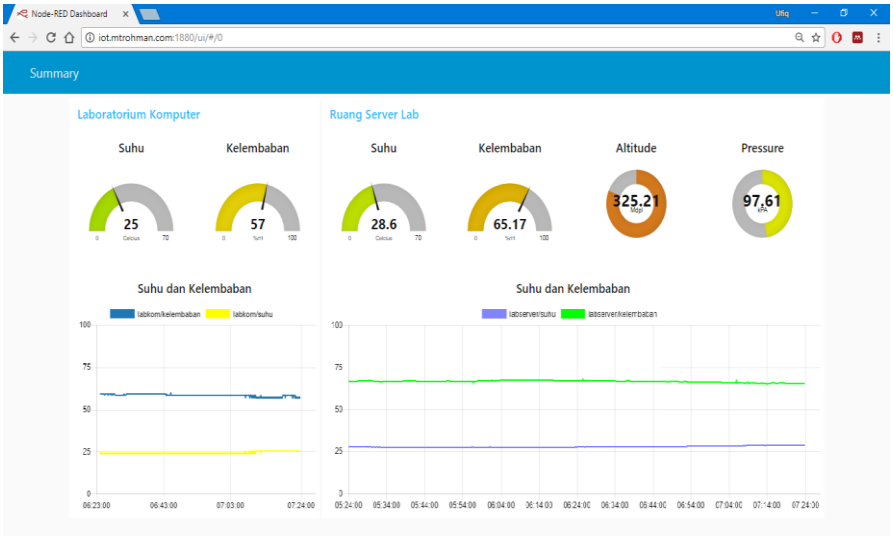



Gambar 16. Grafana pada AWS

2. Pengujian Fungsional
   a. Input: NodeMCU aktif mengirim data sensor selama 10 jam menggunakan MQTT
   b. Proses: Sistem *IoT* memproses data yang dikirimkan NodeMCU
   c. Output: Node-RED Dashboard dan Grafana menampilkan data yg telah diproses oleh Sistem *IoT* dan memvisualisasikannya.
   d. Hasil:



