IMPLEMENTASI MONITORING DEBIT AIR DAN DETEKSI KEBOCORAN PADA JARINGAN PIPA AIR PDAM BERBASIS INTERNET OF THINGS DENGAN PROTOKOL *MQTT*

SKRIPSI

Disusun oleh:
MUHAMMAD ALDIAN FAIZUN IRSYAD
NIM: 175150209111005

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2021

PENGESAHAN

Implementasi Monitoring Debit Air dan Deteksi Kebocoran Pada Jaringan Pipa Air PDAM Berbasis *Internet of Things* Dengan Protokol *MQTT*

SKRIPSI

Diajukan untuk memenuhi Sebagian persyaratan
Memperoleh gelar Sarjana Komputer

Disusun Oleh :
Muhammad Aldian Faizun Irsyad
NIM: 175150209111005

Skripsi ini telah diuji dan dinyatakan lulus pada
10 Juni 2021
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing

Ir. Heru Nurwarsito, M.Kom.
NIP: 196504021990021001

Mengetahui
Ketua Jurusan Teknik Informatika

Achmad Basuki, S.T., M.MG., Ph.D
NIP: 197411182003121002

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata di dalam naskah skripsi ini dapat dibuktikan tedapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 10 Juni 2021

Muhammad Aldian Faizun Irsyad

NIM: 175150209111005

PRAKATA

Puji syukur atas kehadirat Allah SWT dengan segala limpahan rahmat serta karunia-Nya yang telah memberikan kemudahan dan kelancaran sehingga penelitian skripsi dengan judul Implementasi Monitoring Debit Air dan Deteksi Kebocoran Pada Jaringan Pipa Air Berbasis *Internet of Things* Dengan Protokol *MQTT* dapat terselesaikan dengan baik. Penelitian ini dilakukan dalam rangka memenuhi mata kuliah skripsi untuk memperoleh gelar sarjana komputer di Fakultas Ilmu Komputer Universitas Brawijaya.

Dengan terselesaikannya tugas akhir penelitian ini maka penulis mengucapkan terima kasih kepada pihak yang terkait dalam penyusunan penelitian ini. Pihak-pihak yang telah membantu, mendoakan, membimbing, dan mendukung penulis selama penelitian ini berlangsung hingga penelitian ini dapat terselesaikan khususnya kepada:

1. Allah SWT
2. Kedua orang tua penulis yang secara ikhlas telah memberikan dukungan penuh baik dalam bentuk moril maupun materil.
3. Bapak Wayan Firdaus Mahmudi, S.Si, M.T, Ph.D selaku Dekan Fakultas Ilmu Komputer, Bapak Achmad Basuki, S.T., M.MG., Ph.D selaku ketua Jurusan Teknik Informatika dan Bapak Adhitya Bhawiyuga, S.Kom., M.Sc. selaku Ketua Program Studi Teknik Informatika.
4. Bapak Ir. Heru Nurwarsito, M.Kom. selaku dosen pembimbing skripsi penulis yang telah memberikan arahan, bimbingan serta dukungan hingga terselesaikannya penelitian ini.
5. Seluruh pihak yang telah memberikan dukungan dan bantuan selama penelitian ini berlangsung sehingga dapat terselesaikan dengan baik.

Penulis sadar dalam penyusunan tugas akhir penelitian ini masih banyak kekurangan baik dalam penyajian materi maupun pembahasan. Demi pengembangan penelitian selanjutnya dan kesempurnaan laporan ini, penulis sangat mengharapkan kritik dan saran. Semoga laporan ini dapat memberikan manfaat bagi semua pihak yang membutuhkan.

Sidoarjo, 10 Juni 2021

Muhammad Aldian Faizun Irsyad
aldianfaizun@student.ub.ac.id

ABSTRAK

Muhammad Aldian Faizun Irsyad, Implementasi Monitoring Debit Air dan Deteksi Kebocoran Pada Jaringan Pipa Air PDAM Berbasis *Internet of Things* Dengan Protokol *MQTT*

Pembimbing: Ir. Heru Nurwarsito, M.Kom.

Air merupakan sumber daya yang sangat penting bagi kehidupan manusia untuk kebutuhan sehari-hari. Perusahaan Daerah Air Minum (PDAM) adalah perusahaan untuk memenuhi kebutuhan air masyarakat sekitar dengan cara mendistribusikannya melalui jaringan pipa hingga sampai ke rumah pelanggan. Realitanya distribusi air tidak berjalan dengan lancar dikarenakan terjadinya kebocoran pada jaringan pipa sehingga banyak air yang terbuang dan air pada pelanggan tidak dapat dinikmati secara maksimal karena debit air yang mengecil atau tidak mengeluarkan air. Informasi kebocoran berasal dari laporan masyarakat sekitar atau terlihat pada permukaan tanah dan untuk mengetahui titik kebocoran masih dilakukan secara manual sehingga tidak efektif dan efisien. Pada penelitian ini melakukan implementasi *Internet of Things* untuk monitoring debit air dan mendeteksi jika terjadi kebocoran dengan memanfaatkan protokol *Message Queueing Telemetry Transport* sebagai media komunikasi. Implementasi dilakukan dengan menempatkan perangkat pada jaringan pipa untuk mendapatkan informasi debit air dan koordinat lokasi lalu dikirim menuju *cloud* agar data tersebut dapat ditampilkan melalui halaman web pada sisi *client* untuk kebutuhan monitoring debit air dan mengetahui jika terjadi kebocoran secara *real time*. Dari hasil implementasi memberikan hasil bahwa sistem dapat memberikan informasi debit air dan berubahnya warna ikon pada monitoring menjadi warna merah dan mengirimkan lokasi berupa titik koordinat untuk indikasi terjadinya kebocoran. Kualitas performa sistem memberikan hasil dengan *delay* rata-rata 806 ms dan *packet loss* mendekati nilai 0%..

Kata kunci: *IoT*, *MQTT*.

ABSTRACT

Muhammad Aldian Faizun Irsyad, The Implementation of Water Discharge Monitoring and Leakage Detection in an Internet-of-Things-Based PDAM Water Pipe Network With *MQTT*

Supervisors: Ir. Heru Nurwarsito, M.Kom.

Water is one of the most important resources for the lives of humans and is needed to fulfill their daily needs. PDAM is the company that distributes water to the surrounding people, which is the customers, in their local area through a network of pipes. The fact is that the distribution does not go very well because of the occurrence of leakage on the pipes so that some of the water is coming out of them instead of reaching the destination in an intact manner. This causes a reduction in terms of customers' satisfaction for the water they receive from the PDAM. The notification about the leakage comes manually from the report by the surrounding people or is seen on the soil's surface. The way to detect and trace the leakage point is still done manually so it raises some inefficiency for the work itself. In this paper, the author uses Internet of Things (IoT) to monitor water discharge and to detect any leakage point by implementing the Message Queuing Telemetry Transport (MQTT) as the communication medium. The implementation is done by putting the IoT devices into the pipe network in order to collect the data for water discharge and coordinate and send them to a cloud server to be displayed on a web page that will be accessed on client side for the purpose of real-time pipe leakage monitoring. The result of this implementation shows that the sistem *is able to show the water discharge data and to turn the icon into red when there is a leakage. The* system *is also able to send the coordinate to the leakage history list. The quality of this* system*'s performance shows the average value of 806 ms in terms of delay and close to 0% for packet loss.*

Keywords: *IoT*, *MQTT*

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR ISI

BAB 1 PENDAHULUAN

1.1 LATAR BELAKANG

Dunia teknologi informasi pada era ini semakin maju dan terus berkembang dalam berbagai aspek kehidupan manusia. Konsep *Internet of Things* (*IoT*) merupakan salah satu konsep yang tengah berkembang untuk memudahkan manusia dalam melakukan suatu tugas atau pekerjaan. *Internet of Things* adalah konsep yang memanfaatkan internet sebagai media komunikasi agar dapat terhubung setiap saat (Rochman et al., 2017). Teknologi *IoT* dapat diterapkan untuk proses monitoring secara kontinyu pada sebuah sistem. Sistem monitoring sangat dibutuhkan dalam proses keadaan suatu objek yang diamati untuk mendapatkan informasi dengan waktu nyata atau *real time*. Telah banyak penerapan sistem monitoring yang menggunakan konsep *IoT* seperti pada bidang pertanian, peternakan, dan lain-lain. Terdapat beberapa hal penting yang harus diterapkan sistem monitoring yaitu terutama sumber daya air karena air merupakan kebutuhan primer dalam kehidupan manusia (Arsyistawa et al., 2017).

Air adalah sumber daya alam yang memiliki peran penting dalam kehidupan manusia dan makhluk yang hidup di bumi ini (Dewi et al., 2015). Semakin bertambahnya penduduk maka akan meningkat pula kebutuhan air untuk kebutuhan sehari-hari. Agar kebutuhan manusia terpenuhi maka air perlu diolah dan didistribusikan melalui jaringan pipa. Namun distribusi tidak selamanya lancar karena terdapat kemungkinan terjadi kendala sehingga dapat menurunkan performa proses distribusi air. Terganggunya distribusi aliran air disebabkan oleh beberapa faktor seperti pasokan dari hulu yang sedang menipis atau gangguan pada jaringan pipa air seperti kebocoran (Dharmaadi et al., 2020). Kebocoran terdiri dari 2 yaitu secara teknis dan sesuatu yang tidak diinginkan atau diluar dugaan. Persentase kebocoran secara teknis umumnya hanya 5%, sedangkan kebocoran yang tidak diinginkan memiliki beberapa ragam seperti pencurian air, terdapat sambungan yang tidak rapat, pipa patah atau terkikis karena faktor umur, dan lain-lain(Prasetya et al., 2020).

Kebocoran pada saluran pipa menyebabkan air terbuang sia-sia dan memberikan dampak menurunnya jumlah air yang diterima oleh konsumen. Hingga saat ini untuk mengetahui terjadinya kebocoran yaitu hanya melalui laporan dari pelanggan karena air tidak mengalir atau mengecil serta apabila ditemukan genangan air di permukaan. Untuk mengetahui letak terjadinya kebocoran maka masih dilakukan secara manual yaitu memprediksi dan memperkirakan titik lokasi terjadinya kebocoran dengan cara menggali. Proses deteksi kebocoran yang masih manual ini menyita waktu dan membuang energi karena apabila titik penggalian tidak tepat maka semakin banyak air yang terbuang.

Terdapat beberapa penelitian yang berkaitan dengan sistem deteksi kebocoran yaitu telah dilakukan pada penelitian sebelumnya (Bagita, 2019) membangun sistem deteksi kebocoran yang memanfaatkan *fuzzy logic* dan

pengendalian PID untuk mengontrol tekanan air yang mengalir pada saat mengalami kebocoran. Jika terjadi kebocoran maka pompa air akan dimatikan sehingga mengurangi tingkat kehilangan air yang besar. Namun, pada penelitian tersebut tidak fokus kepada informasi debit air yang mengalir serta lokasi apabila terjadi kebocoran.

Maka dari itu diperlukan sebuah solusi untuk menyelesaikan permasalahan tersebut dengan membangun sebuah sistem monitoring pendeteksi kebocoran pada jaringan pipa air untuk mempermudah proses pencarian titik lokasi terjadinya kebocoran air serta dapat monitoring debit air yang didistribusikan kepada pelanggan menggunakan teknologi *IoT* dengan protokol *MQTT* secara *real time*. Untuk dapat mengetahui lokasi dari titik terjadinya kebocoran diperlukan modul (GPS) yang berkolaborasi dengan sensor *flowmeter* untuk membaca debit air. Apabila debit air yang mengalir mengalami penurunan secara drastis atau tidak pada umumnya maka akan mengirimkan notifikasi ke sebuah web sebagai informasi untuk mengetahui tempat terjadinya kebocoran secara akurat karena terdapat informasi koordinat dari GPS.

Dengan adanya sistem monitoring debit air dan deteksi kebocoran pada jaringan pipa distribusi air akan memperoleh perbaikan dari segi informasi debit air yang mengalir dan titik lokasi terjadinya kebocoran secara akurat. Dengan demikian, sistem ini dapat mempermudah dalam melakukan monitoring secara jarak jauh dan *real time* serta dapat mengetahui titik lokasi kebocorannya.

1.2 IDENTIFIKASI MASALAH

Kebocoran merupakan masalah yang sering dihadapi, namun hingga saat ini untuk mengetahui informasi kebocoran tersebut berasal dari laporan atau menemukan genangan air di permukaan serta untuk menemukan titik kebocorannya dilakukan dengan prediksi sehingga tingkat akurasinya sangat rendah dan hingga sampai saat ini tidak banyak sistem untuk monitoring debit air pada jaringan distribusi air sehingga tidak mengetahui kondisi debit yang mengalir.

1.3 RUMUSAN MASALAH

Berdasarkan penjelasan latar belakang diatas maka dapat dirumuskan sebuah permasalahan dalam studi kasus ini yaitu:

1. Bagaimana mengetahui akurasi dari deteksi kebocoran?
2. Berapa cepat respon sistem untuk mengetahui adanya kebocoran?
3. Bagaimana komputasi untuk deteksi kebocoran di *subscriber*?
4. Bagaimana kinerja *delay* dan *packet loss* pada protokol *MQTT* saat transmisi data?

1.4 TUJUAN

Berdasarkan rumusan masalah yang telah diambil maka berikut adalah tujuan dari penelitian yang akan dilakukan:

1. Mengetahui tingkat akurasi sistem dalam mendeteksi kebocoran pada jaringan pipa.
2. Mengetahui lama waktu respon sistem dalam deteksi kebocoran.
3. Mengetahui kemampuan *subscriber* dalam komputasi deteksi kebocoran pada jaringan pipa.
4. Mengetahui kinerja *delay* dan *packet loss* protokol MQTT saat melakukan transmisi data.

1.5 MANFAAT PENELITIAN

Adapun manfaat yang ingin dicapai oleh penulis dalam penelitian ini yaitu sebagai berikut:

1. Mengetahui jika terjadi kebocoran pada jaringan pipa.
2. Mengetahui responbilitas sistem dalam deteksi kebocoran.
3. Mengetahui efektifitas komputasi sistem dalam mendeteksi kebocoran.
4. Mengetahui peforma dari protokol MQTT pada saat transmisi data.

1.6 BATASAN PENELITIAN

Berdasarkan latar belakang dan rumusan masalah yang telah dikemukakan. Penelitian ini mempunyai batasan-batasan masalah sebagai berikut:

1. Kondisi sumber air dari pompa bersifat statis.
2. Tidak ada varasi pipa untuk simulasi kebocoran dan pipa yang digunakan dengan ukuran 0.5 dim berbahan PVC.
3. Proses komputasi deteksi kebocoran dilakukan oleh *subscriber*.
4. Nilai batas ambang untuk deteksi kebocoran dibandingkan dengan kondisi debit air normal.
5. Proses komputasi deteksi kebocoran dilakukan pada *subscriber* dengan nilai batas ambang antara 28% hingga 30%.

1.7 SISTEMATIKA PEMBAHASAN

Sistematika penulisan penelitian ditunjukkan untuk memberikan gambaran dan uraian dari penyusunan tugas akhir secara garis besar yang meliputi beberapa bab, sebagai berikut

BAB I PENDAHULUAN

Bab ini membahas tentang latar belakang permasalahan, perumusan masalah, tujuan, dan sistematika penulisan.

BAB II LANDASAN KEPUSTAKAAN

Bab ini menguraikan kajian pustaka dan teori-teori yang mendasari teknologi pada teknologi *IoT*

BAB III METODOLOGI

Berisi tentang metode penelitian yang akan digunakan dalam impelmentasi teknologi *IoT*

BAB IV HASIL DAN PEMBAHASAN

Berisi tentang hasil dari implementasi sistem yang telah direalisasikan dan kemudian dilakukan pembahasan serta analisis dengan beberapa parameter yang telah ditentukan.

BAB V PENUTUP

Memuat kesimpulan yang diperoleh dari perancangan dan pengujian impelmentasi teknologi *IoT*, serta saran untuk penyempurnaan penerapan dan perancangan sistem yang telah dilakukan.

BAB 2 LANDASAN KEPUSTAKAAN

Landasan kepustakaan merupakan penjelasan tentang sebuah teori-teori pustaka yang berhubungan langsung dengan penelitian yang akan dilakukan serta memperkuat topik yang telah diambil yaitu Implementasi Monitoring Debit Air dan Deteksi Kebocoran pada Jaringan Pipa Air PDAM berbasis Internet of Things dengan protokol *MQTT*.

2.1 Tinjauan Pustaka

Penelitian yang pertama memiliki latar belakang terkait besarnya nilai tingkat kehilangan air yang disebabkan oleh terjadinya kebocoran pada jaringan distribusi sehingga menyebabkan berkurangnya tekanan dan volume yang mengalir pada sambungan rumah serta membuat besarnya nilai non revenue water yang menyebabkan finansial perusahaan menurun. Dalam penelitian ini dibangun sebuah sistem untuk mengetahui apabila terjadi kebocoran pada jaringan pipa dengan menggunakan sensor *water flow* untuk mengetahui nilai debit air yang mengalir sebagai referensi terjadi bocor atau tidak pada jaringan pipa yang dimana dapat dimonitor menggunakan aplikasi Android. Hasil dari penelitian ini yaitu akurasi dari sensor yang digunakan sangat efektif baik saat kondisi debit air normal dan saat mengalami penurunan (Prasetya et al., 2020). Berdasarkan hal tersebut maka peneliti ingin membuat sistem monitoring debit air dan deteksi kebocoran berbasis *IoT* dengan protokol *MQTT* lalu ditampilkan dalam web sehingga lebih mudah diakses.

Penelitian kedua dilatarbelakangi oleh tingkat kebocoran pada PDAM Way Rilau Bandar Lampung yang begitu besar serta dalam mendeteksi kebocorannya masih menggunakan cara yang manual yaitu dengan cara melihat genangan air pada permukaan saluran pipa atau mendapat laporan dari masyarakat tentang adanya kebocoran atau karena tidak mengalirnya air pada rumah mereka. Dalam penelitian ini membuat solusi dari permasalahan yang dijelaskan sebelumnya dengan membangun sistem mendeteksi kebocoran dengan menggunakan dua buah sensor flowmeter yang ditempatkan sebelum dan sesudah titik kebocoran pipa untuk mendapatkan selisih nilai dari debit air yang masuk dan juga yang keluar sehingga didapatkan hasil yang cepat dan akurat serta memanfaatkan komunikasi berbasis TCP/IP melalui kabel ethernet yang kemudian ditransmisikan menuju PC. Hasil dalam penelitian ini yaitu teknologi *water flow sensor* memiliki hasil yang akurat dan dapat mendeteksi saat terjadi kebocoran melalui selisih debit air (Hariyanto, 2016). Berdasarkan hal tersebut maka peneliti menggunakan jenis sensor flow meter yang sama namun sistem yang dibangun dengan konsep *IoT* sehingga dapat dimonitoring secara jarak jauh dan *real time*.

Penelitian ketiga berangkat dari permasalahan saat proses distribusi air ke konsumen yang terkadang mengalami kebocoran karena seharusnya perusahaan daerah air minum memberikan jaminan bahwa air yang didistribusikan dapat sampai menuju ke pelanggan sesuai dengan kebutuhannya serta bertanggung jawab atas perbaikan jaringan yang rusak. Dalam penelitian ini dibangun sistem

untuk mengetahui posisi kebocoran berdasarkan besar tekanan air yang ada pada jaringan pipa dan mengendalikan debit air pada saat kebocoran dengan menggunakan pengendali PID dengan metode *fuzzy* sehingga pada saat terjadi kebocoran maka kecepatan motor dapat diatur sehingga tekanan air antara pipa yang tidak bocor dengan yang bocor tidak akan sama besar. Hasil dari penelitian yaitu penggunaan metode *fuzzy* dapat mendeteksi kebocoran berdasarkan nilai error yang didapat dari pembacaan sensor (Bagita, 2019). Berdasarkan hal tersebut maka penentuan lokasi kebocoran menjadi referensi dalam penelitian ini namun dengan cara yang berbeda yaitu dengan mengirimkan informasi lokasi berupa titik koordinat menggunakan modul GPS.

Penelitian keempat memiliki latar belakang yang berkaitan dengan pembacaan water meter yang masih konvensional sehingga proses pencatatannya harus mendatangi satu per satu rumah pelanggan yang terpasang water meter yang terkadang mengalami kesalahan mencatat dan bahkan lupa untuk dicatat serta masalah selanjutnya adalah kebocoran atau pencurian air yang terletak pada jaringan pipa pelanggan. Dalam penelitian ini dibangun pengukuran debit air untuk mengetahui jumlah pemakaian air sekaligus untuk mendeteksi jika terjadi kebocoran dengan memanfaatkan sensor *flow meter*, Arduino nano, dan Raspberry sebagai tempat pengolahan data lalu ditransmisikan menuju *cloud* untuk ditampilkan melalui web. Hasil dari penelitian ini yaitu dapat monitoring debit air dan dapat mendeteksi kebocoran dengan melihat selisih nilai dari sensor flow meter lalu menjadikannya sebagai *alert* berupa SMS atau *email* (Gunastuti, 2018). Berdasarkan hal tersebut peneliti ingin membuat sistem deteksi kebocoran dengan cara memberikan batas ambang atau nilai minimum dan diinformasikan melalui web secara visual.

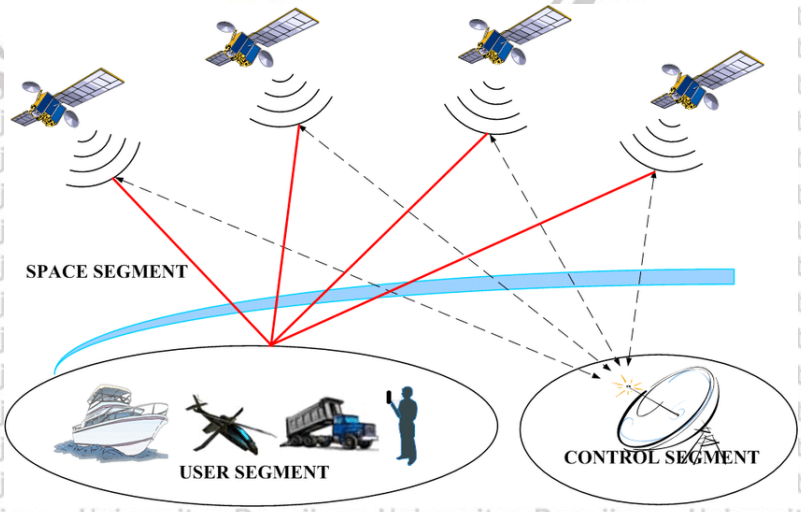
Penelitian kelima dilatarbelakangi oleh sistem pencatatan meter air yang dilakukan oleh PDAM yang masih manual dengan mendatangi langsung setiap rumah pelanggan yang terkadang timbul masalah lainnya seperti kecepatan debit yang tidak diketahui oleh pelanggan maupun petugas, kesalahan pencatatan, dan sulitnya untuk identifikasi kondisi dari meteran air pelanggan. Dalam penelitian ini dibuatlah sebuah sistem untuk mendeteksi debit air menggunakan *water flow* sensor serta didukung oleh Arduino sebagai pengolah data sensor sehingga sistem ini menggunakan konsep *IoT* yang datanya akan ditampilkan melalui aplikasi android. Hasil penelitian ini adalah pembacaan debit air berjalan dengan stabil dan akurat sehingga penggunaan air bisa dimonitoring secara jarak jauh oleh PDAM dan pelanggan (Paksi & Prihartono, 2019). Berdasarkan permasalahan tersebut peneliti ingin membangun sistem debit air beserta deteksi kebocoran menggunakan konsep *IoT* dengan protokol *MQTT* yang dapat dimonitoring melalui web secara jarak jauh dan *real time*.

2.2Air

Air adalah salah satu unsur yang tidak dapat dihindarkan dari kehidupan manusia karena air sudah menjadi kebutuhan pokok. Tidak ada satupun makhluk di dunia ini yang tidak membutuhkan air karena setiap makhluk hidup berasal dari air dan membutuhkannya untuk bertahan hidup (Ardiansyah, 2016). Saat ini taraf kehidupan manusia semakin meningkat, maka meningkat pula kebutuhan air yang diperlukan manusia sehingga air bersih menjadi sulit untuk didapatkan (Waworundeng et al., 2019). Pengembangan sumber daya perlu dilakukan secara konsisten agar kedepannya manusia dapat menikmati air dengan kualitas yang bagus. Manusia membutuhkan air bersih untuk memenuhi kegiatan sehari-hari. Terdapat beberapa syarat air minum yang harus dipenuhi seperti kualitas fisik, kandungan kimia, dan kualitas biologi.

2.3*Global Positioning* Sistem

Global Positioning System adalah sistem navigasi yang berbasis satelit yang saling berkomunikasi satu sama lain sesuai orbitnya. Satelit tersebut milik Departemen Pertahanan Amerika Serikat yang pertama kali memperkenalkan satelit pada tahun 1978 hingga 1944 berjumlah sebanyak 24 satelit. Satelit tersebut mengorbit bumi dengan jarak kurang lebih 20.000 km selama 12 jam untuk memancarkan sinyal. Departemen Pertahanan AS membangun sistem GPS dengan sangat baik sehingga hampir semua wilayah permukaan bumi dapat menerima sinyal sebanyak 4 hingga 10 satelit. GPS mampu menentukan lokasi dengan tingkat ketelitian kurang lebih 1 meter. GPS dapat memberikan informasi berupa letak posisi, kecepatan, dan waktu dengan akurat (Hadi, 2018). Komponen dari GPS terdiri dari tiga bagian antara lain pengguna (*segment*), bumi (*ground segment*), dan luar angkasa (*space segment*). Satelit merupakan bagian dari *space segment* yang berjumlah 24 satelit aktif, 6 orbital planes dengan inklinasi (sudut kemiringan suatu bidang) sebesar 55° dalam kurun waktu 12 jam periode orbital dengan ketinggian 20.000 km serta kecepatan aproksimasi 4 km/detik (Arimbawa et al., 2019).



