BAB 3
METODE PENELITIAN

3.1 ALAT YANG DIGUNAKAN

Dalam penelitian ini dibutuhkan beberapa alat dan bahan, Adapun alat dan bahan yang digunakan dalam penelitia ini yaitu:

Tabel 3. 1 Alat dan Bahan

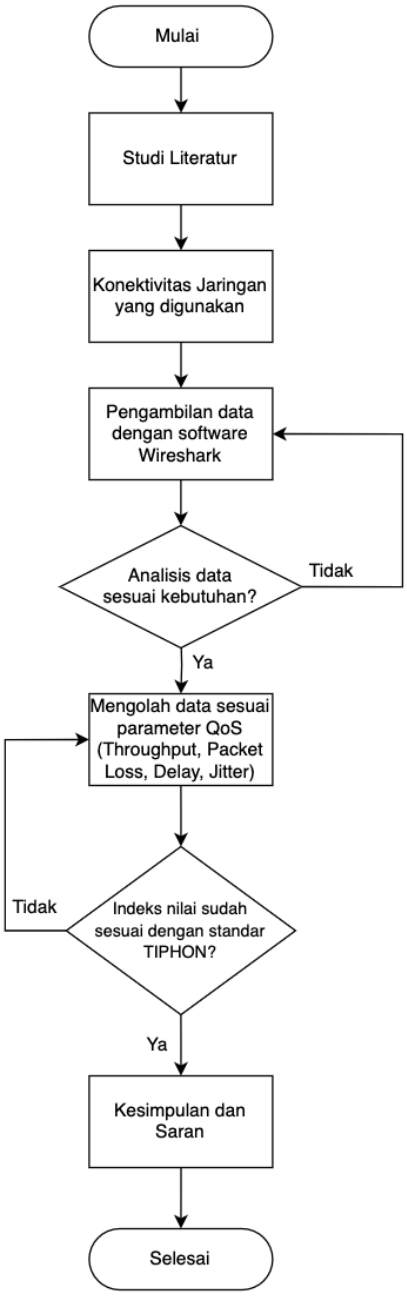
No	Alat dan Bahan
1.	Laptop Hp
2.	Sistem operasi Windows 11
3.	Prosesor AMD Rayzen 5
4.	Ram 8 GB
5.	Ms. Word 2019
6.	Ms. Excel 2019
7.	Jaringan Wi-Fi dan Kartu seluler
8.	Node.js
9.	*Console* Telkom IoT
10.	*Node-Red*

Dalam penelitian ini, kerangka penilaian kualitas layanan QoS dilakukan untuk memeriksa kinerja sistem dan QoS juga termasuk dalam *service-level agreement* (SLA) dengan penyedia layanan jaringan untuk menjamin tingkat kinerja tertentu. Penelitian ini dilakukan melalui berbagai tahap, mulai dari studi literatur, pengumpulan data, dan pencarian nilai-nilai QOS dengan menggunakan satnadar TIPHON serta pemantauan pada perangkat *hydrant* agar bekerja dengan baik dari aspek kualitas layanan (QoS). Standar TIPHON yang digunakan dalam penelitian ini yaitu parameter *throughput, delay, packet loss, dan jitter* [15].

Pengukuran TIPHON disimpulkan dalam indeks parameter QoS, dan tahap terakhir adalah tahap pembuatan hasil data dari hasil pengujian sistem. Salah satu

bentuk dari alur penelitian adalah *flowchart*, gambar dari *flowchart* menjelaskan prosedur alur penelitian yang dapat dilihat pada Gambar 3.2.

3.2 FLOWCHART ALUR PENELITIAN

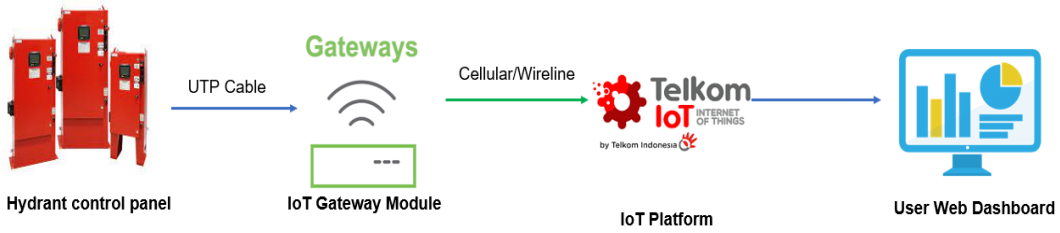


Gambar 3. 1 *Flowchart* Alur Penelitian

Pada Gambar 3.1 meggambarkan *flowchart* alur penelitian yang menjelaskan yaitu dimulai dari dari pencarian studi literatur yang dilakukan dengan menggunakan evaluasi penelitian teoritis dari perancangan sebelumnya. Selain itu studi literatur dilakukan dengan membaca buku analisis, jurnal ilmiah dan banyak artikel dari internet mengenai cara kerja dan sistem setiap perangkat pada alat yang digunakan dalam penelitian ini.