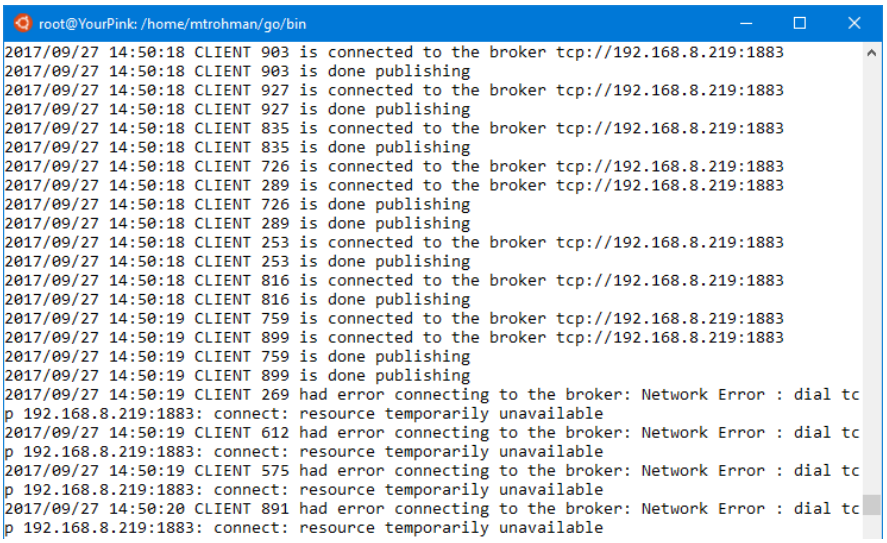

Gambar 17. Grafana *Dashboard* pada AWS



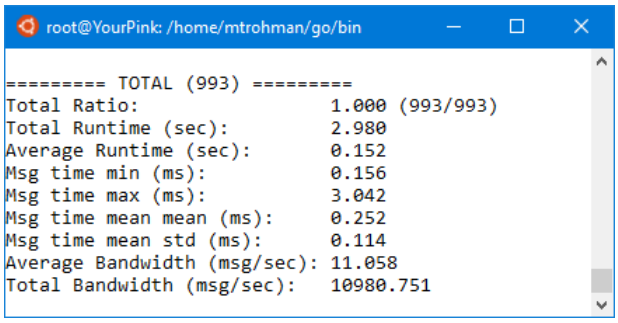

Gambar 18. Node-RED *Dashboard* pada AWS

3. Pengujian Beban

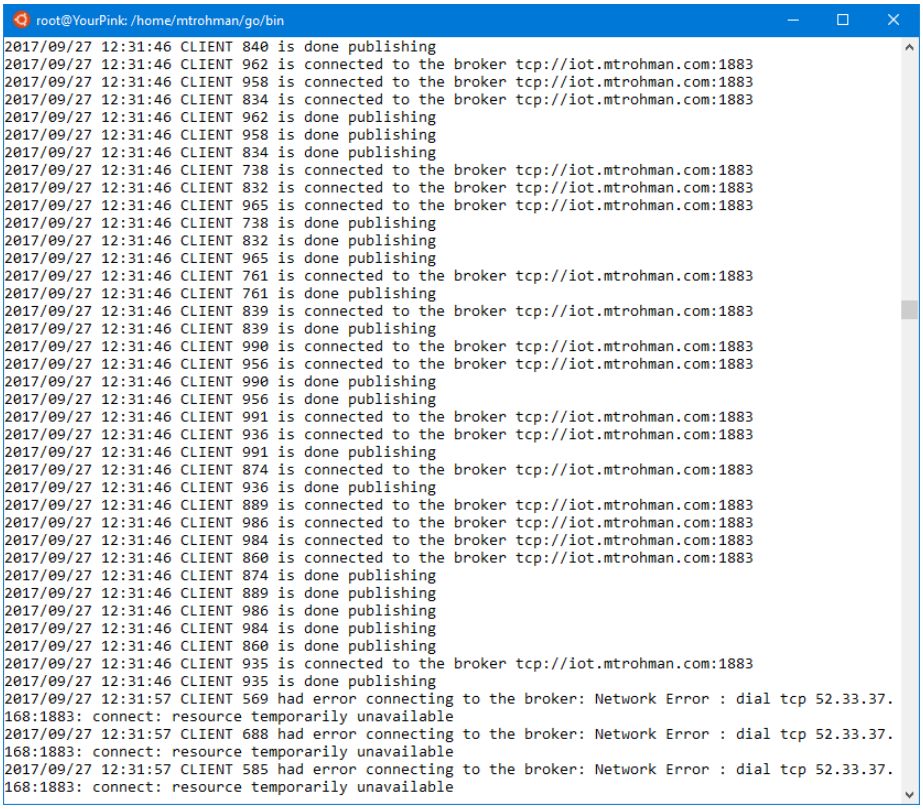
Pengujian beban dilakukan dengan memberi beban berupa *virtual mqtt publisher client*yang jumlahnya ditingkatkan secara bertahap, untuk jumlah *client*di bawah 500 baik itu RPI maupun AWS dapat menanganinya dengan baik, namun ketika ditingkatkan ke 1000, RPI tidak dapat meprosesnya, sehingga dicari batas maksimal jumlah client yang dapat terhubung ke RPI adalah 993, sedangkan untuk AWS setelah beberapa kali mencoba meningkatkan jumlah client ke 1000 koneksi MQTT ke AWS beberapa kali terputus namun pada akhirnya AWS mampu menangani 1000 client dengan penurunan performa (waktu yang dibutuhkan lebih lama).