Gambar 2.1 Arsitektur *Global Positioning System*

(Sumber: Lin, 2014)

Sistem ini menggunakan 24 satelit yang akan mengirimkan sinyal gelombang mikro menuju ke permukaan bumi. Sinyal tersebut diterima oleh alat penerima atau *receiver* yang dimana GPS *receiver* juga mengumpulkan informasi dari satelit GPS dan harus mengunci sinyal setidaknya minim 3 satelit untuk dapat menghitung posisi 2D (*longitude* dan *latitude*) serta arah pergerakan (*tracking*). Jika GPS *receiver* menerima 4 bahkan lebih satelit maka dapat menghitung posisi 3D (*longitude, latitude,* dan *altitude*). Jika GPS *receiver* telah mendapatkan lokasi penerima berada maka dapat menghitung informasi yang lain seperti kecepatan, arah tujuan, jalur, jarak tujuan, dan lain-lain. Sinyal yang dikirim oleh satelit ke GPS *receiver* akan digunakan untuk menghitung waktu perjalanan atau lamanya waktu untuk mencapai tujuan (*travel time*) atau biasa disebut dengan *Time of Arrival* (ToA). Hal ini juga sesuai dengan prinsip fisika yaitu untuk mengukur jarak dapat diperoleh dengan cara waktu dikalikan dengan cepat rambatnya sinyal (Alfeno & Devi, 2017).

2.4*Internet of Things*

Internet of Things adalah sebuah perangkat atau sistem yang terhubung dengan Internet sehingga dapat dimanfaatkan untuk pemantauan dan pengendalian dari jarak jauh (Gunastuti, 2018). Objek yang dapat tersambung dengan perangkat maka akan diketahui perubahan status fisiknya dengan sensor (Sasmoko et al., 2019). *Internet of Things* juga merupakan gambaran komunikasi masa depan dengan menambahkan perangkat seperti mikrokontroler ke kehidupan sehari-hari yang dapat berkomunikasi dengan perangkat lain menggunakan protokol yang telah ditentukan (Mayub et al., 2019). Beberapa penawaran yang diberikan oleh teknologi *IoT* yaitu kemampuan untuk identifikasi sensor, objek, dan menjadi aplikasi yang kooperatif secara independen atau berdiri sendiri. Pada Gambar 2.2 adalah gambaran contoh implementasi node atau objek yang dapat terhubung dengan konsep *IoT*.

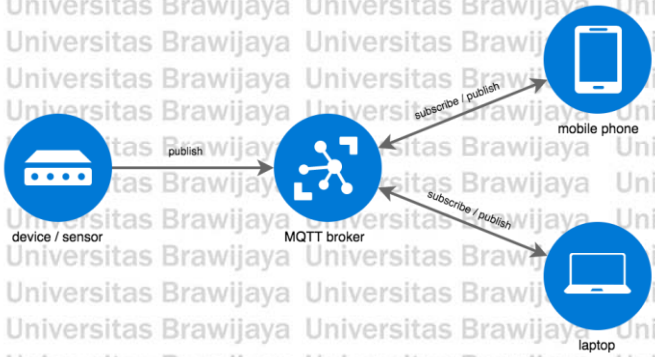
Gambar 2.2 *Internet of Things*

Konsep dari Internet of Things terdiri dari 3 elemen yaitu benda fisik yang terintegrasi dengan modul sensor, internet, dan server sebagai tempat penyimpanan informasi atau data (Setiadi & Abdul Muhaemin, 2018). Beberapa hal yang diperlukan dalam terbentuknya *IoT* yaitu database yang besar, jaringan internet untuk konektivitas, dan kecerdasan buatan yang tertanam yang bertujuan untuk meningkatkan performa dalam memproses suatu data (Lewi et al., 2017). Penggunaan kecerdasan buatan yang tertanam tersebut menjadikan benda cerdas yang dapat membuat keputusan sendiri terhadap respon lingkungan yang diamati sehingga teknologi ini sepenuhnya menjadi responsif dan interaktif.

2.5 *Message Queueing Telemetry Transport*

Message Queueing telemetry transport adalah protokol komunikasi berbasis *publish*/*subscribe*. Karakteristik dari protokol MQTT antara lain rinagn, terbuka, dan simpel sehingga dapat diterapkan dengan mudah. Dengan karakteristik yang dimiliki maka protokol MQTT dapat diimplementasikan dalam berbagai kondisi yaitu pada lingkungan yang memiliki keterbatasan seperti pada komunikasi *Machine to Machine* (M2M) dan IoT yang dimana kebutuhan *bandwidth* sangat diperlukan suatu protokol yang dirancang khusus untuk komunikasi antar perangkat yang berbasis *publish* and *subscribe* (Fauzi, 2019). Protokol *MQTT* merupakan protokol komunikasi yang berjalan pada layer aplikasi dan memiliki sifat *lightweight message* karena *MQTT* pada saat mengirimkan data pesan hanya sebesar 2 *bytes* pada bagian *header*.

Protokol ini mampu memberikan efisiensi untuk mengumpulkan data dalam jumlah yang besar karena ukuran data pesan yang pendek dan prosedur komunikasi yang sederhana (Hwang et al., 2019). *MQTT* sangatlah cocok untuk diterapkan pada lingkungan yang sumber dayanya sangat terbatas seperti rendahnya *bandwidth* dan sumber daya listrik serta protokol ini menjamin terkirimnya pesan walau terjadi putus koneksi (Rochman et al., 2017). Beberapa prinsip yang telah disebutkan maka membuat protokol *MQTT* ideal untuk implementasi pada komunikasi *machine to machine* (M2M) atau *Internet of Things* (Sutiono, 2016). Protokol *MQTT* memiliki jenis protokol data-agnostic yang maksudnya adalah data apapun dapat dikirim seperti binary, text, XML, dan JSON (Putra, 2018). Proses komunikasi *MQTT* dapat dilihat pada Gambar 2.2 yang diawali oleh *publisher* melakukan *publish* data dengan topik yang spesifik menuju *broker*, lalu *subscriber* meminta topik yang diinginkan kepada *broker* dan kemudian *broker* mengirim data kepada *subscriber* sesuai dengan permintaan.



Gambar 2.3 *Message Queueing Telemetry Transport*

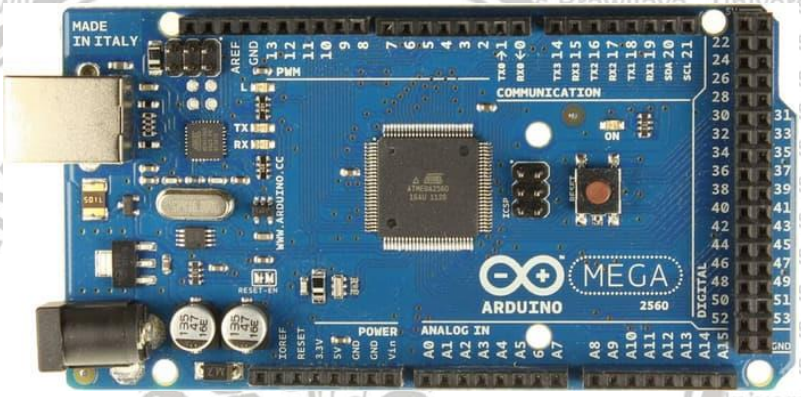
Komunikasi berbasis *publish/subscribe* pada protokol *MQTT* memiliki arti yaitu sebuah pertukaran pesan yang dimana pengirim disebut dengan *publisher* dan penerima disebut *subscriber* serta terdapat *broker* yang bertugas sebagai jembatan komunikasi antara *publisher* dengan *subscriber*. Metode *publish/subscribe* mempunyai kelebihan salah satunya yaitu *loose coupling* atau *decople* yang artinya antara pengirim dan penerima saling tidak mengetahui keberadaannya. Terdapat 3 *decoupling* antara lain *time decoupling*, *synchronization decoupling*, dan *space decoupling*. *Time decoupling* adalah kondisi dimana *publisher* dan *subscriber* tidak harus aktif pada waktu yang bersamaan, *synchronization decoupling* adalah kondisi dimana *publisher/subscriber* untuk mengirim maupun menerima pesan tidak saling mengganggu dan *space decoupling* yaitu kondisi *publisher* dan *subscriber* aktif pada waktu yang sama namun keduanya tidak saling mengetahui keberadaan dan identitas satu sama lain (Rochman et al., 2017).

2.6 Mikrokontroler

Mikrokontroler adalah suatu benda yang mempunyai tugas untuk memproses data atau biasa disebut dengan computer sederhana. Mikrokontroler merupakan *integrated circuit* yang dirancang dengan kepadatan tinggi yang dimana kebutuhan-kebutuhan suatu mikrokontroler telah dibangun dalam bentuk kepingan yang meliputi *Central Processing Unit* (CPU), *Random Access Memory* (RAM), ROM/PROM/EEPROM/EPROM, parallel/serial, *timer* dan *interrupt controller* yang memiliki fungsi untuk mengatur rangkaian elektronik dan dapat dimasukkan suatu program didalamnya (Setiadi & Abdul Muhaemin, 2018). Perangkat ini juga dapat menjalankan perintah yang diberikan atau dibuat oleh programmer. Arduino merupakan salah satu mikrokontroler yang dapat digunakan untuk proyek rintisan maupun akhir atau jangka panjang. Struktur dari Arduino sangat sederhana sehingga membuat pengguna menjadi mudah untuk memahami dalam hal baik visualisasi maupun non visualisasi seperti sensor yang tidak bisa diamati secara visualisasi (Hadi, 2018)

2.7 Arduino Mega 2560

Arduino merupakan sebuah *board* mikrokontroler elektronik yang berbasis *opensource* yang didalamnya terpasang sebuah *chip* dengan jenis AVR ciptaan perusahaan Atmel yang dimana komponen tersebut adalah IC (*integrated circuit*) yang dapat deprogram menggunakan komputer. Pembuatan program pada mikrokontroler tersebut adalah agar dapat membaca *input*, membaca *input*, dan menghasilkan *output* (Andi, 2018). Arduino Mega 2560 adalah *board* mikrokontroler yang menggunakan *chip* ATmega2560 serta memiliki jumlah pin I/O yang banyak yaitu berjumlah 54 buah digital pin yang diantaranya terdiri 15 pin untuk PWM, 16 pin analog, 4 pin UART, 2x3 pin ICSP untuk membuat program Arduino dengan software lain dan kabel USB komputer sebagai sumber tegangan (Junaidi & Prabowo, 2018). Untuk mengaktifkan Arduino Mega 2560 dapat melalui kabel USB tipe B, adapter AC/DC, dan baterai dengan tegangan 7-20VDC. Pada Gambar 2.4 adalah *board* Arduino dengan tipe Arduino Mega 2560.



Gambar 2.4 Arduino Mega 2560

Jika dibandingkan dengan *board* mikrokontroler Arduino yang lain maka terdapat beberapa kelebihan yang dimiliki oleh Arduino Mega 2560 yaitu sebagai berikut:

1. Terdapat *bootloader* yang berfungsi sebagai proses pengendalian jika terdapat *upload* program dari komputer.
2. Terdapat USB sehingga laptop atau komputer yang tidak memiliki *port* RS323 atau serial bisa terhubung dengan *board* Arduino Mega 2560.
3. Memiliki *library* yang lengkap pada *software* IDE Arduino.
4. Beberapa modul telah kompatibel terhadap *board* Arduino Mega 2560.

2.8 Modul SIM5360E

SIM5360E adalah modul yang dapat terhubung dengan perangkat mikrokontroler yang dapat berjalan pada *quad band* (GSM/GPRS/EDGE) dan *dual band* (UMTS/HSDPA/HSUPA/HSPA+) pada frekuensi GSM850, EGSM900, DCS1800, PCS1900, WCDMA2100/900 atau WCDMA1900/ WCDMA850 or WCDMA2100/ WCDMA850. Modul ini memberikan tingkat fleksibilitas yang cukup tinggi dan memiliki beberapa kemampuan aplikasi yang banyak yaitu antara lain TCP/UDP/FTP/FTPS/HTTP/HTTPS/SMTP/POP3 dan MMS sehingga mempermudah

integrasi impelementasi bagi aplikasi pengguna. Detail perangkat modul SIM5360E dapat dilihat pada Gambar 2.5.

Gambar 2.5 Modul 3G SIM5360E

2.9Sensor YF-S201 Flow Meter

Flow meter adalah alat untuk mengukur laju dari aliran air atau jumlah fluida yang mengalir pada suatu pipa yang terbuka maupun tertutup. Jenis fluida yang mengalir yang dapat diukur oleh *flow meter* dapat berupa *liquid*, *solid*, dan gas. Dalam implementasinya, flow meter hanya untuk mengukur aliran saja seperti kecepatan aliran, kapasitas aliran maupun volume. Aplikasi sensor *flow meter* harus sesuai dengan manfaat dan tujuannya karena *flow meter* memiliki banyak istilah berdasarkan fungsinya masing-masing seperti *water meter, flow gauge, liquid meter, flow indicator,* dan lain-lain bergantung pada jenis industri yang berjalan. Namun fungsi utama dari *flow meter* adalah mengukur aliran. Sebagai alat ukur maka variabel yang perlu diperhatikan adalah tingkat akurasinya karena flow meter mempunyai berbagai model dan jenis yang apabila diterapkan memiliki fungsi aplikasi dan tingkat akurasi yang berbeda juga. Semakin tinggi tingkat akurasi data yang didapatkan maka akan semakin bernilai tinggi pula harganya.

Salah satu sensor untuk mengukur *flow meter* air yaitu YF-S201 yang ukurannya kecil dengan 3 kabel *jamper* yaitu data, GND, dan VCC yang dapat dilihat pada Gambar 2.6. Sensor YF-S201 adalah sebuah sensor aliran air yang terbuat dari plastic yang didalamnya terdapat sebuah rotor dan sensor *hall effect*. Rotor akan berputar apabila terdapat aliran yang melewatinya yang menyebabkan terjadinya putaran yang sesuai dengan besarnya aliran air. sensor *hall effect* dapat mendeteksi aliran air sebesar 30 liter/menit sehingga dapat digunakan pada sistem distribusi air, pendinginan berbasis air, dan sistem lainnya yang membutuhkan kontrolisasi debit air. Fungsi utama dari sensor ini yaitu membaca debit dan volume air dengan cara menghitung perputaran kincir didalamnya yang secara otomatis berputar jika terdapat aliran air yang melewatinya(Dewanto & Yoseph, 2018).

Gambar 2.6 Sensor YF-S201

(Sumber: Waworundeng et al., 2019)

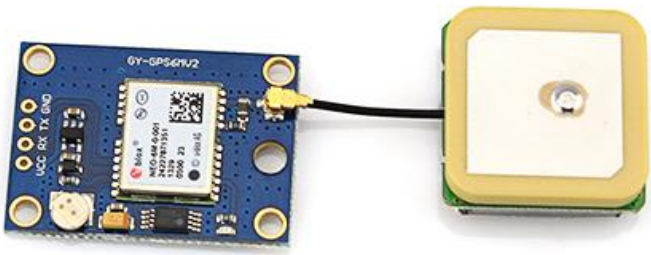
Prinsip kerja pada sensor YF-S201 yaitu dengan menggunakan sensor *hall effect*. Sensor *hall effect* adalah salah satu tranduser untuk mendeteksi medan magnet(Suharjono et al., 2015). *Hall effect* dapat mendeteksi suatu putaran apabila putaran tersebut dipengaruhi oleh medan magnet. Saat air mengalir dan melewati penampang rotor maka kecepatan rotor akan berubah sesuai dengan kecepatan air yang mengalir dan sensor ini memanfaatkan *hall effect* untuk mendeteksi putaran rotor ketika fluida melewatinya(Prasetya et al., 2020). Prinsip kerja sensor *water flow* adalah menghitung putaran kincir air didalamnya yang secara otomatis berputar apabila terdapat aliran yang melewatinya lalu dalam kincir air tersebut terdapat sebuah magnet tetap dan ketika kincir berputar maka akan menghasilkan medan magnet. Kontruksi magnet pada rotor tidak sepenuhnya utuh sehingga *hall effect* akan membaca ada dan tidaknya medan magnet secara berulang kali berdasarkan putaran kincir air dan menghasilkan sinyal pulsa. Sinyal pulsa inilah yang menjadi referensi untuk menentukan berapa besar debit air yang lewat(Gunastuti, 2018).

2.10 Modul GPS Ublox NEO-6MV2

Global Positioning sistem atau biasa disingkat GPS adalah sebuah alat yang berfungsi untuk memberikan informasi tentang posisi atau titik koordinat dari sebuah lokasi di permukaan bumi berbasis satelit (Syaefudin, 2018). GPS memiliki nama lengkap NAVSTAR-GPS yang berhasil dikembangkan oleh departemen pertahanan dan dikelola oleh Angkatan udara milik Amerika Serikat. Meskipun GPS berada dalam pengelolaan milik militer Amerika Serikat, GPS dapat dimanfaatkan oleh rakyat sipil di seluruh dunia tanpa harus mengeluarkan biaya untuk berlangganan dan pemeliharaan sistem GPS (Syaefudin, 2018). Pada dasarnya GPS terdiri dari tiga segmen antara lain segmen angkasa, segmen sistem kontrol, dan segmen pemakai. Terdapat beberapa sistem yang serupa seperti GPS yaitu antara lain IRNSS India, Galileo Uni Eropa, dan GLONASS Rusia(Muhammad, 2017).

Sistem ini berjalan secara terus menerus dan menggunakan 24 satelit yang memiliki tugas untuk mengirim sinyal gelombang mikro ke Bumi. Sinyal akan diterima oleh alat penerima di permukaan bumi untuk menentukan arah, letak,

kecepatan, dan waktu. GPS *receiver* harus mengunci setidaknya minimal tiga satelit untuk dapat menghitung posisi 2D (*latitude* dan *longitude*) serta *tracking position*(Alfeno & Devi, 2017). Apabilla GPS *receiver* menerima sinyal dari empat atau lebih dari satelit maka dapat menentukan posisi 3D yaitu *longitude, latitude,* dan *altitude*. Jika posisi *user* telah ditemukan maka Langkah selanjutnya GPS menghitung informasi lainnya seperti arah yang dituju, kecepatan, jalur, rute perjalanan, jarak tujuan, matahari terbit serta terbenam. Sinyal yang dikirim dari satelit menuju GPS *receiver* berguna untuk menghitung waktu perjalanan yang biasa disebut dengan *Time of Arrival* (TOA). Menurut teori fisika, bahwa mengukur jarak dapat dilakukan dengan menghitung waktu dikali dengan cepat rambat sinyal.



Gambar 2.7 Modul GPS Ublox NEO-6MV2

(Sumber: http://www.labelektronika.com)

Pada Gambar 2.7 adalah gambaran dari modul GPS U-Blox NEO-6MV2 yang bergunsi untuk memberikan informasi berupa lokasi dengan cara menangkap dan memproses sinyal dari satelit(Hermanto et al., 2016). Jenis GPS ini dapat diandalkan karena performanya akurasi informasi yang cukup baik dan beberapa keuntungan lainnya yaitu terdapat baterai cadangan, *built-in* kompas elektronik dan antena keramik untuk meningkatkan saat menangkap sinyal(Muhammad, 2017). Modul ini memiliki tingkat akurasi sekitar 2.5 meter hingga 10 meter(Arimbawa et al., 2019). Arsitektur pada modul ini sangat ideal untuk diterapkan pada perangkat *mobile* yang beroperasi dengan keterbatasan energi dan ruang yang terbatas.

2.11 *Quality of Services*

Quality of Services atau biasa disebut QoS yang sangat popular serta memiliki arti yang berbeda dari berbagai perspektif yaitu dari segi jaringan atau *networking* dan pengembangan aplikasi atau *application development* (Ahmady, 2018). Pada sisi *networking* maka memiliki arti yaitu kemampuan dalam hal memberikan pelayanan atau jaminan kinerja terhadap lalu lintas jaringan (Sugeng et al., 2015). QoS dalam *network* maka fokus kepada tingkat kecepatan dan kehandalan untuk menyampaikan berbagai jenis beban data pada suatu jaringan (Adani, 2016). Kegunaan dari QoS adalah untuk mengukur kinerja dari atribut yang telah dispesifikkan dan diasosiasikan pada suatu layanan atau *service*. Terdapat

beberapa parameter yang digunakan sebagai referensi untuk dapat mengukur kinerja dari suatu jaringan yaitu antara lain *delay* dan *packet loss*.

2.11.1 Delay

Delay adalah waktu yang dibutuhkan oleh suatu paket data yang dikirim oleh sumber kepada tujuan yang diakibatkan oleh antrian atau mengambil rute alternatif agar terhindar dari kemacetan tersebut. Terdapat beberapa pengaruh yang menyebabkan terjadinya *delay* yaitu jarak, media fisik, kongesti, dan lama waktu untuk memprosesnya(Yahdiani, 2020). Rumus perhitungan nilai *delay* dapat dilihat pada Rumus 2.1 dibawah ini.

$$Delay = \frac{total\ delay}{total\ paket\ yang\ diterima} \quad (2.1)$$

Terdapat beberapa kategori performa kualitas *delay* menurut standarisasi TIPHON yaitu pada Tabel 2.1 dibawah ini:

Tabel 2.1 Standarisasi *Delay*

Kategori	*Delay*
Sangat Bagus	< 150 ms
Bagus	150 – 300 ms
Sedang	300 – 450 ms
Jelek	> 450 ms

2.11.2 Packet Loss

Packet loss adalah definisi dari gagalnya atau hilangnya paket pada saat transmisi untuk mencapai tujuannya(Rachman & Iskandar, 2017). Beberapa faktor yang dapat menyebabkan terjadinya *packet loss* antara lain menurunnya sinyal pada media jaringan, melebihi batas saturasi, dan kesalahan *hardware* atau hal-hal yang sifatnya teknis seperti *collision* dan *congestion* pada jaringan.

$$Packet\ Loss = \frac{(paket\ dikirim - paket\ diterima)}{paket\ dikirim}\ x100\% \quad (2.2)$$

Terdapat beberapa kategori performa kualitas *packet loss* menurut standarisasi TIPHON yaitu pada Tabel dibawah ini:

Tabel 2.2 Standarisasi *Packet Loss*

Kategori	*Packet Loss*
Sangat Bagus	0 %
Bagus	3 %
Sedang	15 %
Jelek	25 %

BAB 3 METODOLOGI PENELITIAN

3.1Krangka Penelitian

Krangka penelitian yaitu membahas tentang alur dari penelitian yang akan dilakukan sehingga dapat terarah dengan baik dan sesuai dengan tujuan penelitian. Tahapan penelitian berupa diagram alir yang dapat dilihat pada Gambar 3.1.

Gambar 3.1 Diagram Metode Penelitian

Pada Gambar 3.1 yaitu menjelaskan tentang bagaimana alur penelitian yang diawali dengan pembuatan kerangka penelitian, kemudian merancang sistem, pemilihan metode untuk evaluasi serta melakukan implementasi sistem yang akan dibuat. Kerangka penelitian terdiri dari beberapa hal yaitu antara lain studi literatur sebagai dasar untuk memahami sistem yang akan dibuat dan rekayasa kebutuhan yang berisi tentang apa saja yang diperlukan dalam membuat sebuah sistem. Perancangan sistem adalah uraian tentang bagaimana sistem yang digunakan dan bagaimana sistem dapat bekerja. Metode evaluasi menjelaskan

tentang rancangan untuk melakukan proses pengujian sistem yang dibangun serta melihat performa sistem tersebut. Implementasi berisi tentang realisasi dari perancangan sistem yang telah dirancang sebelumnya.

3.1.1 Studi Literatur

Studi literatur adalah sebagai bahan penunjang dari penelitian ini agar memiliki landasan pengetahuan yang cukup untuk membangun sistem yang akan dibuat oleh penulis. Studi literatur dilakukan dengan cara mengumpulkan beberapa teori dan Pustaka yang berkaitan dengan penelitian ini yaitu terdiri dari:

1. Air

   Air adalah sebuah elemen primer yang dibutuhkan oleh manusia sehingga air tidak dapat dipisahkan dalam kehidupan manusia karena pada dasarnya setiap makhluk hidup memerlukan air untuk bertahan hidup.

2. *Internet of Things*

   Internet of Things adalah perangkat yang terhubung dengan Internet sehingga dapat memantau sebuah objek secara jarak jauh dan *real time*.

3. Arduino

   Arduino adalah rangkaian atau papan elektronik yang berbasis *opensource* yang didalamnya terdapat *chip* mikrokontroler.

4. *Message Queue Telemetry Transport*

   Message Queue Telemetry Transport adalah sebuah protokol komunikasi yang berbasis publish dan subscribe serta bersifat *lightweight.*

3.1.2 Rekayasa Kebutuhan

Kebutuhan sistem adalah suatu kebutuhan yang harus dipenuhi agar sistem dapat berjalan sesuai dengan perancangan dan tujuan awal. Tahap analisis rekayasa kebutuhan sistem sangat dibutuhkan untuk mendefinisikan secara detail tentang apa saja yang dapat dilakukan oleh sistem dan mengetahui apakah sistem yang dibuat telah sesuai pada rancangan sebelumnya. Sistem yang dibangun berdasarkan rumusan masalah sehingga ketika sistem telah berjalan sesuai dengan kebutuhan maka sistem tersebut telah berjalan dengan baik. Rekayasa kebutuhan terdiri dari kebutuhan fungsional dan kebutuhan non fungsional.

3.1.2.1 Rekayasa Kebutuhan Fungsional

Rekayasa kebutuhan fungsional adalah tentang kebutuhan yang harus dapat dilakukan oleh sistem yang akan dibuat pada penelitian ini. Kebutuhan fungsional sangat bergantung pada sistem yang dibuat, tujuan sistem dan perlakuan *user* yang akan menggunakannya. Berikut adalah beberapa daftar dari kebutuhan fungsional pada sistem monitoring debit air dan sistem deteksi kebocoran pada penelitian ini:

1. Sistem mampu mengetahui debit air yang mengalir pada jaringan pipa melalui sensor *flow meter*.

2. Sistem mampu memberikan informasi lokasi berupa titik koordinat melalui modul GPS.
3. Sistem dapat mengirimkan data sensor ke *broker* menggunakan protokol *MQTT* dengan topik yang telah ditentukan sebelumnya.
4. Sistem mampu menampilkan data sesuai topik yang tersedia pada *broker*.
5. Sistem mampu melakukan monitoring debit air dan memberikan informasi kebocoran secara visualisasi pada halaman web.
6. Sistem mampu menampilkan data histori tentang kebocoran yang terjadi sebelumnya.