Flowchart menjelaskan alur dari konektivitas yang digunakan apakah sudah optimal, dengan dilanjutkan pengambilan data dengan menggunakan *software* wireshark. Hasil data dari wireshark nantinya akan di analisis apakah sudah sesuai dengan jumlah data yang di inginkan oleh peneliti atau belum sesuai. Hasil data yang sudah di analisis nantinya akan diolah sesuai dengan parameter QoS yang dicari, selanjutnya akan di hasilkan data dari parameter QoS sudah sesuai dengan indeks berdasarkan standar TR 101 329 V1.2.5.

3.3 ARSITEKTUR SISTEM

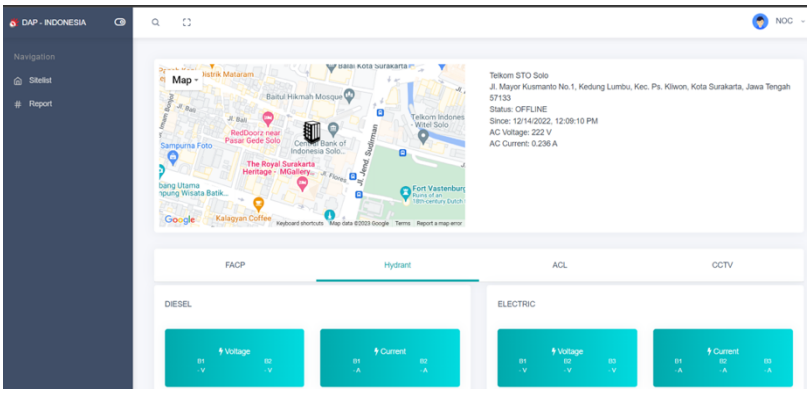


Gambar 3. 2 Arsitektur Sistem

Pada Gambar 3.2 menggambarkan arsitektur sistem, pada *hydrant control panel* di mana *controller* mempunyai spesifikasi *sharing* data melalui media kabel LAN. Selanjutnya, IoT perangkat menerima dan membaca data dari *controller fire alarm* yang diolah dengan menggunakan mini PC *Raspberry* sehingga kemudian meneruskan datanya ke IoT *Platform* melalui konektivitas jaringan 4G LTE / internet *wireline* [16].

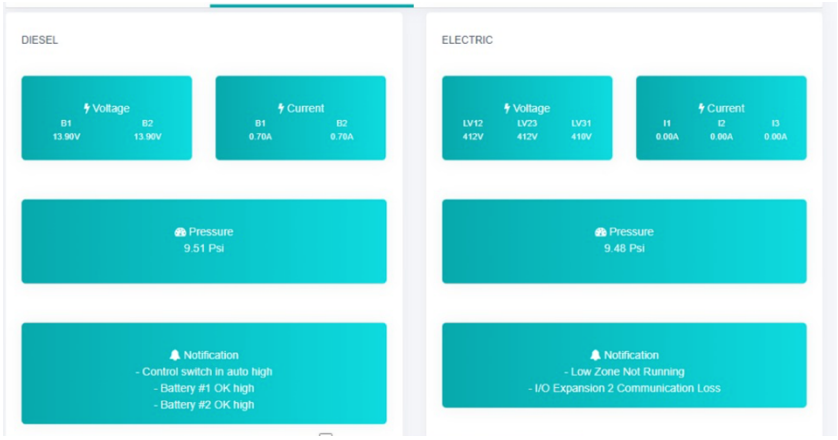
Data kemudian di simpan di *database platform* IoT untuk diteruskan ke *user web dashboard*. Pada tujuan akhirnya adalah emua data dari *fire alarm controller* akan ditampilkan di *user web dashboard* di mana di sana hanya bisa membaca sensor mana

yang hidup dan tidak bisa mengintervensi, contohnya mereset *alarm* yang sedang menyala.



Gambar 3. 3 Tampilan *dashboard hydrant* (a)

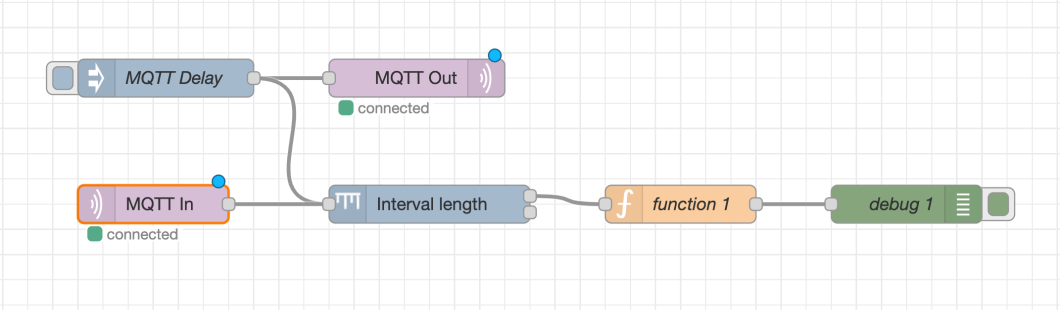
Pada Gambar 3.3 menggambarkan dapat dilihat tampilan *dashboard hydrant* di salah satu STO yaitu Telkom STO Solo.