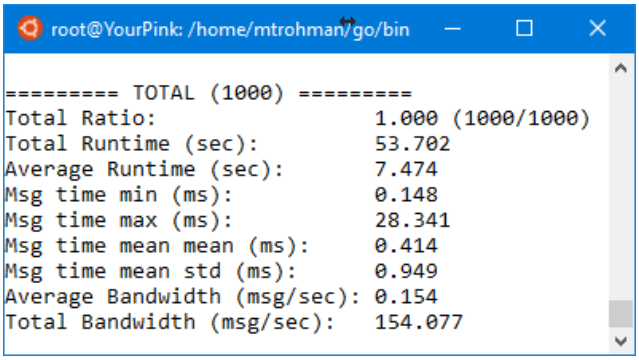

Gambar 19. Pengujian 1000 *client* gagal pada RPI




Gambar 20. *Client* maksimal yang dapat ditangani RPI




Gambar 21. 1000 *Client* gagal pada AWS (karena network latency)




Gambar 22. 1000 *client* berhasil ditangani AWS

## 4. KESIMPULAN

Setelah melaksanakan penelitian tugas akhir ini disimpulkan bahwa sistem *Internet of Things* yang mengterintegrasikan protokol MQTT dan *Time Series Database* telah berhasil dibuat menggunakan *flow based programming* Node-RED. Sistem *IoT* dapat diterapkan di Raspberry Pi 3 maupun Amazon *Web Service* atau perangkat dengan *OS* Linux, disesuaikan dengan kebutuhan dan dari serangkaian pengujian disimpulkan sistem *IoT* pada RPI dan AWS mampu menjadi sistem *IoT* terintegrasi untuk digunakan dalam aplikasi *IoT*. Sistem *IoT* pada RPI lebih ringkas dengan harga yang murah dan *network latency* yang rendah cocok digunakan untuk penerapan aplikasi *IoT* yang bersifat lokal dan untuk diakses *client* maupun perangkat pintar sampai sekitar 900 *clients*. Sedangkan sistem *IoT* pada AWS memberikan kemudahan akses, karena dapat diakses dari mana saja terlebih dengan diterapkannya *domain* akan lebih memudahkan mengaksesnya dari internet dan mendapat jaminan amazon untuk daya dan *broadband*. Kekuatan komputasi dan skalabilitas AWS yang dapat ditingkatkan (RAM, *processor*, *storage*, dll) jika dibutuhkan juga membuat AWS unggul. AWS dapat digunakan untuk aplikasi *IoT* yang membutuhkan kemudahan akses dari mana saja dan dapat menangani sampai dengan 1000 *clients (*dengan spesifikasi 1 vCPU + 1 RAM*)* yang bisa ditingkatkan lagi dengan meningkatkan spesifikasi *server*.

**DAFTAR PUSTAKA**

[1] H. Vestberg, "Ceo to shareholders : 50 billion connections 2020," *Press Release*, Stockholm, hal. 4–6, 13-Apr-2010.
[2] A. Datta, M. Ramakrishna, S. Verma, dan S. Joginapally, "Devices and Circuits to Address the Challenges in IOT," hal. 11.
[3] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, dan J. Alonso-Zarate, "A Survey on Application Layer Protocols for the Internet of Things," *Trans. IoT Cloud Comput.*, vol. 3, no. 1, hal. 11–17, 2015.
[4] M. Taufiqurrohman dan S. A. Firdanila Suparjo, "Sistem Monitor Suhu dan Kelembaban Berbasis IOT dengan Protokol MQTT di Laboratorium Farmasi UNISSULA," Semarang, 2017.
[5] J. Ganz, M. Beyer, dan C. Plotzky, "Time-series based solution using InfluxDB," no. i.
[6] T. Persen dan R. Winslow, "Benchmarking InfluxDB vs. Cassandra for Time-Series Data, Metrics & Management," *InfluxData Tech. Pap.*, 2016.
[7] P. Dix, "Why Time Series Metters for Metrics Real Time and Sensor Data," *InfluxData Whitepaper*, 2016.
[8] E. Iot dan W. Group, "The Three Software Stacks Required for IoT Architectures," *A Collab. Eclipse IoT Work. Gr.*, no. September, 2016.
[9] P. Waher, *Learning Internet of Things*. Packt Publishing Ltd, 2015.