3.1.2.2 Rekayasa Kebutuhan Non Fungsional

Kebutuhan non fungsional adalah tentang kebutuhan yang tidak ada hubungannya dengan kegiatan atau hal yang dapat dilakukan oleh suatu sistem akan tetapi berfokus pada pembatasan fungsi implementasi sebuah sistem, sehingga kebutuhan non fungsional akan memberikan batasan ruang lingkup penerapan sistem yang akan dibuat pada penelitian ini. Kebutuhan non fungsional terdiri dari dua bagian yaitu antara lain kebutuhan perangkat keras dan kebutuhan perangkat lunak.

A. Kebutuhan Perangkat Lunak

Berikut adalah beberapa kebutuhan perangkat lunak yang berfungsi untuk memberikan batasan implementasi:

1. Arduino IDE

   Perangkat lunak untuk menulis *source code* dengan menggunakan Bahasa pemrograman C dan memasukkannya ke mikrokontroler

2. Sistem operasi Windows 10

   Sistem operasi pada penelitian ini menggunakan windows 10 sehingga implementasi dan perangkat lunak berjalan dan diatur oleh sistem operasi.

B. Kebutuhan Perangkat Keras

Berikut adalah beberapa kebutuhan perangkat keras yang digunakan pada penelitian ini:

1. Laptop

   Laptop berfungsi untuk menjalankan sistem operasi serta perangkat lunak pada saat proses implementasi dengan spesifikasi RAM 8GB dan prosesor intel i5.

2. Arduino Mega 2560

   Perangkat mikrokontroler yang digunakan untuk mengumpulkan dan memproses data hasil *sensing* dari sensor yang nanti data tersebut akan diteruskan menuju *cloud*.

3. ESP8266

   Adalah perangkat mikrokontroler yang fungsinya hampir sama dengan Arduino mega namun hanya sebagai pendukung pada saat proses implementasi.

4. YF-S201

   Adalah sebuah sensor untuk menghitung debit air yang mengalir pada jaringan pipa dan data tersebut diteruskan ke mikrokontroler.

5. U-Blox NEO-6M

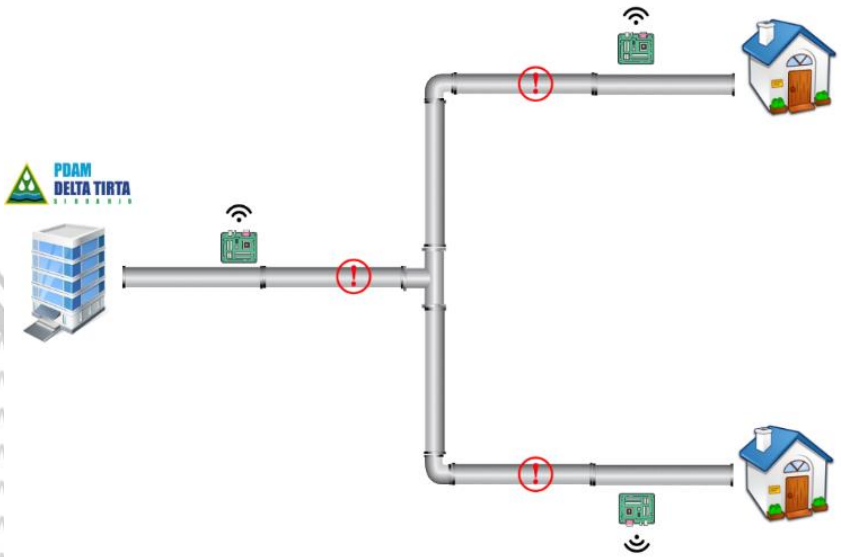
   Adalah perangkat modul GPS yang akan memberikan informasi lokasi berupa titik koordinat yaitu *longitude* dan *latitude* dan data tersebut diteruskan ke mikrokontroler.

6. SIM5360E

   Adalah sebuah modul 3G yang berfungsi sebagai sumber koneksi internet untuk mikrokontroler Arduino mega. Kartu perdana yang digunakan yaitu Telkomsel.

3.2 Perancangan Sistem

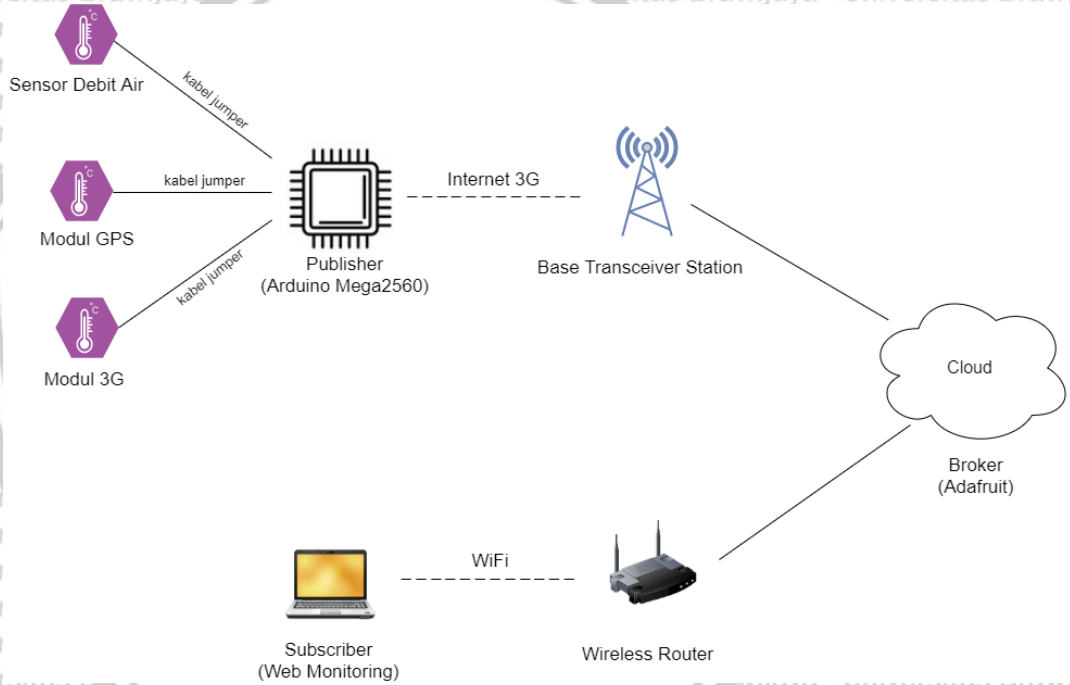
Pada tahap sebelumnya telah menjelaskan tentang kebutuhan apa saja yang diperlukan maka Langkah selanjutnya yaitu melakukan perancangan untuk mempermudah pada saat proses implementasi dan tahapan berikutnya. Perancangan sistem pada penelitian ini dapat dilihat pada Gambar 3.2.



Gambar 3.2 Simulasi Jaringan Pipa Air

Untuk perancangan ini dibuat secara simulasi dengan cara membangun jaringan pipa distribusi air menggunakan pipa pvc ukuran ½. Gambaran topologi jaringan pipa distribusi air seperti pada Gambar 3.2 Rancangan untuk sistem monitoring debit air yaitu dengan cara memberikan aliran air dengan debit yang

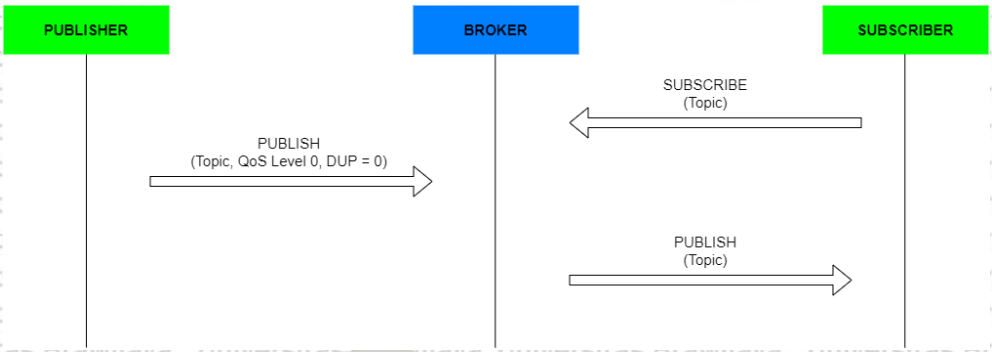
normal atau statis pada jaringan pipa lalu node sensor membaca dan mengirimkannya kepada *broker*. Untuk deteksi kebocoran maka dibuatkanlah titik kebocoran yang berjumlah 3 titik yang tertera pada Gambar 3.2 dengan ikon tanda seru (!) berwarna merah yang nantinya titik tersebut dibuka melalui kran air dengan posisi tuas terbuka lebar. Untuk mengetahui kondisi bocor atau tidaknya pada jaringan pipa maka perlu dikomputasi pada sisi *subscriber*. Pada *subscriber* telah ditentukan nilai batas ambangnya yaitu sebesar kurang lebih 30% dari kondisi normal debit air yang mengalir. Nilai batas ambang tersebut menjadi perbandingan atas debit yang mengalir. Jika nilai debit air pada pipa telah menyentuh atau dibawah dari nilai batas ambang maka dinyatakan terjadi kebocoran, jika tidak maka kondisi dikatakan normal. Ketika terjadi kebocoran maka akan memberikan notifikasi berupa merubah warna pada ikon yang awalnya hijau (normal) menjadi merah serta dimasukkan kedalam tabel histori yang berisi tanggal dan lokasi titik kebocorannya pada halaman web client.



Gambar 3.3 Gambaran Umum Sistem

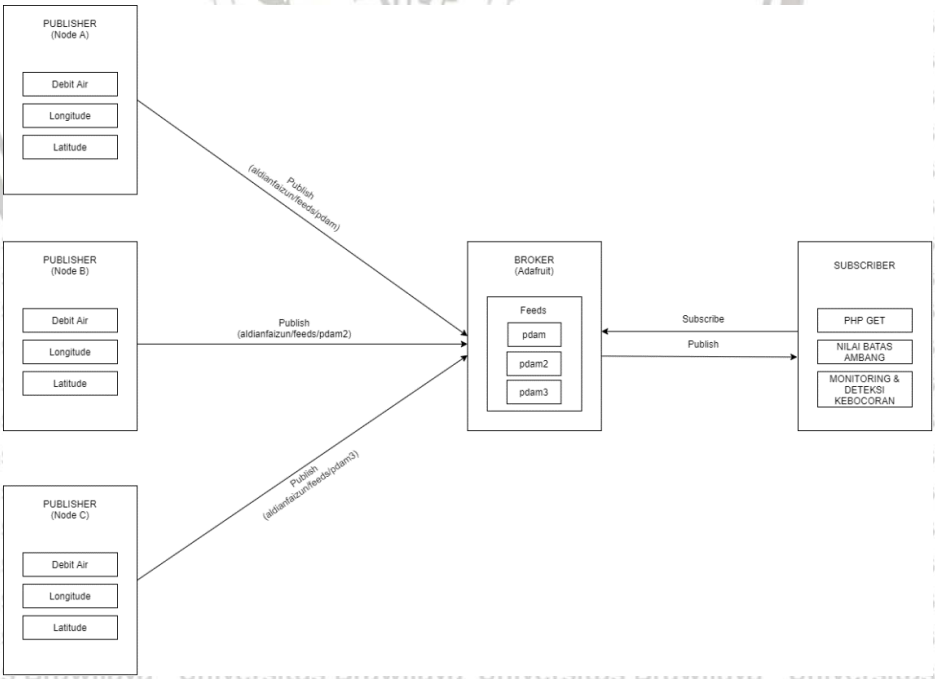
Sistem dibangun dengan menggunakan beberapa perangkat yaitu mikrokontroler, sensor dan modul. Mikrokontroler yang digunakan adalah Arduino mega 2560 yang berperan mengolah data dari hasil sensing yang berhasil dilakukan oleh sensor *flow meter* untuk menghitung debit air yang mengalir dan GPS untuk memberikan informasi berupa titik koordinat. Data yang diproses oleh Arduino maka akan di *publish* ke *broker* Adafruit menggunakan koneksi internet 3G dari kartu perdana Telkomsel melalui protokol komunikasi *MQTT* dengan topik yang telah ditentukan sebelumnya dan berbeda-beda. *Broker* hanya bertugas sebagai jembatan antara *publisher* dengan *subscriber* dan menyimpan data dari *publisher* berdasarkan topiknya masing-masing serta mengirimkan data kepada *subscriber* apabila terdapat *request subscribe* sesuai dengan topik yang diminta dan tersedia pada *broker*. *Subscriber* pada penelitian ini yaitu menggunakan web server yang

akan mengambil atau *subscribe* data sesuai topik yang diinginkan kepada *broker* lalu menampilkannya pada halaman web untuk kebutuhan monitoring secara *real time*.



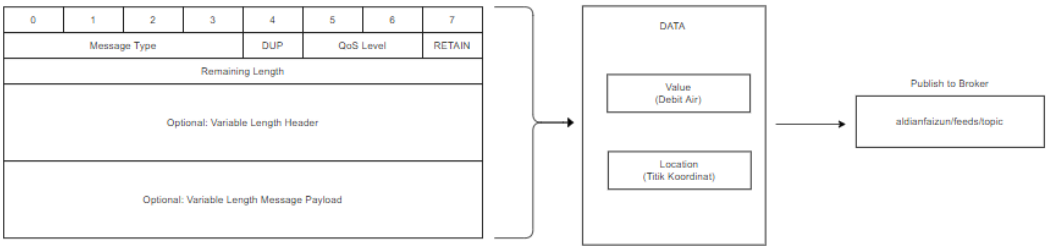
Gambar 3.4 MQTT QoS Level 0

Sistem yang dibangun pada penelitian ini menggunakan konsep *IoT* dengan protokol MQTT. Pada MQTT terdapat 3 jenis level QoS yaitu level 0 (*at most once*), level 1 (*at least once*), dan level 2 (*exactly once*). Level QoS adalah fitur untuk memberikan tingkat garansi transmisi data untuk sampai ke tujuan. Pada penelitian ini menggunakan QoS level 0 atau biasa disebut degan *fire and forget* karena data yang digunakan pada sistem ini membutuhkan data yang cepat tersampaikan pada tujuan tidak seperti level 1 dan level 2 yang membutuhkan proses *handshaking* cukup lama namun garansi data terjamin. Rancangan komunikasi pada sistem ini terdiri dari 3 bagian yaitu *publisher*, *broker*, dan *subscriber*.



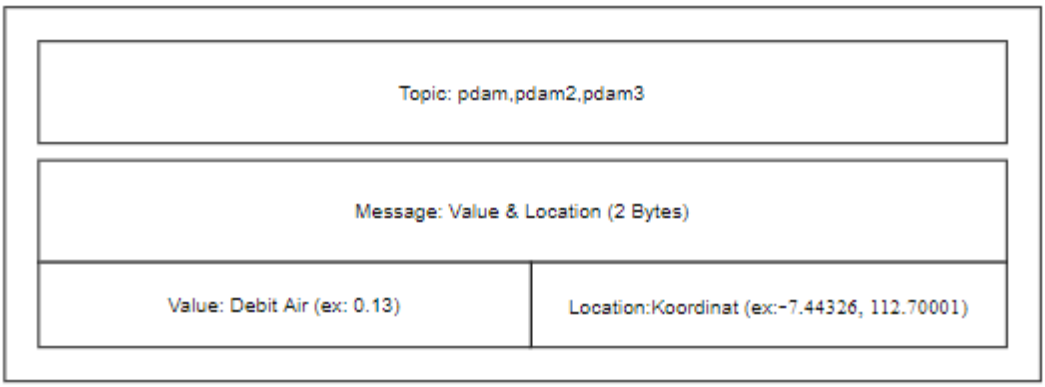
Gambar 3.5 Alur Komunikasi Sistem

Publisher berjumlah sebanyak 3 *node* yang bertugas untuk melakukan *publish* data yang berisikan pesan nilai debit air, *longitude*, dan *latitude* yang dirangkum menjadi kesatuan data dengan topik masing-masing yaitu pdam, pdam2, dan pdam3. Setiap *publisher* memiliki topik tersendiri seperti node A dengan topik pesan pdam, node B dengan topik pesan pdam2, dan node C dengan topik pesan pdam3. Topik disini semacam *labeling* suatu data agar dapat disimpan dan diarsipkan oleh *broker* agar data tidak tercampur dan mudah untuk proses administrasi/*filtering* ketika *subscriber* melakukan *request* topik. *Publisher* dalam waktu tertentu akan melakukan *publish* data berdasarkan topiknya menuju *broker*.



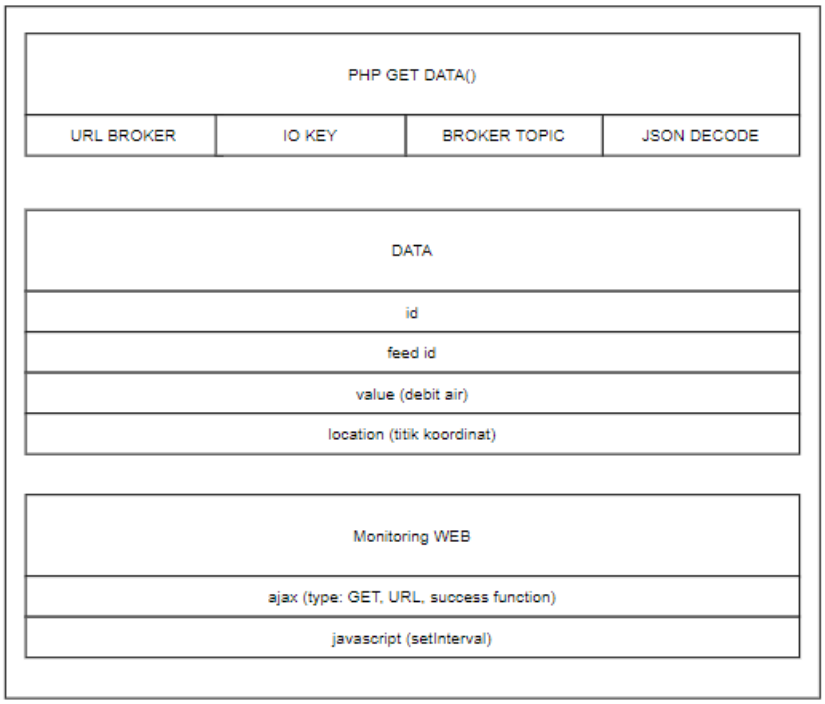
Gambar 3.6 Struktur Perancangan Data Publisher

Struktur perancagan data pada *publisher* dapat dilihat pada Gambar 3.6 yang dimana pada sebuah pesan *publish* maksimal kurang lebih sebesar 2 *Byte*. Byte pertama berisikan *message type* seperti CONNECT ,CONNACK , PUBLISH, dan lain-lain. Selanjutnya DUP adalah sebuah *flag* atau tanda untuk paket apakah mengirim pesan yang sama atau bukan, jika pesan tidak mengalami duplikasi maka *set* nilainya 0 sedangkan duplikasi *set* nilai 1. Bit 5 dan 6 berisi jenis QoS Level yang digunakan yaitu mulai dari level 0, level 1, dan level 2. Bit yang terakhir yaitu RETAIN yang secara *default set true* maka *broker* akan menyimpan 1 pesan terakhir pada masing-masing topik dan mengirimkan pesan terakhir pada topik it uke siapa saja yang melakukan *subscribe*, dengan hal ini akan memudahkan *subscriber* untuk mengetahui pesan terakhir yang ada pada topik tersebut. Setiap bit tersebut dibentuk menjadi satu kesatuan berupa pesan yang akan di *publish* menuju *broker* dengan format *username/feeds/topic*.



Gambar 3.7 Struktur Perancangan Data Broker

Tugas dari *broker* adalah untuk mengolah, filterisasi, dan sebagai jembatan komunikasi antara *publisher* dengan *subscriber*. *Broker* yang digunakan yaitu Adafruit berbasis *cloud service* dan juga menyediakan IO *username* dan IO *key* untuk komunikasi data baik itu *publisher* maupun *subscriber*. Pada *broker* juga menyediakan suatu wadah sebanyak 3 *feeds* untuk menyimpan data yang nanti dikirim oleh *publisher*. Setiap topik yang dikirim oleh *publisher* akan disimpan oleh *broker* menggunakan nama *feeds* yang sama dengan nama topik pesan *publish* oleh *publisher*. Struktur perancangan data pada *broker* dapat dilihat pada Gambar 3.7 yang berisikan beberapa struktur yaitu *topic* dan *message*. *Topic* adalah wadah untuk menampung pesan-pesan yang dikirm dari *publisher*. Topik yang digunakan pada penelitan ini yaitu pdam, pdam2, dan pdam3. Setiap topik berisikan sebuah pesan yang isinya *value* dan *location*. *Value* disini adalah debit air yang mengalir dalam bentuk *decimal*, sedangkan *location* adalah titik lokasi berupa titik koordinat (*longitude* dan *latitude*). Data pada *broker* akan terus disimpan sambil menunggu *request* dari *subscriber* yang kemudian dicocokkan lalu di *publish* menuju *subscriber*.



Gambar 3.8 Struktur Perancangan Data Subscriber

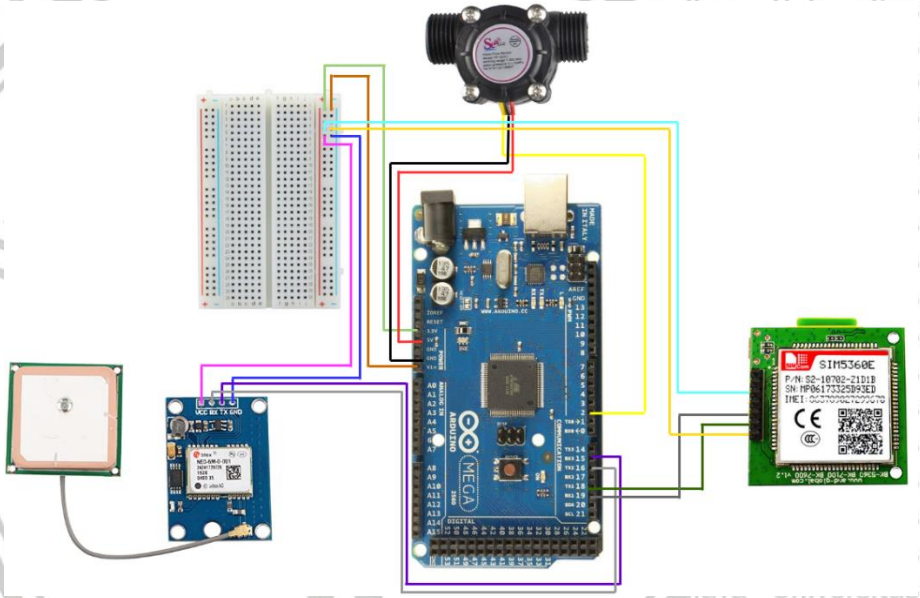
Sedangkan *subscriber* yang digunakan yaitu membangun web lokal pada laptop yang bertugas untuk melakukan *request* data sesuai topiknya kepada *broker* yang nantinya data tersebut ditampilkan pada halaman web untuk kebutuhan monitoring dan deteksi kebocoran pada jaringan pipa. Proses *request* topik menggunakan Bahasa pemrogaman PHP dengan fungsi GET dengan memasukkan IO *username* dan IO *key* kedalam fungsi *getData*() lalu dikeluarkan dalam *output* berupa data JSON. Dari data JSON tersebut maka ditampilkan pada halaman web. Struktur perancangan format data pada *subscriber* dapat dilihat pada Gambar 3.8 yang berisi PHP GETDATA() yang fungsinya untuk mengambil data dari *broker*

dengan memasukkan alamat, io key, topik yang diinginkan, dan kemudian dilakukan json decode untuk menerjemahkan *string* akan mudah diolah. Isi dari data yang berhasil didapatkan yaitu *id*, *feed id*, *value* untuk nilai debit air, dan *location* untuk titik koordinat. Data yang berhasil diambil/*subscribe* dan telah diolah maka selanjutnya ditampilkan pada halaman web menggunakan fungsi ajax untuk mengambil *value* dan *location* serta javascript setInterval sebagai fungsi *refresh* beberapa fungsi untuk mengambil data tersebut.

3.2.1 Perancangan Hardware

Perancangan hardware adalah tahap penjelasan rancangan perangkat keras seperti jenis atau nama perangkat serta fungsi masing-masing perancangan hardware yang digunakan pada penelitian ini. Terdapat tiga perancangan hardware yaitu perancangan *publisher*, perancangan *broker*, dan perancangan *subscriber*.

3.2.1.1 Perancangan *Publisher*



Gambar 3.9 Perancangan Hardware *Publisher*

Pada Gambar 3.9 adalah skema rancangan perangkat keras untuk mendapatkan nilai debit air yang mengalir pada jaringan pipa menggunakan sensor *flow meter* YF-S201 dan modul GPS U-Blox NEO 6M untuk memberikan informasi lokasi jika terjadi kebocoran pada jaringan pipa. Sensor *flow meter* YF-S201dan modul GPS U-Blox NEO 6M akan terus menerus melakukan *sensing* lalu dikirim ke mikrokontroler agar dilakukan proses pengumpulan dan pengolahan data yang nanti dikirim menuju *broker* dengan menggunakan sumber Internet 3G yang berasal dari modul SIM5360E yang didalamnya terpasang kartu perdana Telkomsel. Data yang di *publish* menuju *broker* berdasarkan topiknya masing-masing dengan menggunakan protokol *MQTT*.