Gambar 3. 4 Tampilan *dashboard hydrant* (b)

Gambar 3.4 menampilkan *dashboard hydrant* dengan menampilkan setiap status pada *diesel* dan *electric* yaitu *voltage* dan *current*.

3.4 SKENARIO SIMULASI



Gambar 3. 5 Skenario simulasi setiap node pada *node-red*

Pada Gambar 3.5 menggambarkan yaitu simulasi setiap *node* pada *node-red* yang telah di konfigurasi. Simulasi *node-red* dilakukan pada *flow* 2 dengan *node inject*, *debug*, MQTT *in*, dan MQTT *out*. *Flow* 2 di konfigurasi dengan cara mengkoneksikan masing-masing *node* dan *double* klik pada *node inject* untuk konfigurasi *node*. Konfigurasi *node inject* dengan tipe data JSON dan mengisi pesan seperti "{"pesan':*node-red* kirim"}".

Kemudian *double click node* MQTT out untuk konfigurasi *node* , lalu konfigurasi *node* dengan cara mengisi *topic* yang di dapatkan dari *console*.telkomiot, yang membedakan node mqtt *in* dan mqtt *out* adalah "pubs" dab "subs". Setelah semua *node* di konfigurasi, maka selanjutnya adalah melakukan *deploy* dengan cara mengklik "deploy" > "confirm deploy", setalah itu maka setiap *node* akan terkoneksi atau "connected" seperti pada Gambar 3.5.

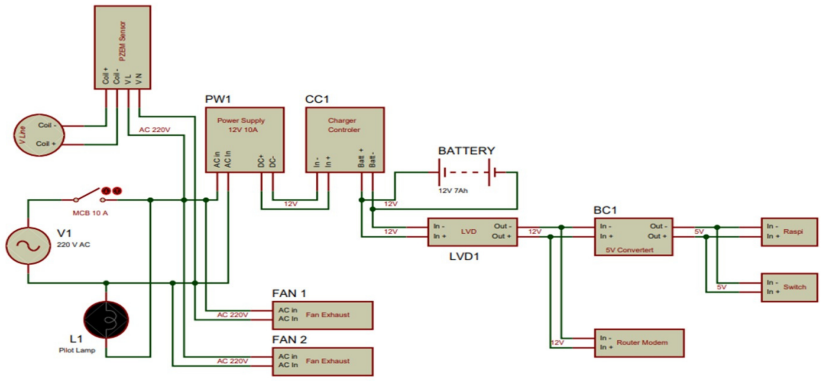
3.5 DIAGRAM BLOK SISTEM

Gambar 3. 6 Diagram Blok Sistem Perangkat *Hydrant Control*

Pada alur penelitian dijelaskan bahwa langkah pertama yang akan dilakukan yaitu mengambil data dengan mengukur kualitas pengiriman data dari perangkat *hydrant* ke Telkom IoT *Platform*. *Hydrant* nantinya akan mengirim data berupa informasi mengenai *Voltage*/Tegangan, *Current*/Arus, dan *Pressure*/Tekanan Air, dan

sudah terkoneksi dengan internet jaringan seluler, sehingga data yang diterima dari *hydrant* ke Telkom IoT *Platform* akan dikirim ke *user web dashboard*.

3.6 SKEMATIK IOT *GATEWAY*



Gambar 3. 7 Skematik IoT Perangkat

Gambar 3.7 menggambarkan skematik IoT Perangkat yang memberikan informasi yaitu nilai *voltage* yang digunakan hingga sensor tegangan untuk mendeteksi arus masuk atau tidak, *battery* sebagai penyimpanan daya, *switch* sebagai penghubung antara *converter* dengan Raspi, *router* modem sebagai jaringan *backbone* dan LTE, dan *fan* yang berfungsi sebagai udara dalam *panel box*.