Tabel 3.1 Pinout Sensor YF-S201

YF-S201 *Flow Meter*	Arduino Mega 2560
Signal (Kabel warna kuning)	Digital PWM (D2)
GND (Kabel warna hitam)	GND
VCC (Kabel warna merah)	5V

Pada Tabel 3.1 adalah konfigurasi *pinout* antara sensor YF-S201 dengan Arduino Mega 2560. Pin signal pada YF-S201 digunakan untuk mengirim data hasil *sensing* yang disimbolkan dengan warna kuning yang dihubungkan ke pin digital pwm (D2) milik Arduino mega 2560 menggunakan kabel pin *female to female*, lalu pada pin GND atau *ground* disimbolkan dengan warna hitam yang dihubungkan ke pin GND milik Arduino Mega 2560 menggunakan kabel pin *female to female*, sedangkan pin VCC digunakan untuk mendapatkan *power* atau energi listrik yang disimbolkan dengan warna merah yang dihubungkan ke pin 5V milik Arduino Mega 2560 menggunakan kabel pin *female to female.*

Tabel 3.2 Pinout Modul SIM5360E

SIM5360E	Project Board	Arduino Mega 2560
V (Kabel warna biru muda)	+	3.3v
G (Kabel warna oranye)	-	Vin
R (Kabel warna hijau tua)	Null	Tx1 (18)
T (Kabel warna abu-abu tua)	Null	Rx1 (19)

Pada Tabel 3.2 adalah konfigurasi *pinout* antara SIM5360E dengan Arduino Mega 2560. Terdapat pin Tx dan Rx pada SIM5360E digunakan untuk proses mengirim dan menerima data yang dihubungkan dengan pin Tx dan Rx namun saling berlawanan (T-Rx1 dan R-Tx1) milik Arduino Mega 2560. *Project board* berguna untuk memberikan alokasi pin + (*plus*) dan – (*minus*) yang telah dijumper oleh Arduino Mega 2560 dengan sumber pin 3.3v dan Vin. Pin V pada SIM5360E berguna untuk mendapatkan *power* atau energi listrik yang dihubungkan dengan pin + pada *project board*, sedangkan pin G atau *ground* pada SIM5360E terhubung pada port pin – pada *project board*.

Tabel 3.3 Pinout Modul U-Blox NEO 6M

U-Blox NEO 6M	Project Board	Arduino Mega 2560
Tx (Kabel warna ungu)	Null	Rx3 (15)
Rx (Kabel warna abu-abu muda)	Null	Tx3 (16)
VCC (Kabel warna merah muda)	+	3.3v
GND (Kabel warna biru)	-	Vin

Pada Tabel 3.3 adalah konfigurasi *pinout* antara modul GPS U-Blox NEO 6M dengan Arduino Mega 2560. Terdapat pin Tx dan Rx pada U-Blox NEO 6M digunakan untuk proses mengirim dan menerima data yang dihubungkan dengan pin Tx dan Rx namun saling berlawanan (Tx-Rx3 dan Rx-Tx3) milik Arduino Mega 2560. *Project board* berguna untuk memberikan alokasi pin + (*plus*) dan – (*minus*) yang telah dijumper oleh Arduino Mega 2560 dengan sumber pin 3.3v dan Vin. Pin VCC pada U-Blox NEO 6M berguna untuk mendapatkan *power* atau energi listrik yang dihubungkan dengan pin + pada *project board*, sedangkan pin GND atau *ground* pada U-Blox NEO 6M terhubung pada port pin – pada *project board*.

3.2.1.2 Perancangan *Broker*

Perancangan *broker* yaitu menggunakan laptop dengan spesifikasi processor intel i5, RAM 8GB, dan storage SSD 250GB yang akan mengakses *cloud service* bernama Adafruit yang bertugas sebagai *broker* pada penelitian ini. *Broker* memiliki fungsi sebagai penjembatan komunikasi antara *publisher* dengan *subscriber*.

3.2.1.3 Perancangan *Subscriber*

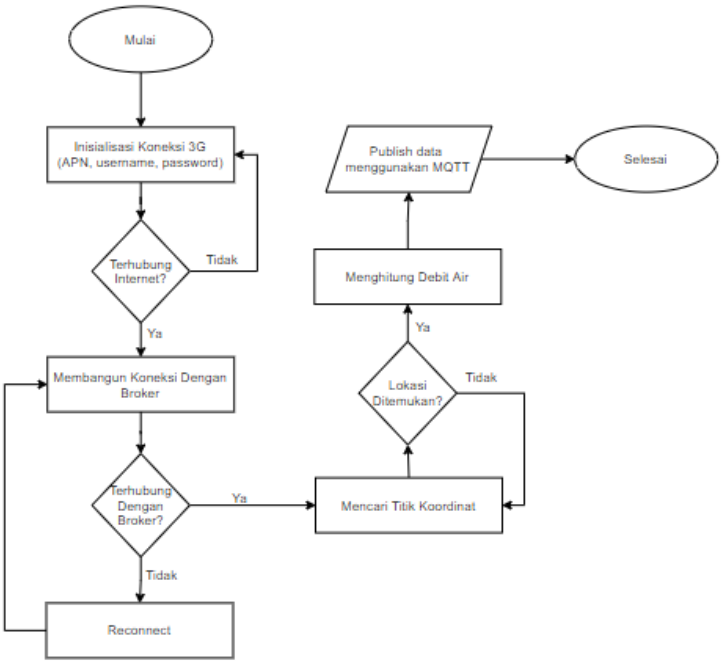
Perancangan *subscriber* pada penelitian ini yaitu berbasis web yang dijalankan secara lokal. Peran *subscriber* adalah untuk mengambil data dari *broker* berdasarkan topiknya lalu ditampilkan pada halaman web secara visual untuk kebutuhan monitoring debit air serta memberikan notifikasi jika terjadi kebocoran pada jaringan pipa.

3.2.2 Perancangan *Software*

Perancangan software adalah tahap untuk menjelaskan rancangan perangkat lunak seperti aplikasi yang digunakakan dan bagaimana alur proses dari sistem pada penelitian ini. Terdapat tiga perancangan *software* yaitu terdiri dari perancangan

3.2.2.1 Perancangan *Publisher*

Perancangan *publisher* yang dibangun menggunakan Bahasa pemrograman C yang ditanamkan pada perangkat mikrokontroler melalui perangkat lunak Arduino IDE. *Publisher* terdiri dari 3 *node* yang masing-masing terhubung dengan modul Internet, *water* flow, dan modul GPS. Setiap node mempunyai tugas untuk melakukan *publish* data yang berasal dari *sensing water flow* dan modul GPS (*longitude* dan *latitude*) yang terhubung ke node *publisher* sehingga setelah semua data terkumpul maka *publisher* melakukan *publish*. *Publish* data yang dilakukan oleh *publisher* menggunakan protokol MQTT dengan Qos Level 0 (*at most once*/*best effort*) menuju *broker* berdasarkan topik masing-masing yaitu topik pdam, pdam2, dan pdam3. Secara umum alur proses *publisher* dari penelitian ini adalah konektivitas internet, mendapatkan data hasil *sensing*, dan pengiriman data yang dapat dilihat pada Gambar 3.10.



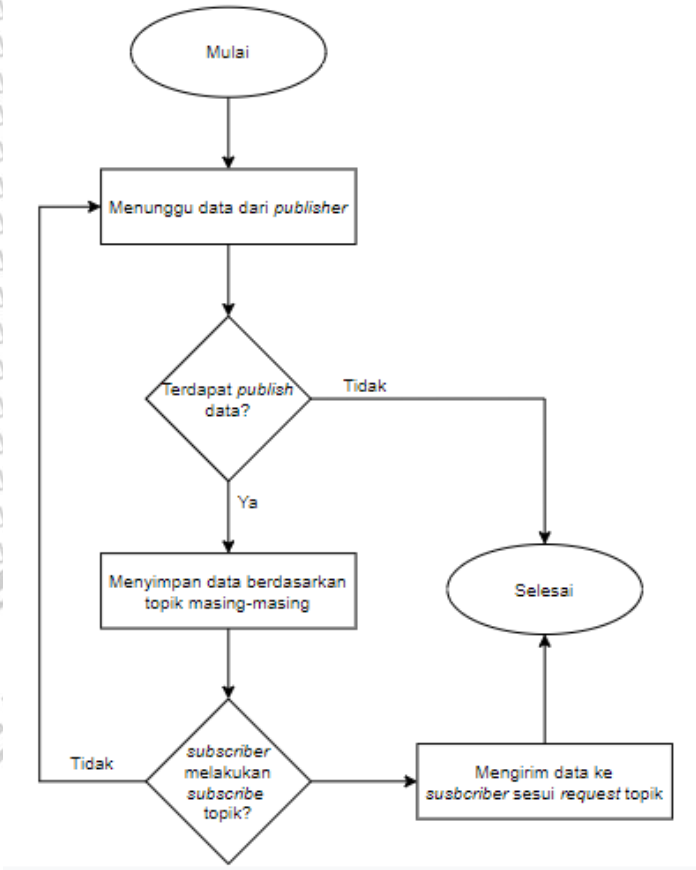
Gambar 3.10 Skema Diagram Alir *Publisher*

Berikut adalah penjelasan detail dari diagram diatas yang dimana dirancang sebagai *publisher*:

1. Proses awal adalah membangun koneksi Internet 3G yaitu dengan cara memasukkan nama APN, *user*, dan *password* sesuai dengan kartu provider yang digunakan yaitu Telkomsel.
2. Ketika telah mendapatkan koneksi Internet berbasis 3G, selanjutnya membangun komunikasi dengan *broker* adafruit dengan cara memasukkan *username* dan IO key yang secara otomatis di *generate* oleh *broker* Adafruit.
3. Jika *publisher* telah terhubung dengan *broker* maka *publisher* mulai mencari lokasi berupa titik koordinat (*longitude* dan *latitude*) secara terus menerus hingga menemukan lokasi saat ini karena jika lokasi tidak ditemukan maka akan tampil INVALID.
4. Apabila titik koordinat lokasi telah ditemukan maka selanjutnya sensor *flow meter* mulai menghitung debit air.
5. Data berupa titik koordinat dan debit air di *publish* menuju *broker* berdasarkan topiknya masing-masing.

3.2.2.2 Perancangan *Broker*

Perancangan *broker* yaitu menggunakan adafruit yang berbasis *online broker*. Tugas *broker* adalah sebagai penjembatan komunikasi antara *publisher* dengan *subscriber*. *Broker* menyimpan data dari *publisher* yang kemudian dikirim menuju *subscriber* sesuai topik yang diminta. Secara umum alur proses *broker* dari penelitian ini dapat dilihat pada Gambar 3.11.



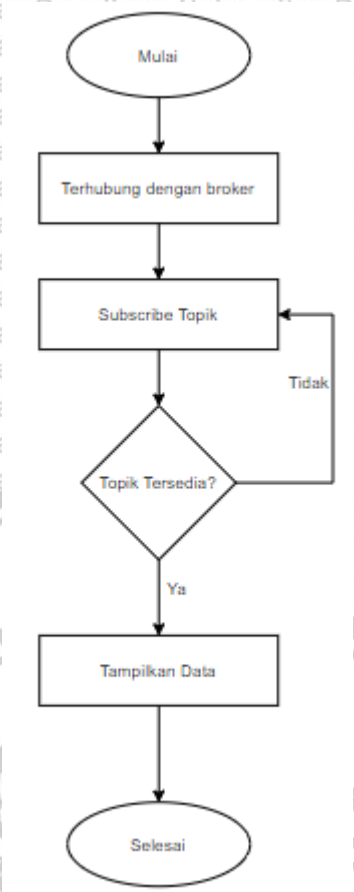
Gambar 3.11 Skema Diagram Alir *Broker*

Berikut adalah penjelasan detail dari diagram diatas yang dimana dirancang untuk *broker*:

1. Proses awalnya adalah *broker* menunggu data yang dipublish dari *publisher*.
2. Jika terdapat data yang di *publish*, maka data tersebut disimpan ke direktori berdasarkan topik masing-masing dan menunggu datangnya *request* dari *subscriber*.
3. Ketika *subscriber* melakukan *request* topik maka *broker* memberikan data tersebut kepada *subscriber* sesuai dengan topik yang diminta.

3.2.2.3 Perancangan *Subscriber*

Subscriber pada penelitian ini yaitu berupa web yang nantinya dapat diakses melalui web browser. Fungsi *subscriber* adalah menampilkan data hasil *request* topik dari *broker* atau biasa disebut dengan proses *subscribe*. Data yang diterima akan ditampilkan secara visual untuk mempermudah proses monitoring sistem. Secara umum alur proses *subscriber* dari penelitian ini dapat dilihat pada Gambar 3.12.



Gambar 3.12 Skema Diagram Alir *Subscriber*

Berikut adalah penjelasan detail dari diagram diatas yang dimana dirancang sebagai *subscriber*:

1. Langkah awalnya adalah *subscriber* membangun koneksi terlebih dahulu dengan *broker*.
2. Kemudian melakukan proses *subscribe* sesuai topik yang diinginkan.
3. Jika topik tersedia pada sisi *broker*, maka *broker* akan mengirimkan data tersebut dan diterima oleh *subscriber*.
4. Data yang diterima selanjutnya ditampilkan secara visual melalui web.

3.3 Metode Evaluasi

Melakukan evaluasi terhadap sistem yang dibangun perlu dilakukan karena untuk mengetahui apakah sistem telah berjalan sesuai dengan rancangan sebelumnya atau tidak sesuai. Proses pengujian dilakukan dengan membangun jaringan pipa air secara simulasi dengan skala ukuran yang kecil dan mengalirkan air secara terus menerus serta memberikan beberapa titik kebocoran. Pengujian dibagi menjadi dua bagian yaitu pengujian fungsional dan pengujian kinerja sistem.

3.3.1 Perancangan Pengujian Fungsional

Pengujian fungsional merupakan proses pengujian beberapa fungsi yang terdapat pada sistem. Skema pengujian pada penelitian ini yaitu dengan menguji apakah sistem yang dibuat telah berjalan dengan baik dan sesuai dengan rancangan sebelumnya. Pada pengujian fungsional terbagi menjadi tiga yaitu pengujian *publisher*, *broker*, dan *subscriber*.

3.3.3.1 Perancangan Pengujian *Publisher*

Pengujian *publisher* yaitu pengujian tentang sistem yang dapat melakukan proses pengambilan data hasil *sensing* dari sensor flow meter dan modul GPS yang kemudian dikirim menuju *broker* menggunakan protokol *MQTT*. Terdapat 3 perangkat mikrokontroler sebagai *publisher* maka dilakukan pengujian secara bersamaan. Rancangan pengujiannya yaitu untuk melihat apakah *publisher* dapat melakukan *publish* data debit air dengan 3 kondisi yaitu tidak ada aliran air, terdapat aliran air (kondisi normal), dan apabila terjadi kebocoran. Hal ini dilakukan untuk mengetahui kemampuan *publisher* dalam membaca data sesuai dengan kondisi yang diberikan.

3.3.3.2 Perancangan Pengujian *Broker*

Pengujian *broker* yaitu pengujian tentang proses *broker* pada saat menerima data dari *publisher* yang nanti akan dikirim menuju *subscriber* sesuai topik yang diinginkan. Rancangan pengujian pada *broker* yaitu mengetahui apakah *broker* dapat menerima data *publish* dari *publisher* dengan 3 kondisi yaitu tidak ada aliran air, terdapat aliran air (kondisi normal), dan terjadi kebocoran. Setiap kondisi akan memiliki nilai yang berbeda-beda, maka dari itu perlu dicocokkan antara data *publish* telah sesuai dengan data yang diterima oleh *broker* dengan 3 kondisi tersebut.

3.3.3.3 Perancangan Pengujian *Subscriber*

Pengujian *subscriber* yaitu tentang bagaimana *subscriber* dapat menampilkan data hasil *subscribe* dari *broker*. Pada *subscriber* telah memberikan nilai batas ambang kurang lebih 30% dari kondisi normal, hal ini digunakan untuk proses komputasi menentukan terjadi indikasi kebocoran atau normal. Pengujian pada *subscriber* yaitu melihat hasil komputasi debit air dengan cara membandingkan nilai batas ambang terhadap 3 kondisi yaitu tidak ada aliran air, terdapat aliran air (kondisi normal), dan terjadi kebocoran. Kondisi normal apabila debit air tidak menyentuh atau dibawah nilai batas ambangnya sehingga kondisi monitoring pada halaman web berwarna hijau, sedangkan jika debit air telah mencapai atau kurang dibawah nilai batas ambangnya maka terjadi kebocoran sehingga warna ikon berubah menjadi warna merah. Pada *subscriber* juga dilakukan pengujian utilitas peforma CPU *usage* dan RAM *usage* menggunakan Bahasa Python yang diukur pada saat melakukan proses *subscribe* 1 node, 2 node, dan 3 node. Proses pengujian peforma dibantu menggunakan aplikasi JMeter yang bertugas sebagai *publisher* yang secara terus menerus melakukan *publish* data selama 5 menit.

3.3.2 Perancangan Pengujian Performa

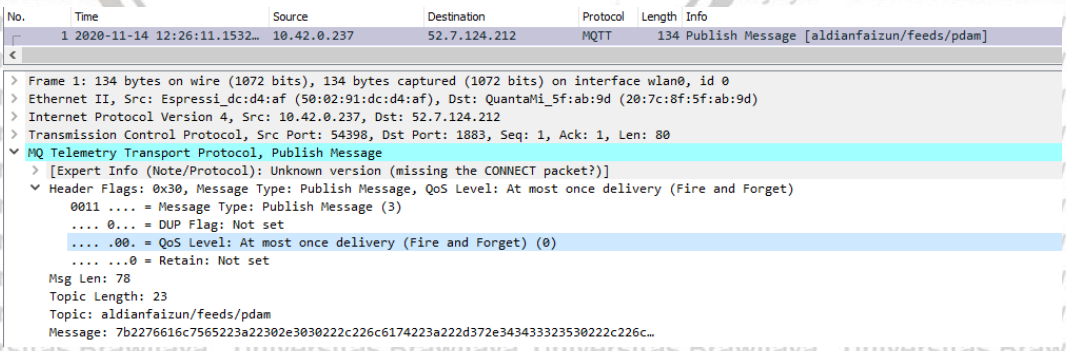
Pengujian kinerja sistem dilakukan dengan menguji kinerja dari protokol *MQTT* dengan beberapa parameter yaitu *packet loss* dan *delay*.Pengujian pertama yang dilakukan adalah pengujian *packet loss* yang bertujuan untuk mengetahui berapa banyak data yang hilang atau tidak berhasil mencapai tujuan dalam bentuk prosentase. Selanjutnya yaitu pengujian *delay* untuk mengetahui lama waktu yang dibutuhkan paket data saat melakukan transmisi data dari sumber ke tujuan.

3.4Implementasi Sistem

Implementasi sistem pada penelitian ini akan mewujudkan rancangan yang telah dibuat sebelumnya sehingga sistem deteksi kebocoran dan monitoring debit air dapat bekerja dengan baik dan sesuai. Tahapan implementasi terdiri dari implementasi *publisher*, implementasi *broker*, dan implementasi *subscriber*.

3.4.1 Implementasi *Publisher*

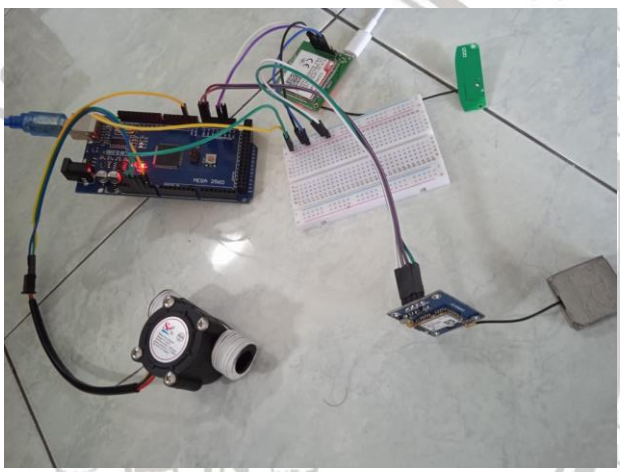
Publisher pada penelitian ini adalah sebuah perangkat yang berperan sebagai mendapatkan data seperti debit air dan titik koordinat yang akan dikirimkan menuju *broker* menggunakan koneksi internet 3G dengan protokol komunikasi *MQTT*. Masing-masing *publisher* akan melakukan *publish* data sebesar kurang lebih 2 Byte. Satu pesan yang berisi nilai debit air dan titik lokasi dikirim sebesar 2 *Byte* (*fixed header*) terdiri dari beberapa bit yang memiliki fungsi masing-masing. Bit 0 sampai 3 yaitu *message type* yang akan diisi dengan *control packet* dengan tipe *PUBLISH* karena metode pengiriman *publisher* pada penelitian ini adalah *fire and forget* atau biasa disebut dengan QoS Level 0. Selanjutnya yaitu bit ke-4 DUP yaitu untuk mengetahui apakah pesan yang dikirim baru atau mengirim ulang, sedangkan jenis QoS yang digunakan adalah level 0 sehingga tidak akan ada pengulangan pesan pada saat transmisi, maka nilai DUP *flag* pada bit ke-4 adalah 0. Lalu pada bit ke 5 dan 6 diisi dengan jenis QoS Level yang digunakan yaitu level 0 dan yang terakhir bit ke 7 yaitu RETAIN yang secara *default* nilainya *set true*. Untuk lebih jelasnya pada Gambar 3.13 adalah detail paket yang dimiliki oleh node A.



```
Message: 7b2276616c7565223a22302e3030222c226c6174223a222d372e343433323530222c226c…

0030  08 5c 92 b7 00 00 30 4e  00 17 61 6c 64 69 61 6e   ·\····0N ··aldian
0040  66 61 69 7a 75 6e 2f 66  65 65 64 73 2f 70 64 61   faizun/f eeds/pda
0050  6d 7b 22 76 61 6c 75 65  22 3a 22 30 2e 30 30 22   m{"value ":"0.00"
0060  2c 22 6c 61 74 22 3a 22  2d 37 2e 34 34 33 32 35   ,"lat":" -7.44325
0070  30 22 2c 22 6c 6f 6e 22  3a 22 31 31 32 2e 36 39   0","lon" :"112.69
0080  39 39 36 30 22 7d                                 9960"}
```

Gambar 3.13 Struktur Implementasi Publisher

Pesan pada *publisher* akan dikirim menuju *broker* dengan menyertakan topiknya dengan format (*username/feeds/topic*). Terdapat 3 topik yang digunakan yaitu pdam, pdam2, dan pdam3. Pesan yang dikirim oleh *publisher* dilakukan setiap 5 detik sekali untuk menghindari *flooding* karena akun yang terdaftar pada *broker* bersifat *free* maka terdapat keterbatasan layanan. Pada Gambar 3.14 adalah bentuk fisik dari *publisher*.



Gambar 3.14 Implementasi *Publisher*

Spesifkasi *hardware* yang digunakan pada *publisher* yaitu terdiri dari Mikrokontroller Arduino Mega2560, sensor *water flow* YF-S201 untuk menghitung debit air yang mengalir, modul GPS U-Blox NEO 6M untuk mencari lokasi berupa titik koordinat (*longitude* dan *latitude*), modul 3G SIM5360E sebagai koneksi Internet yang menggunakan kartu perdana Telkomsel, *project board* untuk manajemen kabel, dan kabel pin sebanyak 11 item untuk menghubungkan modul ke mikrokontroller. Spesifikasi *software* pada *publisher* hanya menggunakan aplikasi Arduino IDE sebagai *text editor source code* dan kompilasi program untuk ditanamkan ke mikrokontroller. Implementasi *publisher* dapat dilihat pada Tabel 3.4.

Tabel 3.4 Implementasi *Publisher*

Algoritme 1: *Publisher*	
1	`#define TINY_GSM_MODEM_SIM5360`
2	`#define TINY_GSM_DEBUG SerialMon`
3	`#define GSM_AUTOBAUD_MIN 9600`
4	`#define GSM_AUTOBAUD_MAX 115200`

```
5
6   #define TINY_GSM_USE_GPRS true
7   #define TINY_GSM_USE_WIFI false
8
9   #define GSM_PIN ""
10  const char apn[] = "internet";
11  const char gprsUser[] = "wap";
12  const char gprsPass[] = "wap123";
13
14  #if TINY_GSM_USE_GPRS
15    SerialMon.print(F("Connecting to "));
16    SerialMon.print(apn);
17
18    if (!modem.gprsConnect(apn, gprsUser, gprsPass)) {
19      SerialMon.println(" fail");
20      delay(10000);
21      return;
22    }
23    SerialMon.println(" success");
24
25    if (modem.isGprsConnected()) {
26      SerialMon.println("GPRS Connected");
27    }
28  #endif
29  . . .
30  #include <TinyGPS++.h>
31  float lat, lon;
32  TinyGPSPlus gps;
33
34  void gpsInfo() {
35    Serial.print(F("Location: "));
36
37    if (gps.location.isValid())
38     {
39      Serial.print("Latitude=");
40      Serial.print(gps.location.lat(), 6);
41      Serial.print(F(", "));
42      Serial.print("Longitude=");
43      Serial.println(gps.location.lng(), 6);
44
45      water flowInfo();
46  pubData(String(flowLitres), String(gps.location.lat(), 6),
47  String(gps.location.lng(), 6));
48     }
49    else
50    {
51      Serial.println(F("INVALID"));
52    }
53    Serial.println();
54    }
55  ...
56  byte sensorInterrupt = 0;
57  byte sensorPin       = 2
58  float calibrationFactor = 4.5;
59  volatile byte pulseCount;
60  float flowRate;
61  float flowLitres;
62  float totalLitres;
63
64  unsigned int flowMilliLitres;
65  unsigned long totalMilliLitres;
66  unsigned long oldTime;
67
68  pulseCount        = 0;
69  flowRate          = 0.0;
```

```
70  flowMilliLitres   = 0;
71  totalMilliLitres  = 0;
72  oldTime           = 0;
73
74  void water flowInfo() {
75    detachInterrupt(sensorInterrupt);
76    flowRate = ((1000.0 / (millis() - previousMillis)) *
77  pulseCount) / calibrationFactor;
78
79    previousMillis = millis();
80    flowLitres = (flowRate / 60);
81    totalLitres += flowLitres;
82
83    unsigned int frac;
84
85    Serial.print("flowrate: ");
86
87    Serial.print(int(flowRate));
88
89    Serial.print(".");
90
91    frac = (flowRate - int(flowRate)) * 10;
92
93    Serial.print(frac, DEC) ;
94
95    Serial.print("L/min");
96
97    Serial.print("  Current Liquid Flowing: ");
98    Serial.print(flowLitres);
99    Serial.print("L / 5 Sec");
100   Serial.print("  Output Liquid Quantity: ");
101
102   Serial.print(totalLitres);
103   Serial.print("L");
104
105   pulseCount = 0;
106
107   attachInterrupt(sensorInterrupt, pulseCounter, FALLING);}
```

Sumber *source code* untuk dapat menghitung debit air yang mengalir yaitu dapat dilihat pada url berikut https://github.com/Almazi/Waterflow-Measurement-Arduino/blob/master/WaterFlow_Demo2.ino. Uraian source code diatas adalah baris nomor 1 sampai 4 adalah mendefinisikan jenis modul apa yang digunakan serta memberikan Batasan komunikasi baudrate. Baris nomor 6 dan 7 yaitu membuat definisi dengan dua kondisi yaitu menggunakan GPRS atau WIFI, pada penelitian ini memberikan nilai *true* pada kondisi TINY_GSM_USE_GPRS. Baris 9 adalah input pin apabila kartu perdana yang digunakan dalam kondisi terkunci, pada kondisi ini dikosongkan. Baris 10 sampai 12 adalah mendaftarkan APN, *username*, dan *password* agar kartu perdana dapat menemukan koneksi yang stabil dan dapat berkomunikasi dengan BTS. Baris nomor 14 sampai 28 yaitu kondisi ketika menggunakan koneksi 3G dan apabila nama apn, username dan password sesuai maka pada serial monitor menghasilkan *success* dan jika telah terhubung ke BTS maka keluar status GPRS *Connected* dan apabila gagal maka akan mengeluarkan *fail*. Pada Gambar 3.9 adalah hasil implementasi koneksi internet menggunakan modul 3G SIM5360E

Selanjutnya *source code* untuk memberikan informasi lokasi berupa titik koordinat *latitude* dan *longitude*. Baris nomor 30 sampai 32 adalah untuk

memanggil *library* GPS lalu dibuatkannya variabel untuk *latitude* dan *longitude* dengan tipe data *float*. Baris nomor 34 sampai 54 adalah method untuk memberikan informasi titik koordinat yang didalamnya terdapat 2 kondisi. Kondisi pertama jika lokasi itu valid atau ditemukan maka modul GPS akan mengeluarkan nilai *latitude* dan *longitude* serta memanggil method water flow yang telah dibuat sebelumnya dan kemudian *publish* semua data tersebut ke *broker* lalu ditampilkan ke serial monitor. Namun jika informasi tidak ditemukan maka pada tampilan serial monitor menghasilkan "INVALID" secara terus menerus hingga mendapatkan titik koordinat.

Kemudian *source code* untuk implementasi monitoring debit air menggunakan YF-S201. Baris nomor 56 dan 57 adalah untuk inisialisasi komunikasi transmisi data kabel pin sensor water flow. Baris nomor 58 sampai 62 adalah menentukan angka kalibrasi nilai sensor serta mendeklarasikan variabel untuk menampung nilai rata-rata air mengalir, air yang mengalir saat ini, dan total air yang mengalir. Baris nomor 64 sampai 72 untuk memberikan nilai awal dengan angka 0 kepada variabel yang telah dibuat sebelumnya. Baris nomor 74 sampai 108 adalah sebuah *method* yang berisikan rumus kepada masing-masing variabel untuk menghitung rata-rata, kondisi saat ini, dan total yang mengalir serta menampilkannya ke serial monitor dengan string tertentu untuk mempermudah pembacaannya. Method water flowInfo() ini akan dipanggil pada method gpsInfo() untuk kebutuhan data perhitungan air yang kemudian di *publish* menuju *broker*. Sumber *source code* agar modul SIM5360E dapat terkoneksi dengan Internet yaitu tertuju pada url berikut https://github.com/vshymanskyy/TinyGSM/blob/master/examples/MqttClient/MqttClient.ino.

```
Wait...
[9] Trying baud rate 115200 ...
[1035] Modem responded at rate 115200
Initializing modem...
[29933] ### TinyGSM Version: 0.10.9
[29933] ### TinyGSM Compiled Module:  TinyGsmClientSIM5360
[29984] ### Modem: SIMCOM SIM5360E
[29985] ### Modem: SIMCOM SIM5360E
[31028] ### Unhandled: Manufacturer: SIMCOM INCORPORATED
Model: SIMCOM_SIM5360E
Revision: SIM5360E_V3.5M8524+PC,,
K
Modem Info:
Waiting for network... success
Network connected
Connecting to internet success
GPRS connected

Location: Latitude=-7.443320, Longitude=112.699905
flowrate: 0.0L/min  Current Liquid Flowing: 0.00L / 5 Sec  Output Liquid Quantity: 0.00LPublish ok

Location: Latitude=-7.443320, Longitude=112.699905
flowrate: 0.0L/min  Current Liquid Flowing: 0.00L / 5 Sec  Output Liquid Quantity: 0.00LPublish ok

Location: Latitude=-7.443320, Longitude=112.699905
flowrate: 0.0L/min  Current Liquid Flowing: 0.00L / 5 Sec  Output Liquid Quantity: 0.00LPublish ok

Location: Latitude=-7.443320, Longitude=112.699905
flowrate: 0.0L/min  Current Liquid Flowing: 0.00L / 5 Sec  Output Liquid Quantity: 0.00LPublish ok

Location: Latitude=-7.443320, Longitude=112.699905
flowrate: 0.0L/min  Current Liquid Flowing: 0.00L / 5 Sec  Output Liquid Quantity: 0.00LPublish ok
```

Gambar 3.15 Hasil Implementasi Monitoring Debit Air

3.4.2 Implementasi *Broker*

Broker pada penelitian ini yaitu menggunakan *service cloud* yang menyediakan layanan *broker* secara gratis yaitu bernama Adafruit IO yang dapat diakses melalui browser dan buat akun terlebih dahulu agar dapat menggunakan layanan *broker* tersebut lalu membuat dashboard untuk menampung topik yang akan dikirim oleh *publisher*. Pesan data yang dikirim oleh *publisher* akan disimpan berdasarkan topiknya masing-masing yaitu pdam, pdam2, dan pdam3. Data pada *broker* akan terus disimpan sampai terdapat *request* dari *subscriber* yang kemudian dicocokkan lalu di *publish* menuju *subscriber*. Pada Gambar 3.16 adalah beberapa pesan yang ditampung oleh *broker* dengan nama topik pdam. Isi dari pesan pada masing-masing topik yaitu *created at* (warna kuning) yang secara *default* di *generate* oleh *broker* untuk informasi tanggal kedatangan paket yang berhasil masuk, *value* (warna hijau) untuk menyimpan nilai debit air, dan *location* (warna biru) untuk titik koordinatnya yang berupa *longitude* dan *latitude*. Pesan yang disimpan pada *broker* telah sesuai dengan struktur pesan yang dikirim oleh *publisher* yang isinya *value* dan *location*.

Created at	Value	Location
2021/07/09 8:57:13AM	0.13	-7.443323, 112.70008
2021/07/09 8:57:08AM	0.13	-7.443323, 112.70008
2021/07/09 8:57:03AM	0.13	-7.443323, 112.70008
2021/07/09 8:56:58AM	0.13	-7.443323, 112.70008
2021/07/09 8:56:53AM	0.13	-7.443323, 112.70008
2021/07/09 8:56:48AM	0.13	-7.443323, 112.70008
2021/07/09 8:56:43AM	0.13	-7.443323, 112.70008
2021/07/09 8:56:38AM	0.13	-7.443323, 112.70008
2021/07/09 8:56:33AM	0.13	-7.443323, 112.70008

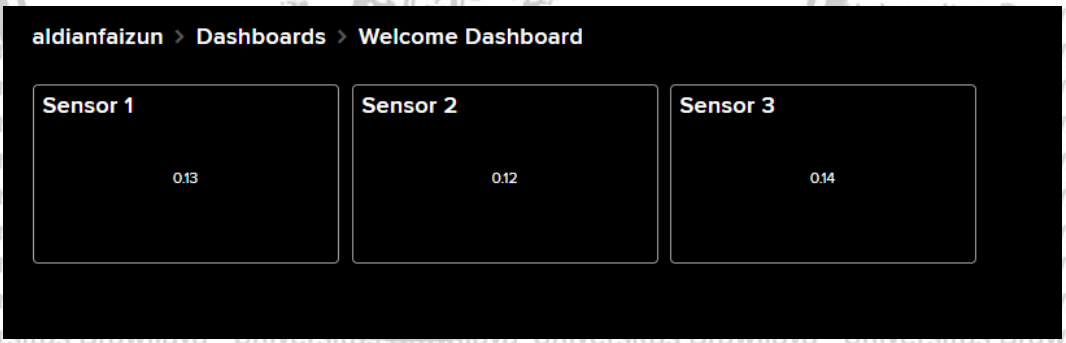
Gambar 3.16 Struktur Implementasi *Broker*

Berikut pada tabel 3.5 adalah gambaran berupa pseudocode dari *broker*.

Tabel 3.5 Implementasi *Broker*
(Sumber: Pratama & Nurwarsito, 2019)

Algoritme 3: *Broker*	
1	MULAI
2	CONNECT *publisher*
3	READ topik dan payload
4	CONNECT *subscriber*
5	READ subscribe topik dari *subscriber*
6	IF topik = payload topik
7	SEND payload ke *subscriber*
8	ELSE IF
9	topik *subscriber* ≠ payload topik
10	HOLD data payload
11	SELESAI

Uraian dari pseudocode diatas adalah baris 2 dan 3 yaitu tentang membangun koneksi dengan *publisher* dan ketika koneksi telah terhubung dengan *publisher* maka *broker* menunggu payload yang dikirim oleh *publisher*. Setelah *payload* masuk maka disimpan berdasarkan topiknya untuk dicocokkan dengan permintaan dari *subscriber* nanti. Selanjutnya pada baris 4 dan 5 adalah kondisi membangun koneksi dengan *subscriber* dan *broker* menerima permintaan topik dari *subscriber*. Baris 6 dan 7 adalah kondisi ketika topik yang diminta terdapat pada payload topik *broker* maka data yang sesuai akan dikirimkan kepada *subscriber*. Selanjutnya pada baris 8 sampai 10 yaitu kondisi Kondisi ketika topik *subscriber* tidak sesuai dengan topik pada *broker* maka data tersebut akan ditahan oleh *broker*. Pada Gambar 3.10 adalah tampilan dashboard implementasi *broker*.

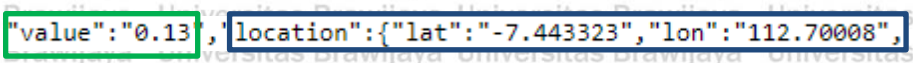


Gambar 3.17 Hasil Implementasi *Broker*

3.4.3 Implementasi *Subscriber*

Implementasi *subscriber* yaitu terletak pada sisi *client* yang bertugas melakukan *subscribe* topik yang diinginkan dan tersedia pada *broker*. *Subscriber* pada penelitian ini dibangun menggunakan web yang akan menampilkan data yang berhasil dilakukan *subscribe*. Tugas dari *subscriber* disini adalah untuk melakukan proses *subscribe* kepada *broker* dengan mengirimkan *request* dengan topik yang diinginkan. Data yang diterima dari *broker* akan diolah menggunakan beberapa fungsi. Metode yang digunakan untuk melakukan *request* yaitu menggunakan fungsi ajax yang nantinya data tersebut akan dilakukan json decode

untuk memudahkan dalam pembacaan string yang nanti akan ditampilkan pada halaman web. Berikut pada Gambar 3.18 adalah struktur data saat *subscriber* melakukan proses *request* kepada *broker*. Struktur pesan yang terdapat pada *broker* telah sesuai apa yang didapatkan oleh *subscriber* untuk node A. Terlihat bahwa *value* (warna hijau) adalah untuk menampung nilai debit air untuk kebutuhan monitoring dan *location* (warna biru) untuk informasi lokasi titik kebocoran berupa titik koordinat (*longitude* dan *latitude*).



Gambar 3.18 Struktur Implementasi Subscriber

Ketika pesan tersbeut telah berhasil didapatkan maka data tersebut akan ditampilkan pada halaman web. Pada Tabel 3.6 adalah *source code* implementasi *subscriber* berbasis web untuk deteksi kebocoran dan monitoring debit air pada jaringan pipa.

Tabel 3.6 Implementasi *Subscriber* Get Data

Algoritme 3: *Subscriber* Get Data

```
1   protected $urladafruitio
2   ='https://io.adafruit.com/api/v2/aldianfaizun/feeds/';
3
4   protected $iokey = 'aio_CKTd93bHVSqjr3ve62qundrpzjIQ';
5
6   public function getDataA() {
7       $url = $this->urladafruitio . 'pdam/data/last';
8       $ch = curl_init(); // Initialize cURL
9       curl_setopt($ch, CURLOPT_URL, $url);
10      curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "GET");
11      curl_setopt($ch, CURLOPT_POSTFIELDS, array('X-AIOKey' =>
12  $this->iokey));
13
14      curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
15      $response = curl_exec($ch);
16
17      curl_close($ch);
18
19      $output = json_decode($response, true);
20
21      return $output;
22  }
```

Sumber *source code* pada proses ini yaitu mengacu pada https://www.petanikode.com/php-curl/ namun telah dimodifikasi sesuai kebutuhan pada penelitian ini. Pada Tabel 3.6 menjelaskan bagaimana proses *subscribe* yang diawali dengan baris nomor 1 dan 2 untuk memasukkan alamat *broker* serta *key* yang digunakan agar bisa terhubung. Baris nomor 6 sampai 22 adalah method untuk node A pada saat pengambilan topik kepada *broker* yang nanti hasil yang dikeluarkan akan berupa data json lalu ditampilkannya ke halaman web. Pada penelitian ini menggunakan tiga node maka untuk node B dan node C juga memiliki method yang sama namun yang membedakannya terletak pada baris nomor 7 yaitu tentang topik apa yang ingin di *subscribe*.

File konten.php menyediakan tentang tampilan pada halaman web untuk kebutuhan deteksi kebocoran dan monitoring debit air yang dapat diakses melalui browser. Implementasi konten.php dapat dilihat pada Tabel 3.7

Tabel 3.7 Implementasi Konten *Subscriber*

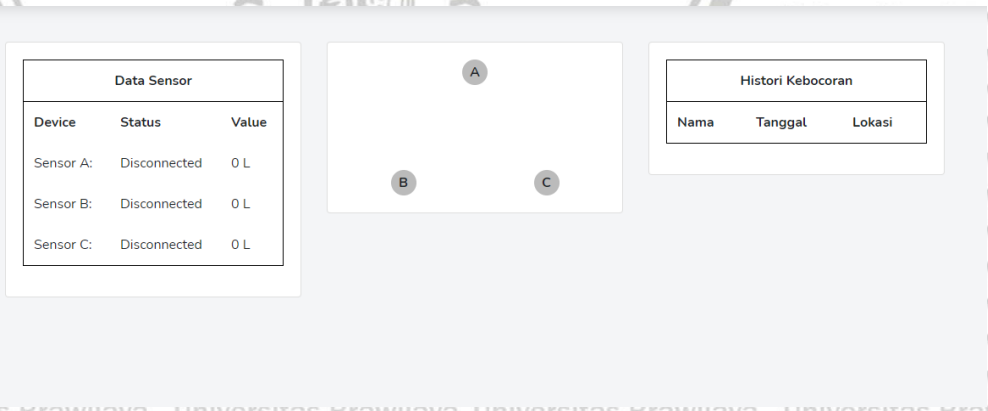
Algoritme 4: *Subscriber* Show Content

```
...

<!DOCTYPE html>
...

<script>
$(function() {

var terhubung = "Connected";
var terputus = "DisConnected";
var disconnColor = '#bbb';
var connColor = '#27AE60';
var warningColor = '#E74C3C';
var timelimit = 60000;

var bocorA = 0.27; // Batas Ambang Kebocoran

var setHistoriA = 0;

function getHistori() {
    $.ajax({
        url: "{{ url('histori') }}",
        type: "GET",
        data: {
            format: 'json',
            _token:"{{ csrf_token() }}"
        },
        sasdataType: 'json',
        success: function(response){
            var table = document.getElementById("tabel-
histori").getElementsByTagName('tbody')[0];
            while (table.childNodes.length) {
                table.removeChild(table.childNodes[0]);
            }
            $.each(response, function(index,item){
                var row = table.insertRow(index);
                var cell1 = row.insertCell(0);
                var cell2 = row.insertCell(1);
                var cell3 = row.insertCell(2);
                var tanggal = new Date(item.updated_at);
                var dd = String(tanggal.getDate()).padStart(2,
'0');
                var mm = String(tanggal.getMonth() +
1).padStart(2, '0'); //January is 0!
                var yyyy = tanggal.getFullYear();
                cell1.innerHTML = item.name
                cell2.innerHTML = dd + '-' + mm + '-' + yyyy;
                cell3.innerHTML = "<a
href='https://www.google.com/maps/place/" + item.latitude + ","
+ item.longitude + "' target='_blank'>" + item.latitude + ', '
+ item.longitude + "</a>";
            });
            console.log(response);
        }
    });
}

```

```
function simpanHistori(name, val, lat, lon) {
    $.ajax({
        url: "{{ url('histori') }}",
        type: "POST",
        data: {
            name:name,
            value:val,
            lat:lat,
            lon:lon,
            _token:"{{ csrf_token() }}"
        },
        success: function(response){
            console.log(response);
        }
    });
}

function getA() {
    var titikName = "Sensor A";
    var conn = document.getElementById("tabel-
status").rows[2].cells;
    var titikStat = document.getElementById("titik-a");
    $.ajax({
        type: 'GET',
        url: "{{ url('api/a') }}",
        data: '_token = <?php echo csrf_token() ?>',
        async: false,
        success: function (response) {
            if (response.error) {
                conn[1].innerHTML = terputus;
                conn[2].innerHTML = "0 L";
                titikStat.style.backgroundColor = disconnColor;
                console.log(response);
            } else {
                conn[2].innerHTML = response.value + " L";
                var start = new
Date(response.updated_at).getTime();
                var end = new Date().getTime();
                var diff = end - start;
                if (diff > timelimit) {
                    conn[1].innerHTML = terputus;
                    titikStat.style.backgroundColor =
disconnColor;
                } else {
                    conn[1].innerHTML = terhubung;
                    if (response.value < bocorA) {
                        titikStat.style.backgroundColor =
warningColor;
                        if (setHistoriA == 0) {
                            simpanHistori(titikName,
response.value, response.lat, response.lon);
                            setHistoriA = 1;
                        }
                    } else {
                        titikStat.style.backgroundColor =
connColor;
                        setHistoriA = 0;
                    }
                }
                console.log(response);
            }
        },
        error: function (xhr, status, error) {
            conn[1].innerHTML = terputus;
            conn[2].innerHTML = "0 L";
```

```
123             titikStat.style.backgroundColor = disconnColor;
124             console.log(error);
125         }
126     });
127 }
128 setInterval(function () {
129     getA();
130     getHistori();
131 }, 5000); // update data di web setiap 5 detik
132 });
133 </script>
```

Pada tabel diatas menjelaskan bagaimana menampilkan data yang telah di *subscribe* dengan diawali pada baris nomor 9 sampai 14 yaitu memberikan nilai terhadap status koneksi terhubung atau tidaknya sebuah sensor serta membuat nilai timelimit sebagai nilai perbandingan dan juga memberikan warna ikon pada setiap node dengan berbagai kondisi. Kondisi pertama yaitu ketika node mati maka berwarna abu-abu, terjadi kebocoran berwarna merah, dan kondisi normal berwarna hijau. Pada baris nomor 16 sampai 18 yaitu memberikan nilai batas ambang untuk indikasi kebocoran, setiap node memiliki batas ambang yang berbeda-beda serta memberikan nilai 0 untuk setHistori. Pada baris nomor 20 sampai 57 adalah fungsi untuk mengambil histori kebocoran pada database yang telah dibuat yang nantinya akan ditampilkan pada halaman web bagian kanan khusus histori kebocoran. Baris nomor 59 sampai 74 yaitu fungsi untuk menyimpan informasi kebocoran kedalam database. Baris nomor 76 sampai nomor 132 adalah fungsi menampilkan data hasil *subscribe* seperti value debit air dan titik koordinat GPS serta merubah status konektivitas node terhubung atau tidak dan merubah warna ikon yang telah didefinisikan sebelumnya lalu terdapat fungsi untuk menyimpan histori kebocoran. Baris nomor 133 sampai 137 adalah sebuah fungsi untuk menjalankan beberapa fungsi di dalamnya secara berkala tergantung waktu yang ditentukan, pada penelitian ini menggunakan waktu setiap 5 detik sekali. Tampilan web yang dihasilkan pada sisi *client* dapat dilihat pada Gambar 3.11.



Gambar 3.19 Hasil Implementasi *Subscriber*

Tampilan web terbagi menjadi 3 sisi yaitu sebelah kiri, tengah dan kanan. Sebelah kiri adalah data sensor yang berisi nama perangkat, status (*connect/disconnect*), dan nilai dari hasil *sensing* sensor YF-S201. Bagian tengah adalah monitoring secara visual bentuk dari jaringan pipa air yang digambarkan

dengan 3 ikon lingkaran yang nantinya dapat berubah warna yaitu abu-abu kondisi sensor mati, merah ketika terjadi kebocoran, dan hijau adalah kondisi normal. Bagian kanan adalah tampilan untuk histori atau menyimpan apabila terjadi kebocoran dengan informasi nama sensor, tanggal kejadian serta lokasi berupa titik koordinat yang telah terhubung dengan google *maps*.

BAB 4 HASIL DAN PEMBAHASAN

4.1Hasil Skenario Pengujian Fungsional

Hasil skenario pengujian fungsional pada penelitian ini untuk mengetahui apakah sistem yang telah diimplementasi telah bekerja sesuai rancangan sebelumnya serta mengetahui nilai dari hasil pengujian yang kemudian hasil pengujian tersebut akan dianalisis dengan tujuan mendapatkan kesimpulan dari masing-masing skenario pengujian yang telah dilakukan

4.1.1 Hasil Skenario Pengujian *Publisher*

4.1.1.1 Tujuan Pengujian Pada Perangkat *Publisher*

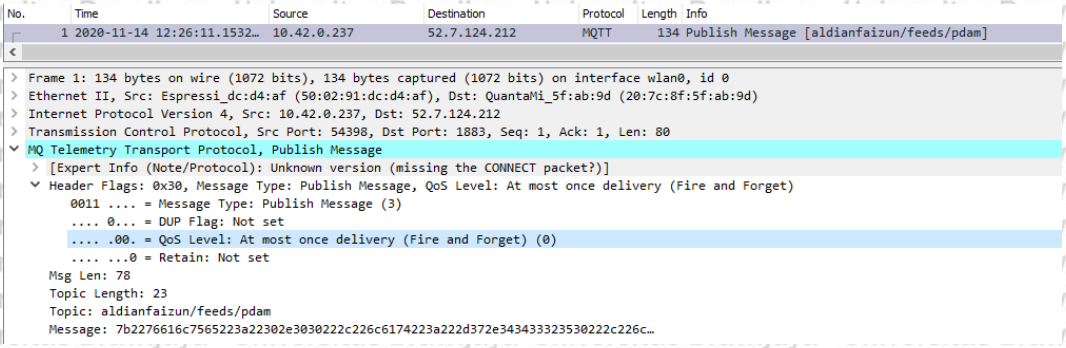
Tujuan dari skenario pengujian ini yaitu mengetahui apakah sistem perangkat *publisher* dapat berfungsi dengan baik dan memastikan data yang berhasil ditangkap oleh sensor dan kemudian dikirim menuju *broker*.

4.1.1.2 Prosedur Pengujian Pada Perangkat *Publisher*

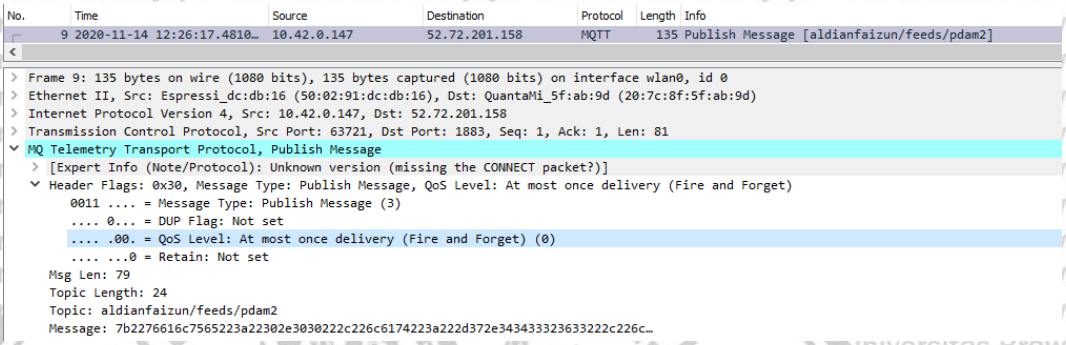
1. Memasang perangkat sensor *flow meter*, modul GPS, dan modul 3G pada perangkat mikrokontroler sesuai dengan skema pin out yang dibantu dengan *project board*.
2. Upload kode program agar perangkat sensor dan modul terhubung dengan mikrokontroler dan mendapatkan data.
3. Melakukan pengujian menghitung debit air dengan tiga kondisi yaitu saat tidak ada aliran air, terdapat aliran air, dan saat terjadi kebocoran.
4. Menjalankan percobaan skenario sebanyak 10 kali untuk meminimalisir terjadinya kesalahan pada saat implementasi.