Tabel 3. 2 Nama komponen dan spesifikasi perangkat

No	Nama Komponen	Spesifikasi Perangkat
1	Input AC	-16A / 220 250V
		- Steker Broco yang dilengkapi dengan arde /
		-ground
2	Sensor PZEM	-Voltage
		- Measuring range : 80~260V
		-Resolution: 0.1V -
		Measurement Accuracy: 0.5%

3	Pilot Lamp	Diameter : 22mm. Voltase : AC220V. Warna : Hijau
4	Power Supply	Brand : FORT Type : S-100-12
		- Single output
		-Output Voltage (Vdc) : 12
		-Output Ampere
		(A) : 8.5
5	Exhaust Fan	Cooling Fan / Exhaust Fan AC 220V 8CM
		- Ukuran : 8cm x 8cm x 2.5cm
		-Input : 220 volt
		-Frekuensi : 50 / 60 Hz
		-Arus : 0.08A
6	MCB	Kapasitas 10 A
		-DOMF01110 = 10A
7	Charger Controller	Voltage: DC 6- 60V (max 80V)
		-Display Precision: 0.1V
		-Control Precision: 0.1V
		-Output Type: direct output Voltage
		Tolerance: +/-
		0.1V
		-Application Fields: 6-60V
		-storage battery
		Size:81*54*18mm
8	Battery 12V 7Ah	TIPE: 12V 7 AH
		-Kapasitas : 7 Ah (Ampere Hour)
		-Tegangan : 12 V (Volt)
		-PxLxT : 15x6x9
		-Cycle Use : 14.40V - 14.80V (25 C)
		-Standby Use: 13.50-13.80V (25C)
		-Intial Current :
		Less Than 2.10 A
9	LVD	Power supply voltage : 12 V- 36 V battery
10	Tombol Starter	rated operational current :
		-3 A at 240 V, AC-

		15, A600
		-0.27 A at 250 V, DC-13, Q600
11	5V Buck Converter	-9-36v 12v to 5V DC 5A
		-Include USB Port
12	Router Modem	-Orbit Star Lite
		- Ukuran modem 135 mm x 112,5
		mm x 30 mm - DL
		/ UL Cat 4 - 2x2
		MIMO LTE
		- 2.4 GHz WiFi
		- 32 pengguna
		- Warna = Putih
13	Raspberri Pi	Raspberry KIT model 4B
		-RAM: 4GB
		- Memory: 16 GB
		-Include cassing dan Fan
14	Switch 5 Port	5-port 10/100M Fast Ethernet

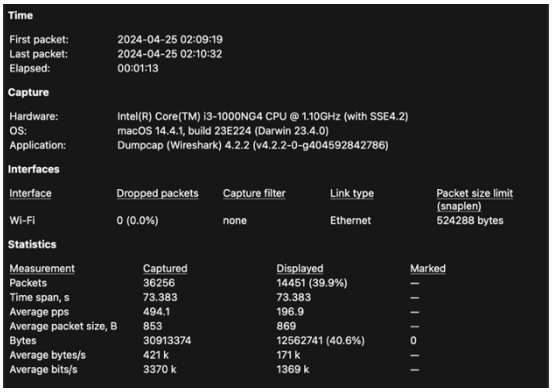
3.7 *QUALITY OF SERVICE*

Quality of Service (QoS) merupakan metode pengukuran tentang seberapa baik jaringan dan merupakan suatu usaha untuk mendefinisikan karakteristik dan sifat dari satu *service*. QoS digunakan untuk mengukur sekumpulan atribut kinerja yang telah dispesifikasikan dan diasosiasikan dengan suatu *service*. Model *monitoring* QoS terdiri dari komponen *monitoring application*, QoS *monitoring, monitor*, dan *monitored object*. *Monitoring application* merupakan sebuah antarmuka bagi administrator jaringan.

Komponen ini berfungsi mengambil informasi lalu lintas paket data dari monitor, menganalisanya dan mengirimkan hasil analisis kepada pengguna. QoS *monitoring* menyediakan mekanisme *monitoring* QoS dengan mengambil informasi nilai-nilai parameter QoS dari lalu lintas paket data. QoS memiliki beberapa parameter yaitu *throughput, packet loss, delay*, dan *jitter* [17].

3.7.1 Throughput

Throughput yaitu kecepatan (*rate*) transfer data efektif, yang diukur dalam bps (bit per *second*). *Throughput* adalah jumlah total kedatangan paket yang sukses yang diamati pada tujuan selama interval waktu tertentu dibagi oleh durasi interval waktu tersebut [18].



Time

First packet:	2024-04-25 02:09:19
Last packet:	2024-04-25 02:10:32
Elapsed:	00:01:13

Capture

Hardware:	Intel(R) Core(TM) i3-1000NG4 CPU @ 1.10GHz (with SSE4.2)
OS:	macOS 14.4.1, build 23E224 (Darwin 23.4.0)
Application:	Dumpcap (Wireshark) 4.2.2 (v4.2.2-0-g404592842786)

Interfaces

Interface	Dropped packets	Capture filter	Link type	Packet size limit (snaplen)
Wi-Fi	0 (0.0%)	none	Ethernet	524288 bytes