4.1.1.3 Hasil Pengujian Pada Perangkat *Publisher*

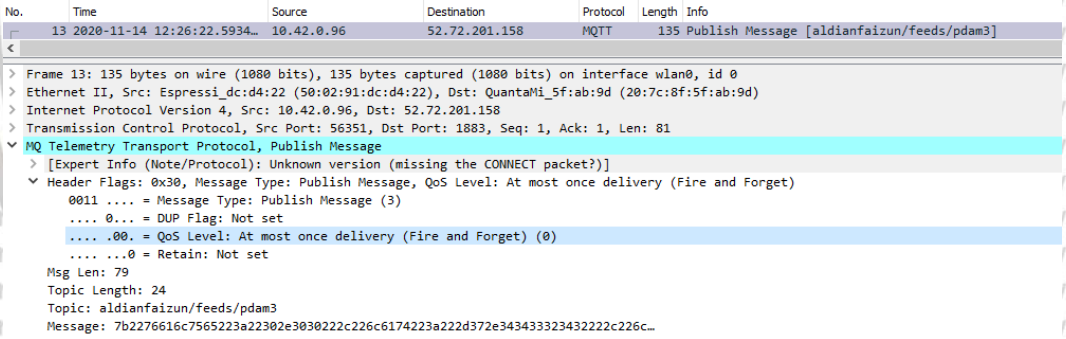
Hasil pengujian perangkat *publisher* dikatakan berhasil apabila telah terkoneksi dengan jaringan 3G, mendapatkan titik koordinat, dan membaca debit air yang mengalir. *Publisher* akan mulai membaca debit air saat kondisi tidak ada aliran air, terdapat aliran air, dan saat terjadi kebocoran. *Publisher* berjumlah 3 node yang berfungsi untuk *collect* data atau *sensing* objek yaitu debit air dan titik koordinat yang nanti data tersebut dikirim atau *publish* menuju *broker* berdasarkan topik masing-masing yaitu pdam (node A), pdam2 (node B), dan pdam3(node C). *Publish* data menggunakan protokol MQTT dengan QoS level 0 atau biasa disebut dengan *fire and forget* seperti pada Gambar 4.1 untu node A, Gambar 4.2 untuk node B, dan Gambar 4.3 untuk node C.



Gambar 4.1 Publish QoS Level 0 Node A



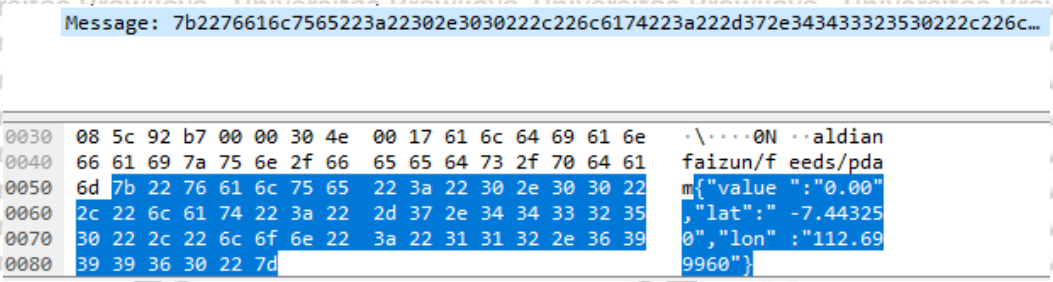
Gambar 4.2 Publish QoS Level 0 Node B



Gambar 4.3 Publish QoS Level 0 Node C

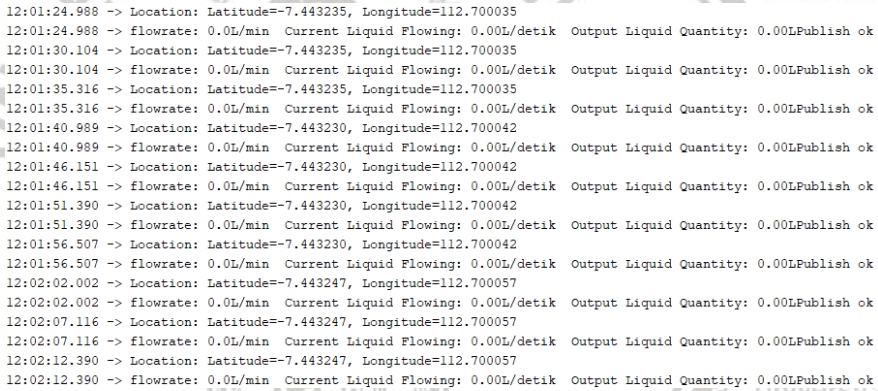
Informasi paket pada *publisher* dapat dilihat menggunakan aplikasi *wireshark* untuk dapat dilihat detail isi setiap paketnya mulai dari alamat sumber/tujuan, besar paket, protokol yang digunakan, dan lain-lain. Terlihat bahwa tipe *publish message* QoS yaitu *at most once delivery* atau *fire and forget* lalu dijelaskan lagi detail informasinya yaitu pada header pesan tidak di *set* atau bernilai 0 untuk flag *duplicate* karena QoS level 0 akan terus menerus mengirim pesan tanpa membutuhkan respon dari *client* sehingga pesan yang dikirim akan dilakukan secara terus menerus. *Retain message* atau pesan yang disimpan juga tidak terdapat pada QoS level 0 karena pada level ini pesan langsung dihapus sesaat setelah dikirim sehingga *client* tidak dapat menerima data jika pesan tersebut tidak terkirim dan mendapat pesan yang terbaru apabila pesan sampai tujuan.

Susunan *topic* pada detail informasi pesan MQTT yaitu *username broker*, feed, dan nama topik yang dikirim. Kemudian *message* yaitu isi pesan yang dikirim oleh *publisher* namun pesan dienkripsi dengan cara merubah teks menjadi *hexadecimal* agar tidak dapat dilihat secara jelas oleh pihak yang bertanggung jawab. Isi pesan yang dirubah menjadi *hexadecimal* tersebut dapat dilihat pada bagian bawah detail informasi *wireshark* yaitu sepert pada gambar 4.4 yang berisikan *value* untuk debit air yang dihasilkan oleh *water flow* serta *lat* dan *long* sebagai titik koordinat yang dihasilkan oleh modul GPS.



Gambar 4.4 Pesan Publisher

A. *Publish* Data Saat Tidak Terdapat Aliran Air



Gambar 4.5 Tampilan *Publish* Saat Tidak Terdapat Aliran Air

Pada Gambar 4.1 dapat dilihat bahwa *publisher* dapat membaca data dari sensor air dan modul GPS. Ketika tidak terdapat aliran air pada jaringan pipa maka output pembacaan debit air bernilai 0 L, sedangkan modul GPS berhasil mendapatkan titik koordinatnya yang terkadang mengalami perubahan yang tidak terlalu signifikan. Telah diambil *sampling* data *publish* menuju *broker* sebanyak 10 data dengan status berhasil terkirim yang akan dijelaskan secara rinci pada Tabel 4.1.

Tabel 4.1 Pengujian *Publish* Kondisi Tidak Terdapat Aliran Air

No	Debit Air (L)	Koordinat	Status
1	0.0	-7.443235, 112.700035	Berhasil
2	0.0	-7.443235, 112.700035	Berhasil
3	0.0	-7.443235, 112.700035	Berhasil

4	0.0	-7.443230, 112.700042	Berhasil
5	0.0	-7.443230, 112.700042	Berhasil
6	0.0	-7.443230, 112.700042	Berhasil
7	0.0	-7.443230, 112.700042	Berhasil
8	0.0	-7.443247, 112.700057	Berhasil
9	0.0	-7.443247, 112.700057	Berhasil
10	0.0	-7.443247, 112.700057	Berhasil

B. *Publish* Data Saat Terdapat Aliran Air

```
11:18:51.027 -> Location: Latitude=-7.443244, Longitude=112.699958
11:18:51.027 -> flowrate: 7.8L/min  Current Liquid Flowing: 0.13L/detik  Output Liquid Quantity: 15.48LPublish ok
11:18:56.143 -> Location: Latitude=-7.443244, Longitude=112.699958
11:18:56.143 -> flowrate: 7.4L/min  Current Liquid Flowing: 0.12L/detik  Output Liquid Quantity: 15.61LPublish ok
11:19:01.366 -> Location: Latitude=-7.443244, Longitude=112.699958
11:19:01.366 -> flowrate: 7.5L/min  Current Liquid Flowing: 0.13L/detik  Output Liquid Quantity: 15.73LPublish ok
11:19:06.996 -> Location: Latitude=-7.443222, Longitude=112.699966
11:19:07.044 -> flowrate: 8.4L/min  Current Liquid Flowing: 0.14L/detik  Output Liquid Quantity: 15.87LPublish ok
11:19:12.184 -> Location: Latitude=-7.443222, Longitude=112.699966
11:19:12.184 -> flowrate: 7.4L/min  Current Liquid Flowing: 0.12L/detik  Output Liquid Quantity: 16.00LPublish ok
11:19:17.404 -> Location: Latitude=-7.443222, Longitude=112.699966
11:19:17.404 -> flowrate: 7.6L/min  Current Liquid Flowing: 0.13L/detik  Output Liquid Quantity: 16.13LPublish ok
11:19:22.575 -> Location: Latitude=-7.443222, Longitude=112.699966
11:19:22.575 -> flowrate: 7.4L/min  Current Liquid Flowing: 0.12L/detik  Output Liquid Quantity: 16.25LPublish ok
11:19:28.008 -> Location: Latitude=-7.443200, Longitude=112.699974
11:19:28.008 -> flowrate: 8.1L/min  Current Liquid Flowing: 0.14L/detik  Output Liquid Quantity: 16.39LPublish ok
11:19:33.169 -> Location: Latitude=-7.443200, Longitude=112.699974
11:19:33.169 -> flowrate: 7.4L/min  Current Liquid Flowing: 0.12L/detik  Output Liquid Quantity: 16.51LPublish ok
11:19:38.433 -> Location: Latitude=-7.443200, Longitude=112.699974
11:19:38.433 -> flowrate: 7.6L/min  Current Liquid Flowing: 0.13L/detik  Output Liquid Quantity: 16.64LPublish ok
```

Gambar 4.6 Tampilan *Publish* Saat Terdapat Aliran Air

Pada Gambar 4.2 dapat dilihat bahwa *publisher* dapat membaca data dari sensor air dan modul GPS. Ketika terdapat aliran air pada jaringan pipa maka output pembacaan debit air nilainya dinamis antara 0.12 L hingga 0.14 L, sedangkan modul GPS berhasil mendapatkan titik koordinatnya yang terkadang mengalami perubahan yang tidak terlalu signifikan. Telah diambil *sampling* data *publish* menuju *broker* sebanyak 10 data dengan status berhasil terkirim yang akan dijelaskan secara rinci pada Tabel 4.2.

Tabel 4.2 Pengujian Publish Kondisi Terdapat Aliran Air

No	Debit Air (L)	Koordinat	Status
1	0.13	-7.443244, 112.699958	Berhasil
2	0.12	-7.443244, 112.699958	Berhasil
3	0.13	-7.443244, 112.699958	Berhasil
4	0.14	-7.443222, 112.699966	Berhasil
5	0.12	-7.443222, 112.699966	Berhasil
6	0.13	-7.443222, 112.699966	Berhasil

7	0.12	-7.443222, 112.699966	Berhasil
8	0.14	-7.4433200, 112.699974	Berhasil
9	0.12	-7.4433200, 112.699974	Berhasil
10	0.13	-7.4433200, 112.699974	Berhasil

C. *Publish* Data Saat Terjadi Kebocoran

```
11:42:03.019 -> Location: Latitude=-7.443274, Longitude=112.700096
11:42:03.019 -> flowrate: 5.2L/min  Current Liquid Flowing: 0.09L/detik  Output Liquid Quantity: 44.30LPublish ok
11:42:08.162 -> Location: Latitude=-7.443274, Longitude=112.700096
11:42:08.162 -> flowrate: 4.6L/min  Current Liquid Flowing: 0.08L/detik  Output Liquid Quantity: 44.37LPublish ok
11:42:13.384 -> Location: Latitude=-7.443274, Longitude=112.700096
11:42:13.384 -> flowrate: 4.8L/min  Current Liquid Flowing: 0.08L/detik  Output Liquid Quantity: 44.45LPublish ok
11:42:18.525 -> Location: Latitude=-7.443274, Longitude=112.700096
11:42:18.525 -> flowrate: 4.6L/min  Current Liquid Flowing: 0.08L/detik  Output Liquid Quantity: 44.53LPublish ok
11:42:24.018 -> Location: Latitude=-7.443226, Longitude=112.700103
11:42:24.018 -> flowrate: 5.3L/min  Current Liquid Flowing: 0.09L/detik  Output Liquid Quantity: 44.62LPublish ok
11:42:29.145 -> Location: Latitude=-7.443226, Longitude=112.700103
11:42:29.145 -> flowrate: 4.6L/min  Current Liquid Flowing: 0.08L/detik  Output Liquid Quantity: 44.70LPublish ok
11:42:34.407 -> Location: Latitude=-7.443226, Longitude=112.700103
11:42:34.407 -> flowrate: 4.9L/min  Current Liquid Flowing: 0.08L/detik  Output Liquid Quantity: 44.78LPublish ok
11:42:39.520 -> Location: Latitude=-7.443226, Longitude=112.700103
11:42:39.520 -> flowrate: 4.5L/min  Current Liquid Flowing: 0.08L/detik  Output Liquid Quantity: 44.85LPublish ok
11:42:45.022 -> Location: Latitude=-7.443198, Longitude=112.700065
11:42:45.022 -> flowrate: 5.3L/min  Current Liquid Flowing: 0.09L/detik  Output Liquid Quantity: 44.94LPublish ok
11:42:50.180 -> Location: Latitude=-7.443198, Longitude=112.700065
11:42:50.180 -> flowrate: 4.7L/min  Current Liquid Flowing: 0.08L/detik  Output Liquid Quantity: 45.02LPublish ok
```

Gambar 4.7 Tampilan *Publish* Saat Terjadi Kebocoran

Pada Gambar 4.3 dapat dilihat bahwa *publisher* dapat membaca data dari sensor air dan modul GPS. Ketika terjadi kebocoran pada jaringan pipa maka output pembacaan debit air nilainya dinamis yaitu 0.08 L dan 0.09 L, sedangkan modul GPS berhasil mendapatkan titik koordinatnya yang terkadang mengalami perubahan yang tidak terlalu signifikan. Telah diambil *sampling* data *publish* menuju *broker* sebanyak 10 data dengan status berhasil terkirim yang akan dijelaskan secara rinci pada Tabel 4.3.

Tabel 4.3 Pengujian *Publish* Kondisi Terjadi Kebocoran

No	Debit Air (L)	Koordinat	Status
1	0.09	-7.443274, 112.700096	Berhasil
2	0.08	-7.443274, 112.700096	Berhasil
3	0.08	-7.443274, 112.700096	Berhasil
4	0.08	-7.443274, 112.700096	Berhasil
5	0.09	-7.443226, 112.700103	Berhasil
6	0.08	-7.443226, 112.700103	Berhasil
7	0.08	-7.443226, 112.700103	Berhasil
8	0.08	-7.443226, 112.700103	Berhasil

9	0.09	-7.443198, 112.700065	Berhasil
10	0.08	-7.443198, 112.700065	Berhasil

4.1.1.4 Analisis Hasil Pengujian Pada Perangkat *Publisher*

Dari hasil pengujian *publisher* yang telah dilakukan yaitu pengujian berjalan dengan baik karena sensor dapat menghitung debit air dengan tiga kondisi yaitu ketika tidak ada aliran air, terdapat aliran air, dan saat terjadi kebocoran yang kemudian di *publish* menggunakan protokol *MQTT*. Dari Gambar 4.1, 4.2, 4.3 terlihat bahwa ketika kondisi tidak ada aliran maka bernilai 0.0 L, ketika terdapat aliran air bernilai antara 0.12 L hingga 0.14 L, dan saat bocor maka debit air menurun menjadi kisaran 0.08L dan 0.09L. Debit air tidak stabil karena beberapa faktor antara lain adanya fraksi pipa PVC, belokan pada jaringan pipa, dan tidak rapatnya drat pipa sambungan. Data yang di *publish* dari ketiga kondisi diatas menunjukkan hasil yang bagus karena keberhasilannya yang hampir tidak terjadi kegagalan saat mengirim data menuju *broker*. Hasil menunjukkan bahwa *publisher* dengan ketiga kondisi diatas dapat melakukan *publish* data menuju *broker* dengan status "publish ok" setiap pesan yang dikirim. *publisher* juga mampu membaca nilai debit air dengan akurat sesuai dengan berbagai kondisi air yang mengalir pada jaringan pipa.

4.1.2 Hasil Skenario Pengujian *Broker*

4.1.2.1 Tujuan Pengujian Pada Perangkat *Broker*

Tujuan dari skenario pengujian ini yaitu mengetahui apakah *broker* dapat berfungsi dengan baik dan memastikan data yang di *publish* oleh *publisher* telah diterima berdasarkan topiknya masing-masing.

4.1.2.2 Prosedur Pengujian Pada Perangkat *Broker*

1. Memasang perangkat sensor *flow meter*, modul GPS, dan modul 3G pada perangkat mikrokontroler sesuai dengan skema pin out yang dibantu dengan *project board*
2. Upload kode program agar perangkat sensor dan modul terhubung dengan mikrokontroler dan mendapatkan data
3. *Publisher* melakukan *publish* data ke *broker* menggunakan protokol *MQTT*.
4. Membuka topik pada halaman *dashboard broker* Adafruit.

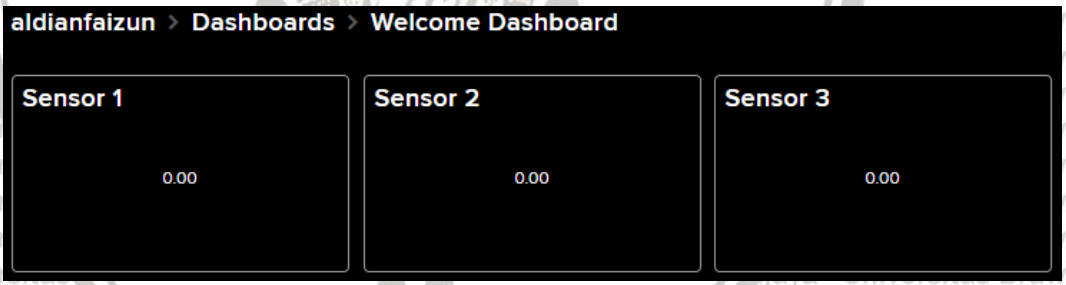
4.1.2.3 Hasil Pengujian Pada *Broker*

Hasil pengujian perangkat *publisher* dikatakan berhasil apabila data yang dikirim oleh *publisher* yaitu debit air dan titik koordinat (*longitude* dan *latitude*) telah berhasil diterima oleh pada *broker* pada topiknya masing-masing. *Broker* akan menerima data debit air saat kondisi tidak ada aliran air, terdapat aliran air, dan saat terjadi kebocoran. *Broker* akan menerima data dari tiga *publisher* dengan topiknya masing-masing dan memiliki jeda waktu sebesar 5 detik karena keterbatasan akun Adafruit sehingga tidak bisa menerima data yang sifatnya banyak dan cepat.

Broker bertugas sebagai penjembatan komunikasi antara *publisher* dengan *subscriber*. *Broker* menunggu pesan yang dikirim oleh *publisher* kemudian data tersebut diolah dan melakukan *manage* pesan sesuai topiknya agar ketika *subscriber* melakukan *request* topik yang diinginkan maka *broker* dapat melayani permintaan tersebut dan dikirim ke *subscriber*. Untuk membuat topik pada sisi *broker* maka pada halaman web io.adafruit.com pilih tab *Feeds* lalu pilih *view all* maka disitu dapat membuat nama topik yang sesuai seperti *publisher* kirim agar data tersebut tersimpan sesuai topiknya. Dapat dilihat pada **Gambar** adalah contoh isi dari topik "pdam" yang terdapat informasi waktu pesan tersampaikan, nilai debit air, dan lokasinya.

Created at	Value	Location
2021/06/10 2:58:18PM	0.00	-7.4433, 112.70001
2021/06/10 2:58:12PM	0.00	-7.4433, 112.70001
2021/06/10 2:58:09PM	0.00	-7.4433, 112.70001
2021/06/10 2:58:02PM	0.00	-7.44329, 112.70003
2021/06/10 2:57:57PM	0.00	-7.44329, 112.70003
2021/06/10 2:57:51PM	0.00	-7.44329, 112.70003
2021/06/10 2:57:46PM	0.00	-7.44329, 112.70003
2021/06/10 2:57:41PM	0.00	-7.443293, 112.70004
2021/06/10 2:57:35PM	0.00	-7.443293, 112.70004
2021/06/10 2:57:30PM	0.00	-7.443293, 112.70004
2021/06/10 2:57:25PM	0.00	-7.44329, 112.70004
2021/06/10 2:57:20PM	0.00	-7.44329, 112.70004

A. *Broker* Saat Tidak Terdapat Aliran Air



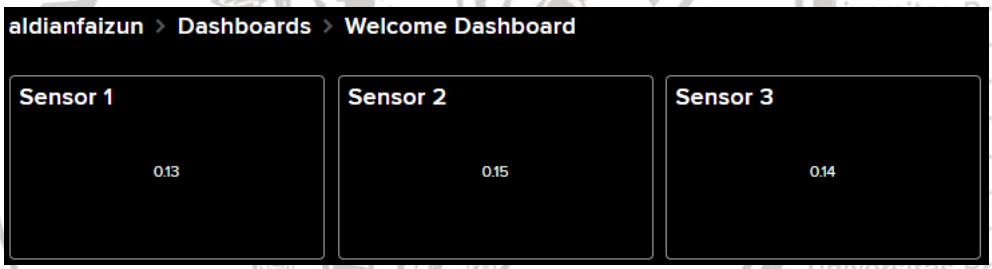
Gambar 4.8 *Dashboard broker* Saat Tidak Terdapat Aliran Air

2021/01/23 12:02:13PM	0.00	-7.443247, 112.70006
2021/01/23 12:02:08PM	0.00	-7.443247, 112.70006
2021/01/23 12:02:02PM	0.00	-7.443247, 112.70006
2021/01/23 12:01:57PM	0.00	-7.44323, 112.70004
2021/01/23 12:01:52PM	0.00	-7.44323, 112.70004
2021/01/23 12:01:47PM	0.00	-7.44323, 112.70004
2021/01/23 12:01:41PM	0.00	-7.44323, 112.70004
2021/01/23 12:01:35PM	0.00	-7.443235, 112.70004
2021/01/23 12:01:30PM	0.00	-7.443235, 112.70004
2021/01/23 12:01:25PM	0.00	-7.443235, 112.70004

Gambar 4.9 Data *broker* Saat Tidak Terdapat Aliran Air

Pada Gambar 4.4 adalah tampilan *broker* saat kondisi tidak terdapat aliran air serta detail data yang diterima dapat dilihat pada Gambar 4.5. Ketika kondisi tidak terdapat aliran air maka data yang ditampilkan pada *dashboard* dan detail datanya untuk debit air bernilai 0.00 sedangkan koordinat nilainya bervariasi.

B. *Broker* Saat Terdapat Aliran Air



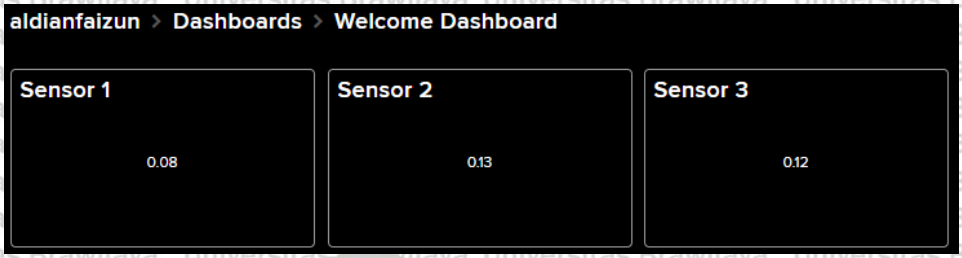
Gambar 4.10 *Dashboard broker* Saat Terdapat Aliran Air

2021/01/23 11:19:39AM	0.13	-7.4432, 112.69997
2021/01/23 11:19:33AM	0.12	-7.4432, 112.69997
2021/01/23 11:19:28AM	0.14	-7.4432, 112.69997
2021/01/23 11:19:23AM	0.12	-7.443223, 112.69997
2021/01/23 11:19:18AM	0.13	-7.443223, 112.69997
2021/01/23 11:19:13AM	0.12	-7.443223, 112.69997
2021/01/23 11:19:07AM	0.14	-7.443223, 112.69997
2021/01/23 11:19:01AM	0.13	-7.443244, 112.69996
2021/01/23 11:18:57AM	0.12	-7.443244, 112.69996
2021/01/23 11:18:51AM	0.13	-7.443244, 112.69996

Gambar 4.11 Data *broker* Saat Terdapat Aliran Air

Pada Gambar 4.6 adalah tampilan *broker* saat kondisi terdapat aliran air serta detail data yang diterima dapat dilihat pada Gambar 4.7. Ketika kondisi terdapat aliran air maka data yang ditampilkan pada *dashboard* dan detail datanya untuk debit air bernilai antara 0.12 sampai 0.14 sedangkan koordinat nilainya bervariasi.

C. *Broker* Saat Terjadi Kebocoran



Gambar 4.12 *Dashboard Broker* Saat Terjadi Kebocoran

2021/01/23 11:42:50AM	0.08	-7.443198, 112.70007
2021/01/23 11:42:45AM	0.09	-7.443198, 112.70007
2021/01/23 11:42:40AM	0.08	-7.443226, 112.7001
2021/01/23 11:42:35AM	0.08	-7.443226, 112.7001
2021/01/23 11:42:29AM	0.08	-7.443226, 112.7001
2021/01/23 11:42:24AM	0.09	-7.443226, 112.7001
2021/01/23 11:42:19AM	0.08	-7.443274, 112.7001
2021/01/23 11:42:14AM	0.08	-7.443274, 112.7001
2021/01/23 11:42:08AM	0.08	-7.443274, 112.7001
2021/01/23 11:42:03AM	0.09	-7.443274, 112.7001

Gambar 4.13 Data *broker* Saat Terjadi Kebocoran

Pada Gambar 4.8 adalah tampilan *dashboard broker* saat kondisi terjadi kebocoran serta detail data yang diterima dapat dilihat pada Gambar 4.9. Ketika kondisi terjadi kebocoran maka data yang ditampilkan pada *dashboard* dan detail datanya untuk debit air bernilai 0.08 dan 0.09 sedangkan koordinat nilainya bervariasi.

4.1.2.4 Analisis Hasil Pengujian Pada *Broker*

Dari hasil pengujian *broker* yaitu berjalan dengan baik karena data yang di *publish* telah diterima oleh *broker* saat kondisi tidak ada aliran air, terdapat aliran air, dan saat terjadi kebocoran. Berikut adalah kumpulan data yang dikirim oleh *publisher* dan diterima oleh *broker*:

Tabel 4.4 Pengujian *Broker* Terhadap Semua Kondisi

Kondisi	Dikirim		Diterima	
	Air (L)	Koordinat	Air (L)	Koordinat
Tidak Ada aliran	0.0	-7.443235, 112.700035	0.0	-7.443235, 112.70004
	0.0	-7.443235, 112.700035	0.0	-7.443235, 112.70004
	0.0	-7.443235, 112.700035	0.0	-7.443235, 112.70004
	0.0	-7.443230, 112.700042	0.0	-7.44323, 112.70004
	0.0	-7.443230, 112.700042	0.0	-7.44323, 112.70004
	0.0	-7.443230, 112.700042	0.0	-7.44323, 112.70004
	0.0	-7.443230, 112.700042	0.0	-7.44323, 112.70004
	0.0	-7.443247, 112.700057	0.0	-7.443247, 112.70006
	0.0	-7.443247, 112.700057	0.0	-7.443247, 112.70006
	0.0	-7.443247, 112.700057	0.0	-7.443247, 112.70006
Terdapat Aliran Air	0.13	-7.443244, 112.699958	0.13	-7.443244, 112.69996
	0.12	-7.443244, 112.699958	0.12	-7.443244, 112.69996
	0.13	-7.443244, 112.699958	0.13	-7.443244, 112.69996
	0.14	-7.443222, 112.699966	0.14	-7.443223, 112.69997
	0.12	-7.443222, 112.699966	0.12	-7.443223, 112.69997
	0.13	-7.443222, 112.699966	0.13	-7.443223, 112.69997
	0.12	-7.443222 ,112.699966	0.12	-7.443223, 112.69997
	0.14	-7.4433200, 112.699974	0.14	-7.4433,112.69997
	0.12	-7.4433200, 112.699974	0.12	-7.4433,112.69997
	0.13	-7.4433200, 112.699974	0.13	-7.4433,112.69997
Terjadi Kebocoran	0.09	-7.443274, 112.700096	0.09	-7.443274, 112.7001
	0.08	-7.443274, 112.700096	0.08	-7.443274, 112.7001
	0.08	-7.443274, 112.700096	0.08	-7.443274, 112.7001
	0.08	-7.443274, 112.700096	0.08	-7.443274, 112.7001
	0.09	-7.443226, 112.700103	0.09	-7.443226, 112.7001
	0.08	-7.443226, 112.700103	0.08	-7.443226, 112.7001
	0.08	-7.443226, 112.700103	0.08	-7.443226, 112.7001
	0.08	-7.443226, 112.700103	0.08	-7.443226, 112.7001

	0.09	-7.443198, 112.700065	0.09	-7.443198, 112.70007
	0.08	-7.443198, 112.700065	0.08	-7.443198, 112.70007

Pada Tabel 4.4 menunjukkan perbandingan data antara data yang dikirim oleh *publisher* dengan data yang berhasil diterima dengan *broker*. Hasilnya adalah data yang dikirim sepenuhnya terkirim dan diterima oleh *broker* namun terdapat perbedaan pada nilai koordinatnya. Pada *broker*, nilai koordinat terjadi pembulatan angka pada akhiran nilainya, hal itu terjadi secara otomatis oleh sistem milik Adafruit. Data pada tabel diatas menunjukkan konsistensi data sehingga data tersebut dapat dinyatakan valid.

4.1.3 Hasil Pengujian *Subscriber*

4.1.3.1 Tujuan Pengujian Pada *Subscriber*

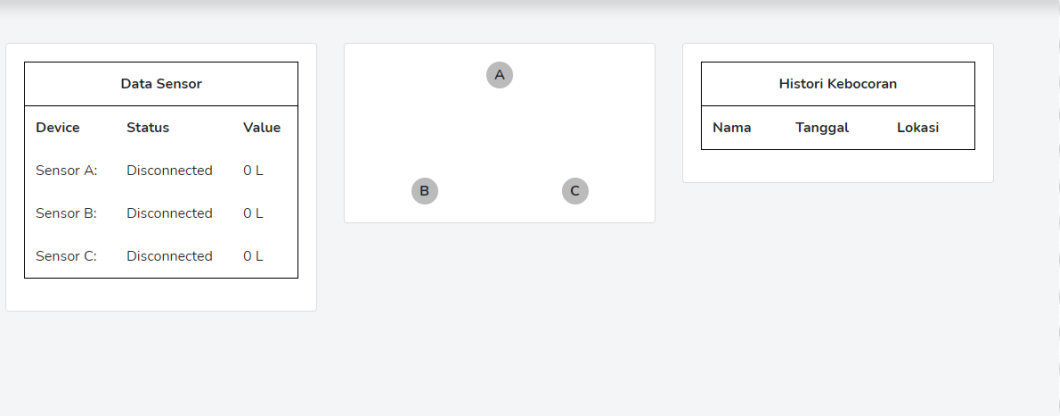
Tujuan dari skenario pengujian ini yaitu mengetahui apakah sistem pada *subscriber* dapat berfungsi dengan baik untuk menerima data dari *broker* lalu menampilkannya pada halaman web. Halaman web memiliki fungsi untuk monitoring debit air dan deteksi kebocoran. Pada *subscriber* terdapat nilai batas ambang sebesar 30% dari kondisi normal untuk menentukan kondisi normal atau terjadi kebocoran pada jaringan pipa. Ketika kondisi normal maka warna ikon pada halaman monitoring berwarna hijau, jika terjadi kebocoran berwarna merah, dan warna abu-abu saat node *publisher* tidak menyala atau mati. Saat terjadi kebocoran maka dapat menampilkan histori kebocoran dan memberikan informasi titik koordinatnya. Serta dilakukan pengujian peforma utilitas yaitu CPU *usage* dan RAM *usage*.

4.1.3.2 Prosedur Pengujian Pada *Subscriber*

1. *Subscribe* dan menampilkan data pada halaman web.
2. Melakukan pengujian monitoring debit air dan deteksi kebocoran saat kondisi node tidak menyala, tidak dan terdapat aliran air serta jika terjadi kebocoran pada jaringan pipa.
3. Melakukan percobaan sebanyak 10 kali untuk melihat tingkat efektifitasnya.
4. Selanjutnya pengujian peforma CPU dan RAM menggunakan aplikasi JMeter.

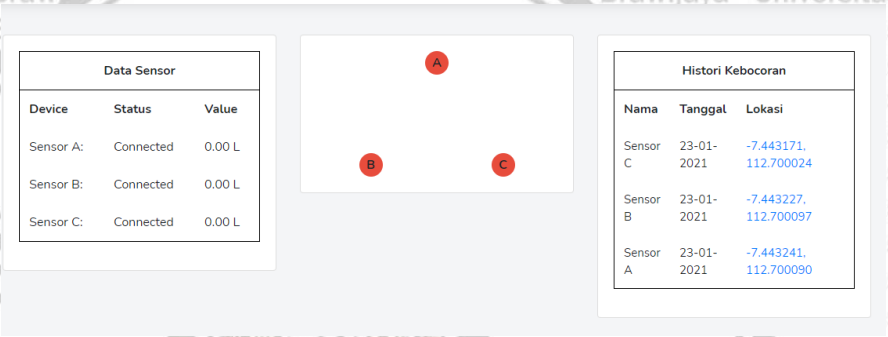
4.1.3.3 Hasil Pengujian Pada *Subscriber*

Hasil pengujian *subscriber* dikatakan berhasil apabila data yang dikirim oleh *broker* diterima oleh *subscriber* yang kemudian ditampilkan secara visual melalui web. Pada *subscriber* akan menampilkan data untuk kebutuhan monitoring debit air dan deteksi kebocoran pada jaringan pipa. Pada Gambar 4.14 adalah tampilan *subscriber*. Ketika belum terjadinya pertukaran data maka tampilan pada web untuk node *publisher* statusnya *disConnected* dengan *value* 0 L, tidak ada histori kebocoran dan ikon node *publisher* warnanya abu-abu pada gambar monitoring jaringan pipa. Berikut adalah beberapa hasil pengujian *subscriber*. Untuk menentukan terjadi kebocoran atau tidak maka pada sisi *subscriber* diberikan batas ambang yang nilainya kurang lebih 30% dari nilai debit air saat kondisi normal.



Gambar 4.14 Kondisi Awal Tampilan Web *Subscriber*

A. *Subscriber* Saat Tidak Terdapat Aliran Air



Gambar 4.15 Tampilan Web Kondisi Tidak Terdapat Aliran Air

Pada Gambar 4.15 adalah tampilan *subscriber* ketika tidak terdapat aliran pada jaringan pipa. Node *publisher* menunjukkan bahwa kondisinya telah aktif dan sedang melakukan *publish* data debit air dan titik koordinat. Warna ikon monitoring node *publisher* berubah menjadi warna merah yang awalnya berwarna abu-abu. Terlihat juga pada histori kebocoran menampilkan nama, tanggal, dan titik koordinatnya itu dikarenakan tidak terdapat aliran air atau 0.0 L dan dibawah nilai batas ambang sehingga pada sistem terindikasi terjadinya kebocoran.

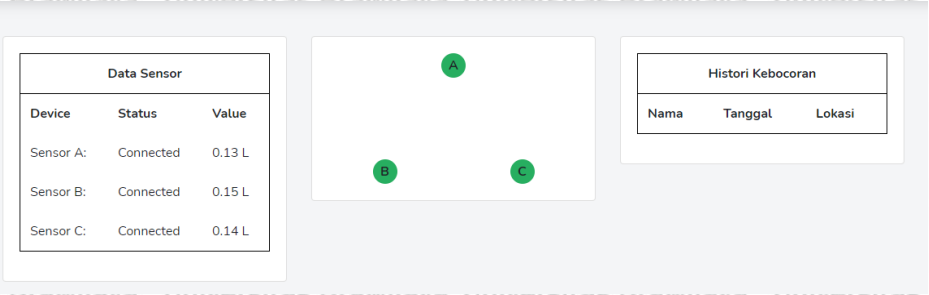
Tabel 4.5 Deteksi Kebocoran Kondisi Tidak Terdapat Aliran Air

Nama	Status	Debit Air (L)	Warna	Keterangan
Sensor A	*Connected*	0.00	Merah	Berhasil
Sensor B	*Connected*	0.00	Merah	Berhasil
Sensor C	*Connected*	0.00	Merah	Berhasil

Pada Tabel 4.5 menunjukkan tingkat keberhasilan *subscriber* dalam menampilkan data pada saat kondisi tidak terdapat aliran air pada jaringan pipa. Tiga node *publisher* telah berstatus *Connected,* debit air 0.00 L, monitoring ikon

berwarna merah sehingga proses transmisi data dari *publisher* yang dikirim menuju *broker* lalu ditampilkan pada *subscriber* dinyatakan berhasil.

B. *Subscriber* Saat Terdapat Aliran Air



Gambar 4.16 Tampilan Web Kondisi Terdapat Aliran Air

Pada Gambar 4.16 adalah tampilan *subscriber* ketika terdapat aliran pada jaringan pipa. Debit air pada data sensor telah berubah karena node *publisher* mengirimkan debit air masing-masing sehingga warna ikon monitoringnya berubah menjadi warna hijau. Nilai debit air sifatnya dinamis bergantung pada data yang dikirim oleh *publisher*. Warna hijau menunjukkan bahwa debit air yang mengalir dalam kondisi normal atau tidak terindikasi adanya kebocoran sehingga pada histori kebocoran tidak menampilkan informasi.

Tabel 4.6 Deteksi Kebocoran Kondisi Terdapat Aliran Air

Nama	Status	Debit Air (L)	Warna	Keterangan
Sensor A	*Connected*	0.13	Hijau	Berhasil
Sensor B	*Connected*	0.15	Hijau	Berhasil
Sensor C	*Connected*	0.14	Hijau	Berhasil

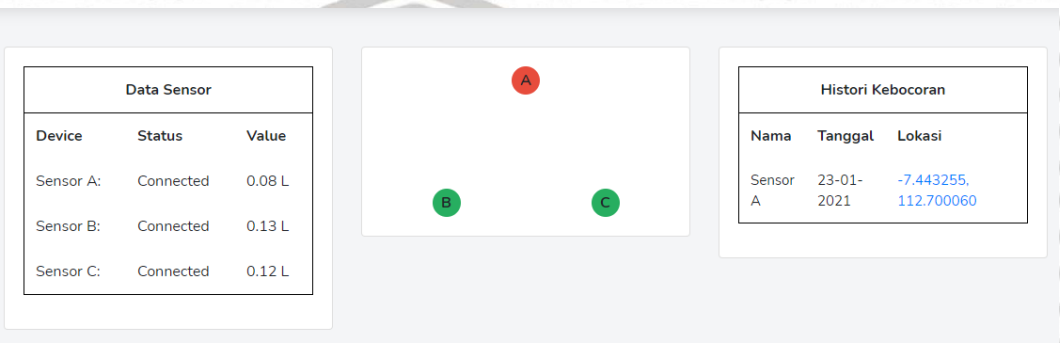
Pada Tabel 4.6 menunjukkan tingkat keberhasilan *subscriber* dalam menampilkan data pada saat kondisi terdapat aliran air pada jaringan pipa. Tiga node *publisher* telah berstatus *Connected* dengan nilai debitnya masing-masing, dan monitoring ikon berwarna hijau sehingga proses transmisi data dari *publisher* yang dikirim menuju *broker* lalu ditampilkan pada *subscriber* dinyatakan berhasil.

C. *Subscriber* Saat Terjadi Kebocoran

Pengujian *subscriber* saat terjadi kebocoran yaitu dilakukan dengan cara membuka tuas kran air hingga lebar sehingga membuat debit air yang mulanya normal atau stabil menjadi berkurang. Pada jaringan pipa terdapat 3 titik kebocoran yang letaknya sebelum node *publisher* ditempatkan. Pengujian titik kebocoran dilakukan satu per satu untuk menguji apakah sistem dapat mendeteksi kebocoran dan berjalan dengan baik. Pada *subscriber* terdapat nilai batas ambang sebagai referensi terjadi kebocoran atau tidak. Nilai batas ambang untuk sensor A yaitu 0.09 L sedangkan untuk sensor B dan sensor C sebesar 0.10 L.

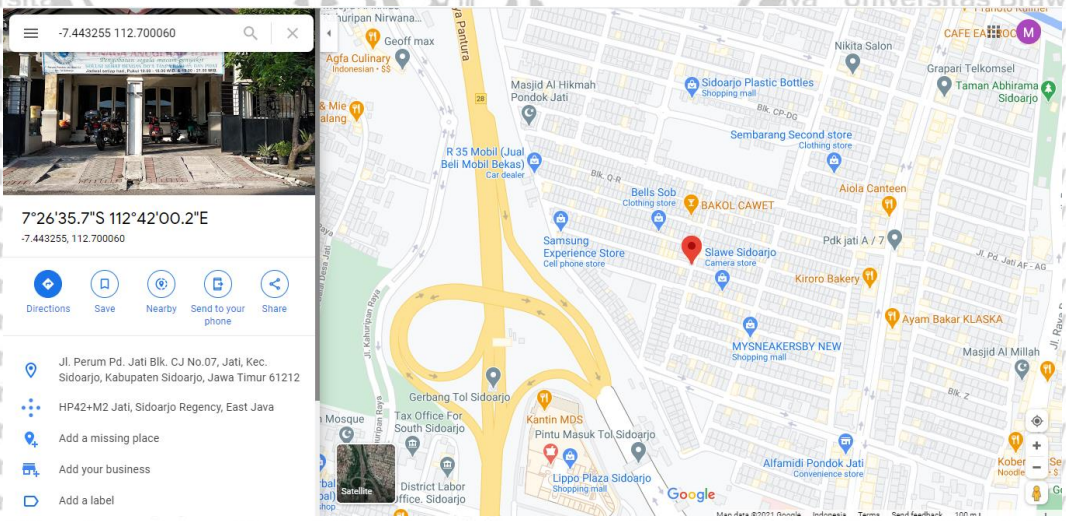
1. Titik Kebocoran 1

Titik kebocoran 1 diletakkan sebelum sensor A atau *publisher* A ditempatkan. Awal mula nilai debit air sensor A bernilai 0.13 L namun setelah tuas kran air dibuka lebar maka terjadi berkurangnya debit air yang mengalir pada jaringan pipa sehingga nilai debit air sensor A menjadi 0.08 L dan warna ikon sensor A menjadi merah. Dikarenakan terindikasi kebocoran maka tabel histori kebocoran tampil informasi kebocoran yang terjadi pada sensor A, tanggal kejadian, dan lokasinya yang berupa titik koordinat. Skenario pengujian ini juga mempengaruhi nilai debit air sensor B dan C yang berkurang menjadi 24% dari nilai semula namun masih dalam keadan normal karena debit airnya belum menyentuh nilai batas ambangnya. Untuk lebih jelas maka dapat dilihat pada Gambar 4.17 untuk hasil skenario kebocoran titik 1.



Gambar 4.17 Tampilan Web Kondisi Terjadi Kebocoran Skenario 1

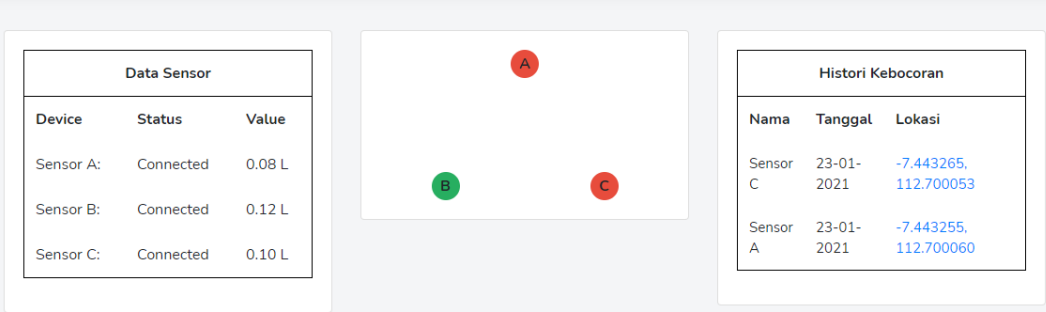
Saat terjadi kebocoran maka menampilkan informasi yang berupa nama, tanggal, dan lokasi. Pada lokasi menampilkan titik koordinat *longitude* dan *latitude* milik sensor A yang dapat diakses melalui google *maps* dengan cara klik lokasinya maka akan menghasilkan gambar peta sesuai titik koordinatnya yang dapat dilihat pada Gambar 4.18.



Gambar 4.18 Titik Koordinat Kebocoran Skenario 1

2. Titik Kebocoran 2

Setelah titik kebocoran 1 dibuka maka selanjutnya titik kebocoran 2 dibuka juga sehingga pada skenario in terdapat dua titik kebocoran. Titik kebocoran 2 diletakkan sebelum sensor C atau *publisher* C ditempatkan. Awal mula nilai debit air sensor C bernilai 0.14 L namun setelah tuas kran air dibuka lebar maka terjadi berkurangnya debit air yang mengalir pada jaringan pipa sehingga nilai debit air sensor C menjadi 0.10 L dan warna ikon sensor C menjadi merah. Skenario pengujian ini juga mempengaruhi nilai debit air sensor B berkurang menjadi 0.12 L, sensor C berkurang menjadi 0.10 L, sedangkan sensor A tidak mengalami penurunan debit air. Sensor A dan C terindikasi terjadinya kebocoran sehingga masuk pada informasi histori kebocoran. Untuk lebih jelas maka dapat dilihat pada Gambar 4.19 untuk hasil skenario kebocoran titik 2.



Data Sensor

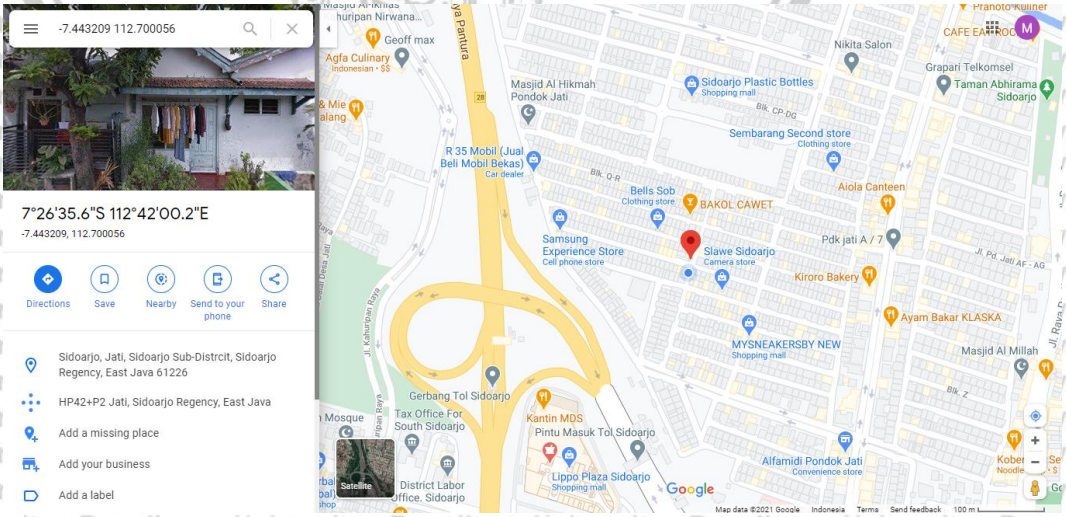
Device	Status	Value
Sensor A:	Connected	0.08 L
Sensor B:	Connected	0.12 L
Sensor C:	Connected	0.10 L

Histori Kebocoran

Nama	Tanggal	Lokasi
Sensor C	23-01-2021	-7.443265, 112.700053
Sensor A	23-01-2021	-7.443255, 112.700060

Gambar 4.19 Tampilan Web Kondisi Terjadi Kebocoran Skenario 2

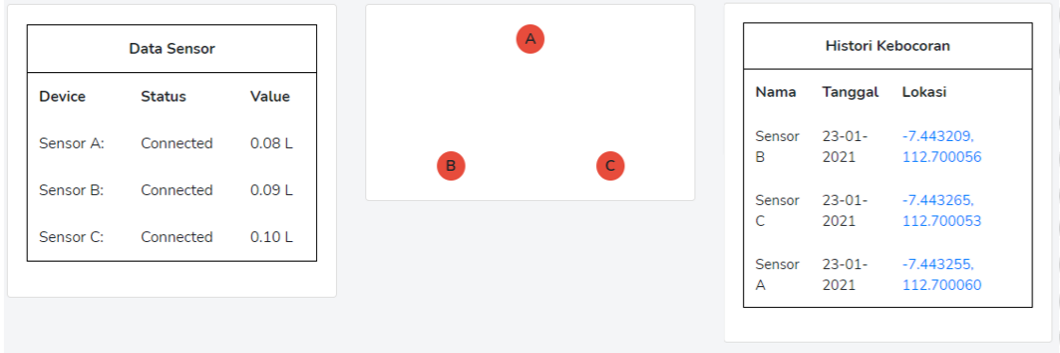
Saat terjadi kebocoran maka menampilkan informasi yang berupa nama, tanggal, dan lokasi. Pada lokasi menampilkan titik koordinat *longitude* dan *latitude* milik sensor C yang dapat diakses melalui google *maps* dengan cara klik lokasinya maka akan menghasilkan gambar peta sesuai titik koordinatnya yang dapat dilihat pada Gambar 4.20.



Gambar 4.20 Titik Koordinat Kebocoran Skenario 2

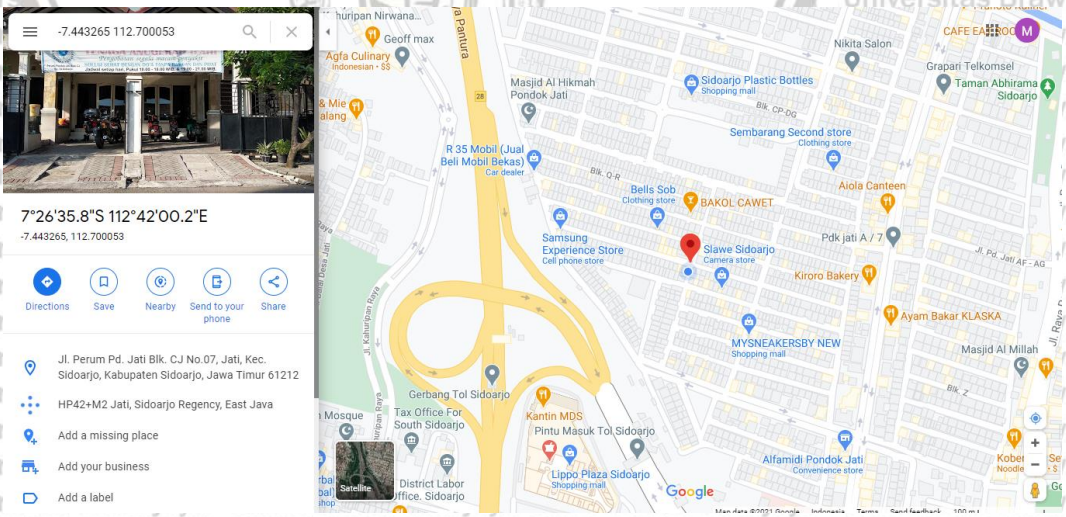
3. Titik Kebocoran 3

Setelah titik kebocoran 1 dan 2 dibuka maka selanjutnya titik kebocoran 3 dibuka juga sehingga pada skenario ini semua titik kebocoran terbuka semua. Titik kebocoran 3 diletakkan sebelum sensor B atau *publisher* B ditempatkan. Awal mula nilai debit air sensor B bernilai 0.15 L namun setelah tuas kran air dibuka lebar maka terjadi berkurangnya debit air yang mengalir pada jaringan pipa sehingga nilai debit air sensor B menjadi 0.09 L dan warna ikon sensor B menjadi merah. Skenario pengujian ini tidak mempengaruhi nilai debit air pada sensor C dan sensor A. Skenario ini menyebabkan sensor A, B dan C terindikasi terjadinya kebocoran sehingga masuk pada informasi histori kebocoran. Untuk lebih jelas maka dapat dilihat pada Gambar 4.21 untuk hasil skenario kebocoran titik 2.



Gambar 4.21 Tampilan Web Kondisi Terjadi Kebocoran Skenario 3

Saat terjadi kebocoran maka menampilkan informasi yang berupa nama, tanggal, dan lokasi. Pada lokasi menampilkan titik koordinat *longitude* dan *latitude* milik sensor B yang dapat diakses melalui google *maps* dengan cara klik lokasinya maka akan menghasilkan gambar peta sesuai titik koordinatnya yang dapat dilihat pada Gambar 4.22.