Statistics

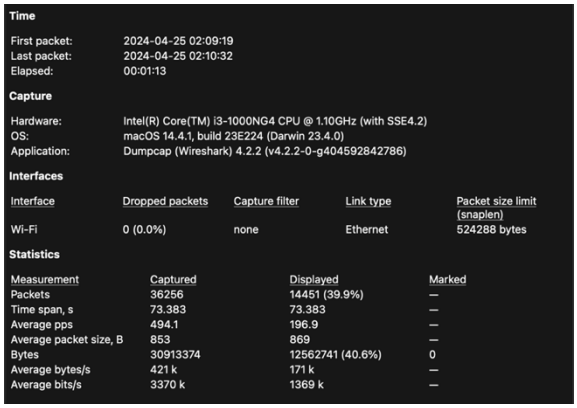
Measurement	Captured	Displayed	Marked
Packets	36256	14451 (39.9%)	—
Time span, s	73.383	73.383	—
Average pps	494.1	196.9	—
Average packet size, B	853	869	—
Bytes	30913374	12562741 (40.6%)	0
Average bytes/s	421 k	171 k	—
Average bits/s	3370 k	1369 k	—

Gambar 3. 8 Tampilan *Capture File Properties* untuk *throughput*

Gambar 3.8 hasil *statistic* dari *capture file properties* pada aplikasi wireshark dengan menampilkan nilai dari *time span* dalam satuan *second* dan nilai *Bytes* yang akan dihitung nilai *averagenya*, sehingga nilai dari *bandwidth internet* dibagi dengan nilai rata-rata dalam satuan bps dan menghasilkan nilai akhir dalam satuan persen yaitu *throughput*.

3.7.2 Packet Loss

Packet Loss merupakan suatu parameter yang menggambarkan suatu kondisi yang menunjukkan jumlah total paket yang hilang dapat terjadi karena *collision* dan *congestion* pada jaringan [18].



Time

First packet:	2024-04-25 02:09:19
Last packet:	2024-04-25 02:10:32
Elapsed:	00:01:13

Capture

Hardware:	Intel(R) Core(TM) i3-1000NG4 CPU @ 1.10GHz (with SSE4.2)
OS:	macOS 14.4.1, build 23E224 (Darwin 23.4.0)
Application:	Dumpcap (Wireshark) 4.2.2 (v4.2.2-0-g404592842786)

Interfaces

Interface	Dropped packets	Capture filter	Link type	Packet size limit (snaplen)
Wi-Fi	0 (0.0%)	none	Ethernet	524288 bytes

Statistics

Measurement	Captured	Displayed	Marked
Packets	36256	14451 (39.9%)	—
Time span, s	73.383	73.383	—
Average pps	494.1	196.9	—
Average packet size, B	853	869	—
Bytes	30913374	12562741 (40.6%)	0
Average bytes/s	421 k	171 k	—
Average bits/s	3370 k	1369 k	—

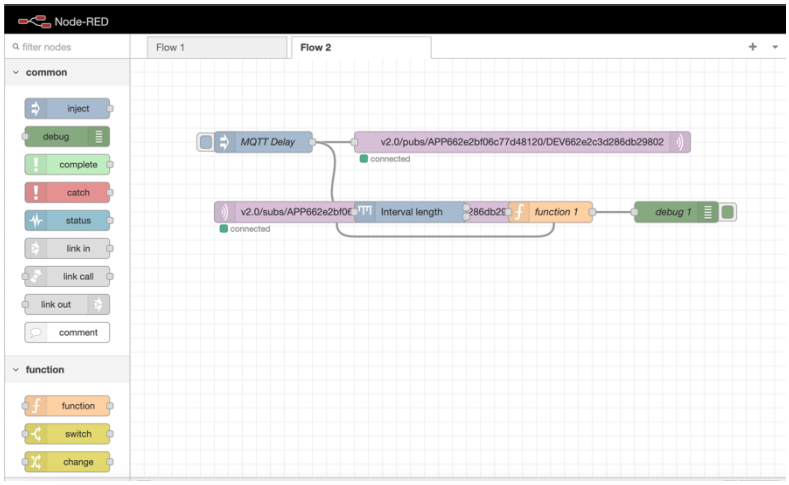
Gambar 3. 9 Tampilan *Capture File Properties* untuk *packet loss*

Gambar 3.9 menampilkan hasil *statistic* dari aplikasi wireshark dengan mengetikkan *tcp.analysis.lost_segment* agar dapat mengetahui *packet loss* dengan menampilkan selisih data yang di *captured* dan data yang di *displayed*.

3.7.3 *Delay*

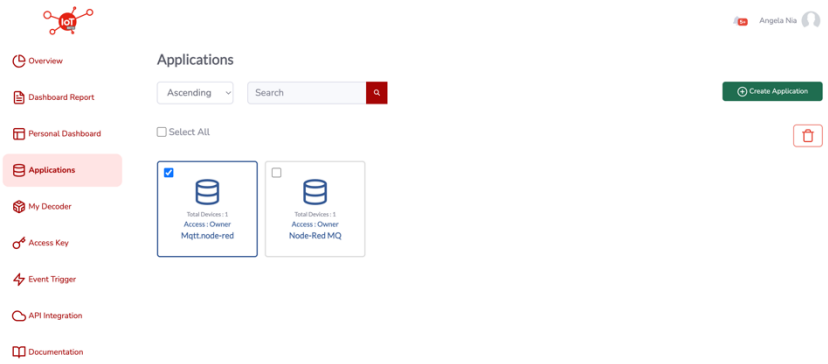
Delay merupakan waktu yang dibutuhkan data untuk menempuh jarak dari asal ke tujuan. *Delay* dapat dipengaruhi oleh jarak, media fisik, *congesti* atau juga waktu proses yang lama [18]. Dalam penelitian ini, untuk mendapatkan nilai *delay* dilakukan dengan menggunakan *node-red*.

Sebelum menggunakan *node-red*, pastikan laptop sudah terinstall Node.js dikarenakan agar *node-red* dapat dijalankan. Sebelum menjalankan *node-red* dan node.js pada laptop, pastikan *node-red* sudah ada pada *local* laptop dengan cara membuka terminal pada laptop yang digunakan seperti pada Gambar 3.5 berikut.

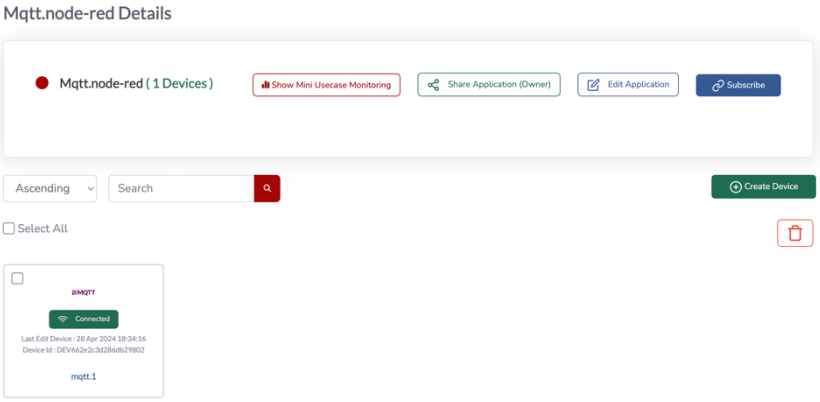


Gambar 3. 10 Tampilan simulasi *node-red hydrant flow* 2

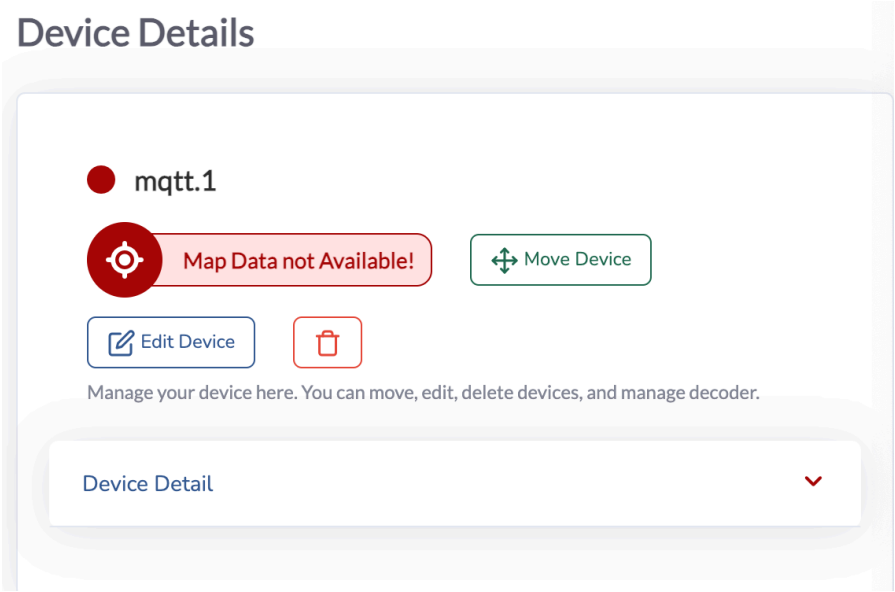
Gambar 3.10 menggambarkan tampilan simulasi pada *node-red* yang mana terdapat 5 *node* dan tiap *node* sudah diberi nama.



Gambar 3. 11 Tampilan *user name profile* dan nama device

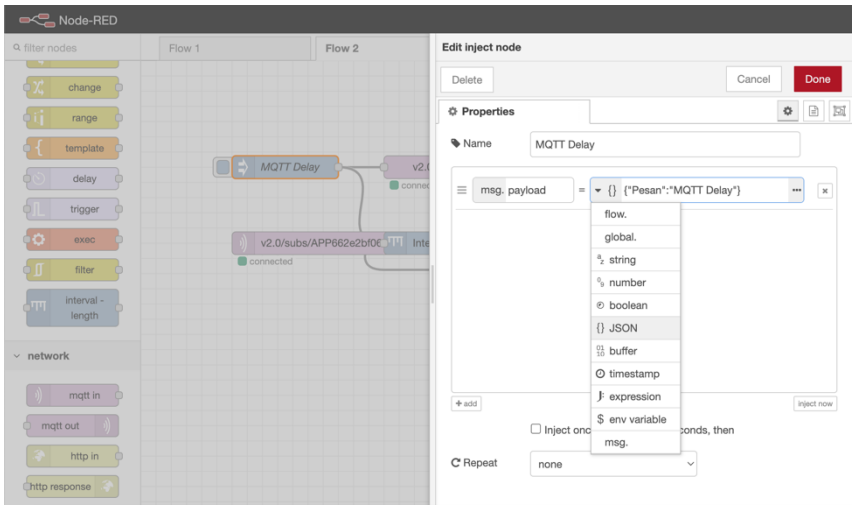


Gambar 3. 12 Tampilan *device* dengan nama yang sudah disesuaikan



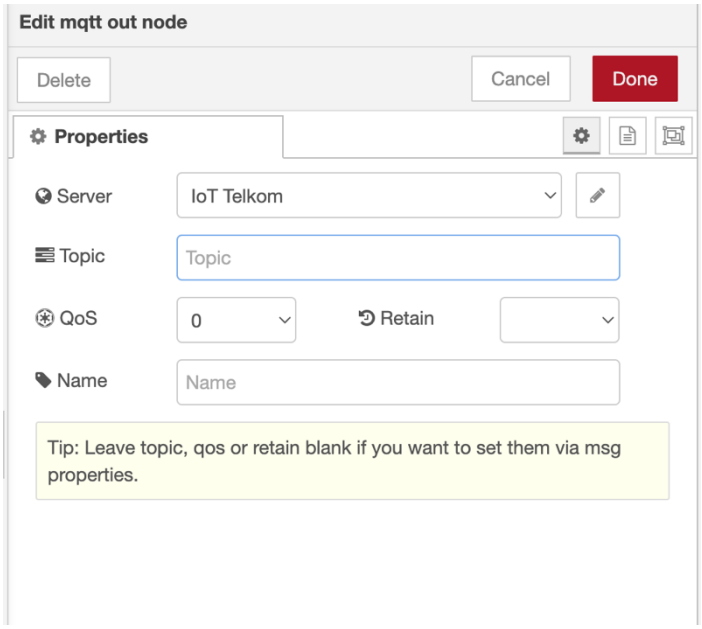
Gambar 3. 13 Tampilan *device details* pada mqtt.1

Gambar 3.13 menampilkan *device* details yang apabila di klik maka akan menampilkan *link* MQTT *topic* dan akan di salin pada *node* mqtt in dan mqtt *out*.



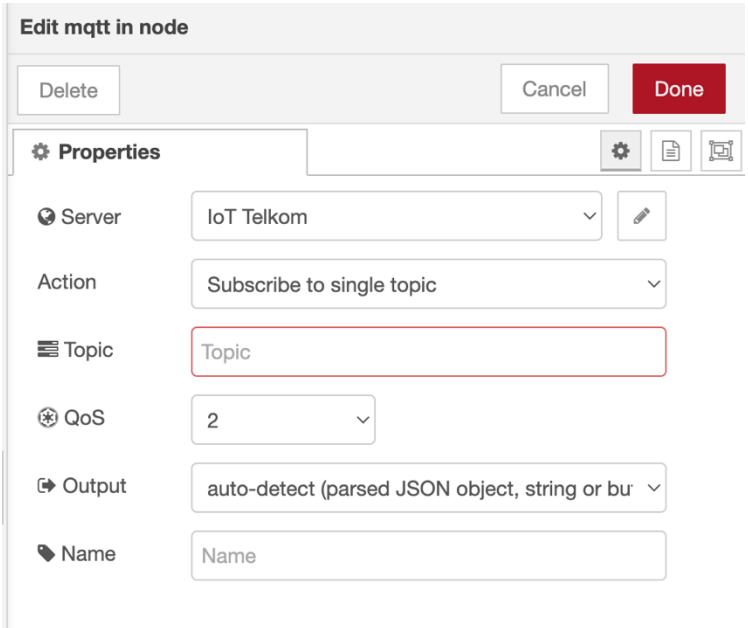
Gambar 3. 14 Tampilan MQTT delay pada *node-red*

Gambar 3.14 menampilkan mqtt *delay* dengan format *file* yang digunakan adalah JSON dan dapat dilihat pada menu *msg.payload.*



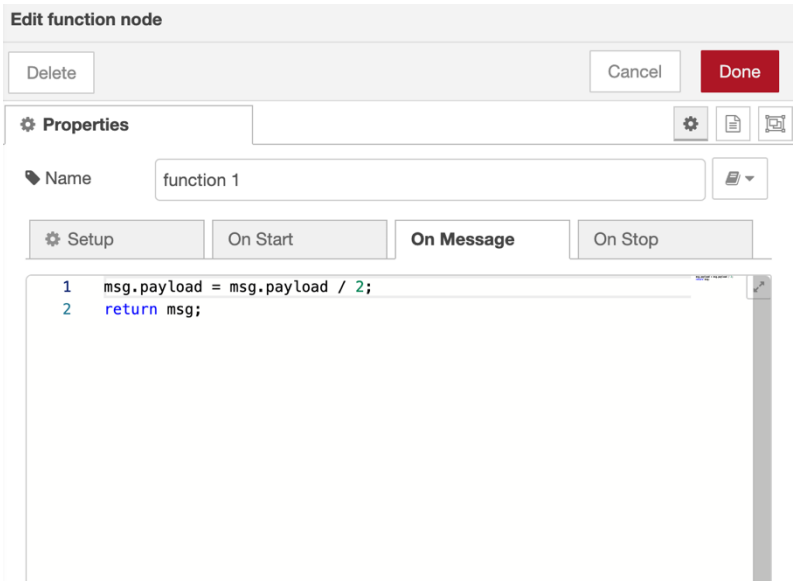
Gambar 3. 15 Tampilan *node* mqtt *out*

Gambar 3.15 menampilkan *node* mqtt *out* dan pada menu *topic* di isi link *pubs* yang didapatkan dari *console* telkomiot yaitu dibagian *device* detail pada mqtt *topic*.



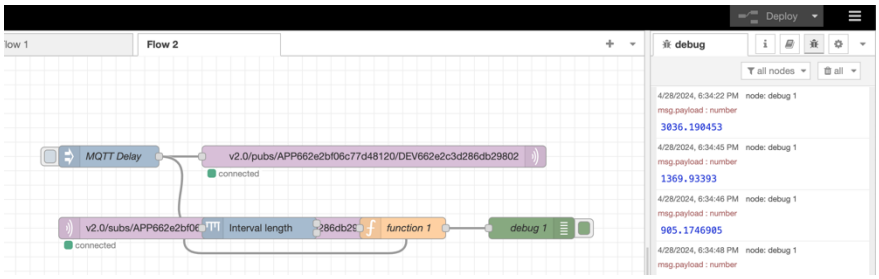
Gambar 3. 16 Tampilan *node* mqtt in

Gambar 3.16 menampilkan *node* mqtt in dan pada menu *topic* di isi *link subs* yang didapatkan dari *console* telkomiot yaitu dibagian *device* detail pada mqtt *topic*.



Gambar 3. 17 Tampilan node function

Gambar 3.17 menampilkan *node function* dengan mengisi msg pada menu *on messag*e yaitu “msg.payload / 2 ;”. Setelah semua node sudah terkonfigurasi dan sesuai, maka selanjutnya adalah *running* dan validasi data.

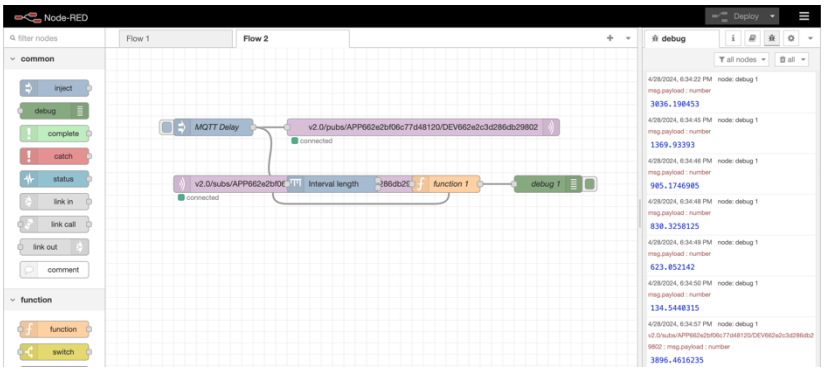


Gambar 3. 18 Tampilan *deploy node* pada *flow* 2

Gambar 3.18 menampilkan hasil *inject node* yang berisi pesan dan sudah di konfigurasi dengan sesuai, sehingga muncul pesan yang di kirim ke IoT *Platform* seperti pada menu *debug messages*.

3.7.4 *Jitter*

Jitter atau variasi Kedatangan packet. *Jitter* diakibatkan oleh variasi-variasi dalam panjang antrian, dalam waktu pengolahan data, dan juga dalam waktu penghimpunan ulang paketpaket diakhir perjalanan *jitter*. *Jitter* umumnya disebut variasi *delay*, berhubungan erat dengan *delay*, yang menunjukkan banyaknya variasi *delay* pada transmisi data di jaringan [18].



Gambar 3. 19 simulasi node-red hydrant pada flow 2

Gambar 3.19 menampilkan hasil simulasi *node-red* dengan melakukan pengujian pada perangkat *hydrant*, di mana *hydrant* disimulasikan pada *node-red* kemudian melakukan *capturing* data pada *software* wireshark dengan interval waktu 3 menit.