Gambar 4.22 Titik Koordinat Kebocoran Skenario 2

Berikut juga dilakukan pengujian lama waktu sistem untuk dapat mendeteksi kebocoran yang dilakukan sebanyak 10 kali. Pengujian dilakukan dengan melihat perubahan warna ikon monitoring yang semulanya normal berwarna hijau lalu berubah menjadi merah karena terjadi kebocoran. Tujuan pengujian ini untuk mengetahui seberapa cepat respon waktu yang dibutuhkan dalam satuan detik. Perhitungan ini dilakukan secara manual yaitu melihat perubahan warna dengan dibantu *stopwatch* untuk mengetahui lama waktu yang diberikan.

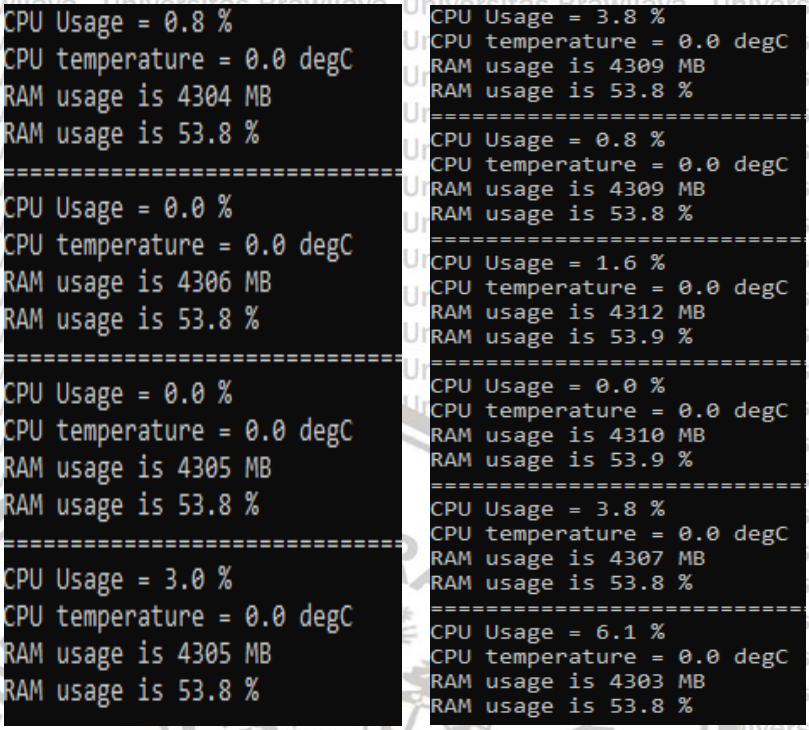
Tabel 4.7 Respon Waktu Deteksi Kebocoran

Percobaan Ke -	Lama Waktu (detik)
1	5
2	5
3	6
4	7
5	5
6	6
7	7
8	8
9	4
10	5

Berdasarkan tabel 4.7 pengujian diatas maka respon waktu tercepat untuk mendeteksi kebocoran yaitu 4 detik dan waktu terlamanya membutuhkan waktu 8 detik. Jika dilakukan rata-rata maka sistem untuk dapat mendeteksi kebocoran sebesar 5.8 detik.

Pengujian utilitas yaitu untuk melihat peforma CPU dan RAM pada *subscriber* pada saat melakukan *subscribe* dengan berbagai kondisi yaitu 1 node, 2 node, dan 3 node. Pengujian ini dilakukan untuk melihat perbedaan performa dengan 3 kondisi.

- Kondisi *subscribe* 1 Node



Gambar 4.23 Monitoring Peforma Subscribe 1 Node

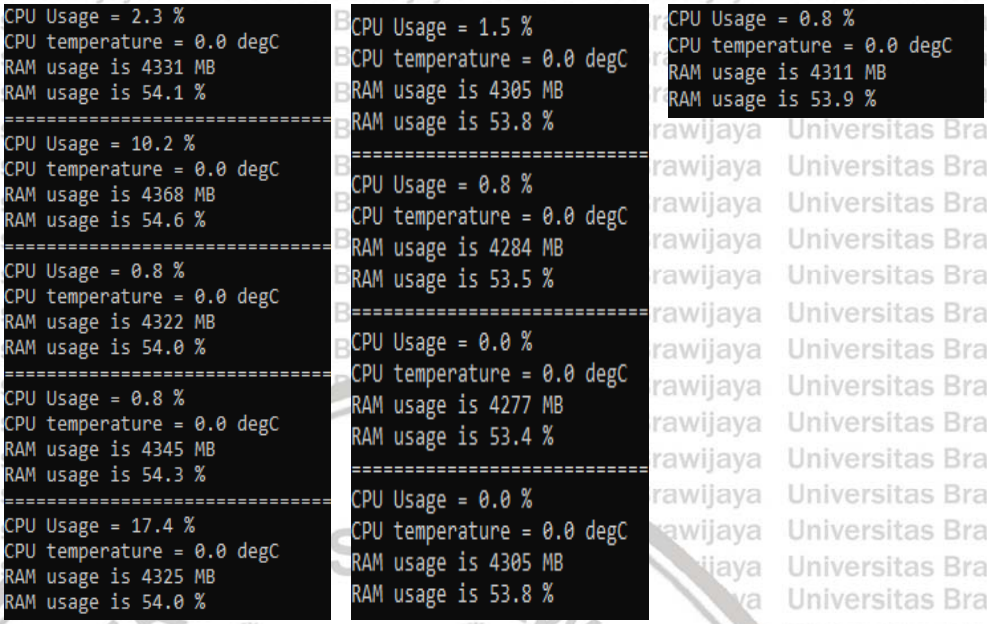
Pada Gambar 4.23 adalah kondisi peforma *subscriber* pada saat melakukan proses *subscribe* 1 node. Hal ini dilakukan untuk mengetahui berapa prosentase dari CPU dan RAM yang terpakai. Nilai terendah dari CPU *usage* yaitu 0.0%, sedangkan yang tertinggi yaitu 6.1% namun secara keseluruhan nilainya didominasi oleh 0.0% dan 3.8%. RAM *usage* dengan pemakaian terendah yaitu bernilai 53.8% dan untuk pemakaian tertinggi sebesar 53.9%.

Tabel 4.8 Rata-rata Peforma Subscribe 1 Node

Peforma	Rata-rata (%)
CPU Usage	1.99
RAM Usage	53.82

Hasil monitoring peforma akan dihitung nilai rata-ratanya yang dapat dilihat pada Tabel 4.8. Terlihat bahwa nilai rata-rata peforma dari CPU *usage* adalah 1.99% dan RAM *usage* sebesar 53.82%.

- Kondisi *subscribe* 2 Node



Gambar 4.24 Monitoring Peforma Subscribe 2 Node

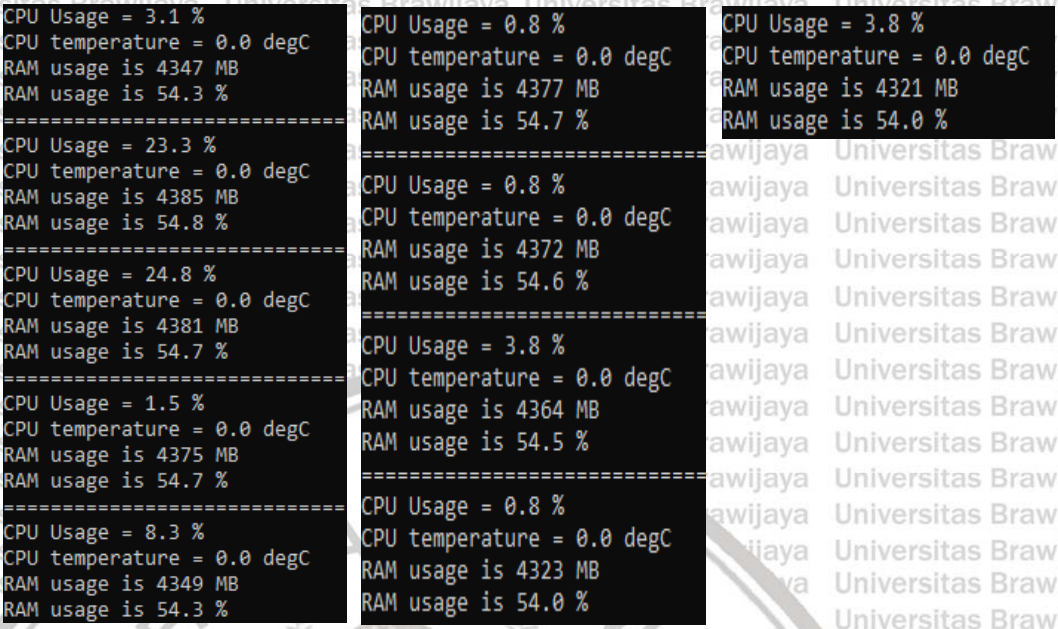
Pada Gambar 4.24 adalah kondisi peforma *subscriber* pada saat melakukan proses *subscribe* 2 node. Hal ini dilakukan untuk mengetahui berapa prosentase dari CPU dan RAM yang terpakai. Nilai terendah dari CPU *usage* yaitu 0.0%, sedangkan yang tertinggi yaitu 17.4% namun secara keseluruhan nilainya didominasi oleh 0.0% dan 0.8%. RAM *usage* dengan pemakaian terendah yaitu bernilai 53.4% dan untuk pemakaian tertinggi sebesar 54.6%.

Tabel 4.9 Rata-rata Peforma Subscribe 2 Node

Peforma	Rata-rata (%)
CPU Usage	3.46
RAM Usage	53.94

Hasil monitoring peforma akan dihitung nilai rata-ratanya yang dapat dilihat pada Tabel 4.9. Terlihat bahwa nilai rata-rata peforma dari CPU *usage* adalah 3.46% dan RAM *usage* sebesar 53.94%. CPU *usage* pada kondisi *subscribe* 2 node ini mengalami peningkatan sebanyak 1.47% jika dibandingkan dengan data pada Tabel 4.8.

- Kondisi *subscribe* 3 Node



Gambar 4.25 Monitoring Peforma Subscribe 3 Node

Pada Gambar 4.25 adalah kondisi peforma *subscriber* pada saat melakukan proses *subscribe* 3 node. Hal ini dilakukan untuk mengetahui berapa prosentase dari CPU dan RAM yang terpakai. Nilai terendah dari CPU *usage* yaitu 0.8%, sedangkan yang tertinggi yaitu 24.8% namun secara keseluruhan nilainya didominasi oleh 0.8%. RAM *usage* dengan pemakaian terendah yaitu bernilai 54.0% dan untuk pemakaian tertinggi sebesar 54.7%.

Tabel 4.10 Rata-rata Peforma Subscribe 3 Node

Peforma	Rata-rata (%)
CPU Usage	7.1
RAM Usage	54.46

Hasil monitoring peforma akan dihitung nilai rata-ratanya yang dapat dilihat pada Tabel 4.10. Terlihat bahwa nilai rata-rata peforma dari CPU *usage* adalah 7.1% dan RAM *usage* sebesar 54.46%. Rata-rata penggunakan CPU *usage* mengalami peningkatan sebanyak 3.64% hal ini dikarenakan bertambahnya proses *subscribe* sejumlah 3 node.

4.1.3.4 Analisis Hasil Pengujian Pada *Subscriber*

Hasil pengujian *subscriber* yaitu berjalan dengan baik karena dapat menampilkan data hasil *subscribe* dari *broker* dengan kondisi node tidak menyala, terdapat aliran air dan tidak terdapat aliran serta jika terjadi kebocoran pada jaringan pipa.

Tabel 4.11 Hasil Pengujian *Subscriber* Terhadap Semua Kondisi

Nama	Kondisi	Nilai		Warna	Status
		Air (L)	Batas Ambang (L)		
Sensor A	Tidak Ada Air	0.0	0.09	Merah	Berhasil
	Ada Air	0.13	0.09	Hijau	Berhasil
	Kebocoran 1	0.08	0.09	Merah	Berhasil
	Kebocoran 2	0.08	0.09	Merah	Berhasil
	Kebocoran 3	0.08	0.09	Merah	Berhasil
Sensor B	Tidak Ada Air	0.0	0.10	Merah	Berhasil
	Ada Air	0.15	0.10	Hijau	Berhasil
	Kebocoran 1	0.13	0.10	Hijau	Berhasil
	Kebocoran 2	0.12	0.10	Hijau	Berhasil
	Kebocoran 3	0.09	0.10	Merah	Berhasil
Sensor C	Tidak Ada Air	0.0	0.10	Merah	Berhasil
	Ada Air	0.14	0.10	Hijau	Berhasil
	Kebocoran 1	0.12	0.10	Hijau	Berhasil
	Kebocoran 2	0.10	0.10	Merah	Berhasil
	Kebocoran 3	0.10	0.10	Merah	Berhasil

Pada Tabel 4.11 memperlihatkan tingkat keberhasilan *subscriber* dalam menampilkan data yang telah di *subscribe* kepada *broker* sebelumnya. Nilai batas ambang memiliki perbedaan karena faktor output kran air yang berbeda pula sehingga batas ambang kebocoran untuk sensor A bernilai 0.09 L, sedangkan untuk sensor B dan sensor C sebesar 0.10 L. Hasilnya *subscriber* dapat menampilkan data dengan berbagai kondisi yaitu saat *publisher* mati, tidak terdapat air, terdapat aliran air serta jika terjadi kebocoran yang memiliki tingkat keberhasilan yang baik. *Subscriber* mampu memberikan notifikasi saat debit air telah mencapai batas ambang yaitu dengan merubah warnanya menjadi merah dan dimasukkan ke dalam informasi histori kebocoran, sedangkan apabila nilai debit air diatas nilai batas ambang maka berwarna hijau. Terlihat bahwa saat terjadi kebocoran, nilai debit air tidak stabil dikarenakan beberapa faktor yaitu sambungan drat pada pipa yang tidak rapat, belokan pada jaringan pipa, dan terdapat fraksi yang dimiliki oleh pipa PVC. Tingkat keberhasilan monitoring debit air sebesar 100% karena mampu menampilkan nilai debit air sesuai dengan warna ikon masing-masing node. Keberhasilan sistem dalam mendeteksi kebocoran yaitu

100% karena mampu mengetahui bahwa debit air yang normal telah bernilai sama atau lebih kecil dari nilai batas ambang sehingga merubah warna ikon menjadi merah.

4.2 Pengujian Kinerja Pada Protokol *MQTT*

4.2.1 Tujuan Pengujian Kinerja Pada Protokol *MQTT*

Tujuan dari pengujian kinerja protokol *MQTT* yaitu mengetahui tingkat efektivitas protokol *MQTT* pada sistem monitoring debit air dan deteksi kebocoran. Beberapa parameter yang dilakukan yaitu *delay* dan *packet loss*.

4.2.2 Prosedur Pengujian Kinerja Pada Protokol *MQTT*

1. Memasang sensor dan modul ke perangkat mikrokontroler.
2. Upload kode program ke mikrokontroler.
3. Jalankan aplikasi coolterm untuk *publish* data dan menampilkan *timestamp*.
4. Melakukan *subscribe* melalui aplikasi *MQTT*.fx
5. Hitung *delay* dengan menghitung selisih waktu penerimaan data dengan waktu pengiriman.
6. Hitung *packet loss* dari jumlah paket yang dikirim dibandingkan dengan paket yang berhasil diterima.

4.2.3 Hasil Pengujian Kinerja Pada Protokol *MQTT*

Hasil pengujian kinerja pada protokol *MQTT* diukur menggunakan dua parameter yaitu *delay* dan *packet loss*. Pengujian *delay* dilakukan untuk mengukur lama waktu yang dibutuhkan oleh suatu paket dari sumber ke tujuan, sedangkan *packet loss* untuk menghitung besarnya paket yang hilang dalam kurun waktu tertentu.

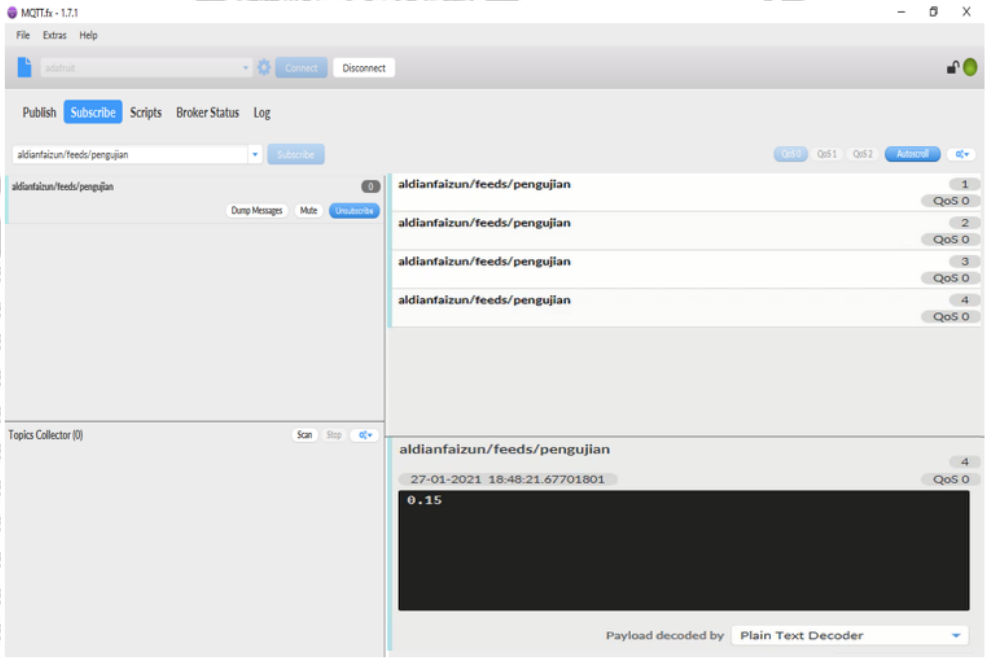
4.2.3.1 Delay

Skenario pengujian *delay* dilakukan sebanyak 15 percobaan paket sehingga dari beberapa data tersebut dihitung waktu yang dibutuhkan oleh paket mulai dari *publisher* hingga sampai ke *subscriber*. Nilai *delay* berasal dari selisih waktu paket diterima dengan waktu paket dikirim. *Delay* yang dihitung pada scenario yaitu waktu Ketika *publisher* mengirim data hingga sampai pada tujuan yaitu *subscriber*. Pada Gambar 4.26 adalah kondisi ketika *publisher* mengirim paket pada waktu tertentu dan terus menerus.

```
18:48:05.577    Location: Latitude=-7.443225, Longitude=112.699966
18:48:05.655    flowrate: 3.7L/min  Current Liquid Flowing: 0.06L/detik  Output Liquid Quantity: 0.06LPublish ok
18:48:10.773    Location: Latitude=-7.443225, Longitude=112.699966
18:48:10.834    flowrate: 8.9L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 0.21LPublish ok
18:48:15.994    Location: Latitude=-7.443225, Longitude=112.699966
18:48:16.104    flowrate: 9.0L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 0.37LPublish ok
18:48:21.154    Location: Latitude=-7.443225, Longitude=112.699966
18:48:21.239    flowrate: 8.9L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 0.51LPublish ok
18:48:26.515    Location: Latitude=-7.443218, Longitude=112.699996
18:48:26.603    flowrate: 9.3L/min  Current Liquid Flowing: 0.16L/detik  Output Liquid Quantity: 0.67LPublish ok
18:48:31.621    Location: Latitude=-7.443218, Longitude=112.699996
18:48:31.708    flowrate: 8.8L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 0.82LPublish ok
18:48:36.840    Location: Latitude=-7.443218, Longitude=112.699996
18:48:36.919    flowrate: 8.9L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 0.97LPublish ok
18:48:42.495    Location: Latitude=-7.443207, Longitude=112.699996
18:48:42.637    flowrate: 9.9L/min  Current Liquid Flowing: 0.17L/detik  Output Liquid Quantity: 1.13LPublish ok
18:48:47.672    Location: Latitude=-7.443207, Longitude=112.699996
18:48:47.798    flowrate: 8.9L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 1.28LPublish ok
18:48:52.951    Location: Latitude=-7.443207, Longitude=112.699996
18:48:53.028    flowrate: 9.1L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 1.43LPublish ok
18:48:58.078    Location: Latitude=-7.443207, Longitude=112.699996
18:48:58.141    flowrate: 8.7L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 1.58LPublish ok
18:49:03.502    Location: Latitude=-7.443214, Longitude=112.699989
18:49:03.581    flowrate: 9.4L/min  Current Liquid Flowing: 0.16L/detik  Output Liquid Quantity: 1.74LPublish ok
18:49:08.625    Location: Latitude=-7.443214, Longitude=112.699989
18:49:08.734    flowrate: 8.7L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 1.88LPublish ok
18:49:13.895    Location: Latitude=-7.443214, Longitude=112.699989
18:49:13.974    flowrate: 9.1L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 2.04LPublish ok
18:49:19.021    Location: Latitude=-7.443214, Longitude=112.699989
18:49:19.112    flowrate: 8.8L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 2.18LPublish ok
```

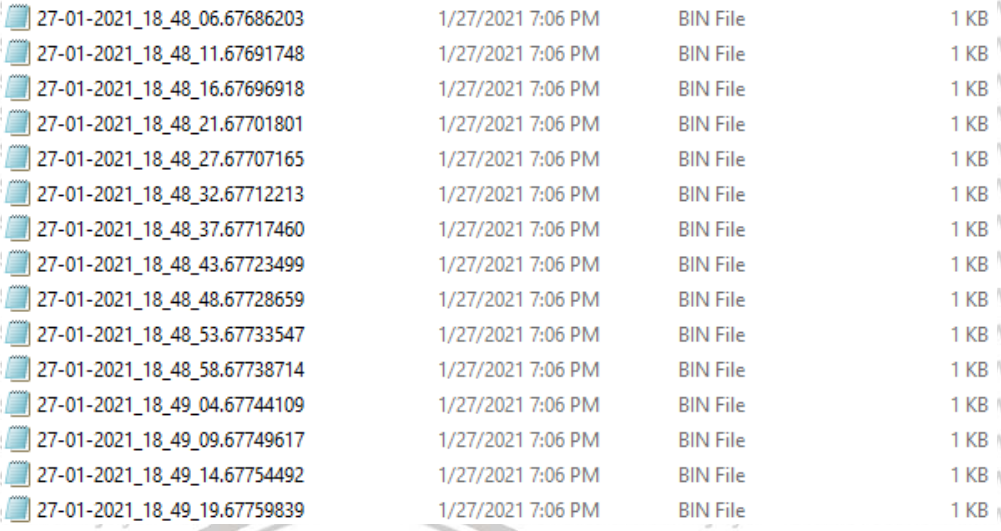
Gambar 4.26 Waktu Awal *Publisher* Mengirim Paket

Gambar diatas adalah aktivitas *publisher* saat mengirim paket yang berisikan informasi titik lokasi dengan koordinat *longitude* dan *latitude* serta debit air yang mengalir. Apabila paket berhasil terkirim ke *broker* maka akan tampil informasi "*publish* ok". Setelah paket tiba di *broker* maka *subscriber* melakukan *request* topik menggunakan aplikasi *MQTT*.fx. Pada Gambar 4.27 adalah tampilan aplikasi *MQTT*.fx ketika melakukan *subscribe* dan paket berhasil diterima serta terdapat juga informasi berupa *timestamp* untuk perhitungan *delay* pada scenario ini.



Gambar 4.27 Subscribe Paket Pada *MQTT*.fx

Pada *MQTT*.fx terdapat fitur *dump message* yang berfungsi untuk membuatkan log aktivitas data yang berhasil di *subscribe*. Gambar 4.28 adalah beberapa file log yang berhasil dibuat oleh *MQTT*.fx.



27-01-2021_18_48_06.67686203	1/27/2021 7:06 PM	BIN File	1 KB
27-01-2021_18_48_11.67691748	1/27/2021 7:06 PM	BIN File	1 KB
27-01-2021_18_48_16.67696918	1/27/2021 7:06 PM	BIN File	1 KB
27-01-2021_18_48_21.67701801	1/27/2021 7:06 PM	BIN File	1 KB
27-01-2021_18_48_27.67707165	1/27/2021 7:06 PM	BIN File	1 KB
27-01-2021_18_48_32.67712213	1/27/2021 7:06 PM	BIN File	1 KB
27-01-2021_18_48_37.67717460	1/27/2021 7:06 PM	BIN File	1 KB
27-01-2021_18_48_43.67723499	1/27/2021 7:06 PM	BIN File	1 KB
27-01-2021_18_48_48.67728659	1/27/2021 7:06 PM	BIN File	1 KB
27-01-2021_18_48_53.67733547	1/27/2021 7:06 PM	BIN File	1 KB
27-01-2021_18_48_58.67738714	1/27/2021 7:06 PM	BIN File	1 KB
27-01-2021_18_49_04.67744109	1/27/2021 7:06 PM	BIN File	1 KB
27-01-2021_18_49_09.67749617	1/27/2021 7:06 PM	BIN File	1 KB
27-01-2021_18_49_14.67754492	1/27/2021 7:06 PM	BIN File	1 KB
27-01-2021_18_49_19.67759839	1/27/2021 7:06 PM	BIN File	1 KB

Gambar 4.28 Waktu Kedatangan Paket Pada *Dump* Message *MQTT*.fx

Berdasarkan informasi yang diperoleh untuk menghitung nilai *delay* yaitu waktu pengiriman paket dan waktu paket diterima, maka dari itu dihitung selisih dari kedua waktu tersebut. Pada Tabel 4.12 menjelaskan tentang perhitungan delay pada masing-masing paket.

Tabel 4.12 Hasil Pengujian Nilai *Delay*

No	Waktu Paket Pengiriman	Waktu Paket Diterima	*Delay* (ms)
1	18:48:05.655	18:48:06.677	1022
2	18:48:10.834	18:48:11.677	843
3	18:48:16.104	18:48:16.677	573
4	18:48:21.239	18:48:21.677	438
5	18:48:26.603	18:48:27.677	1074
6	18:48:31.708	18:48:32.677	969
7	18:48:36.919	18:48:37.677	758
8	18:48:42.637	18:48:43.677	1040
9	18:48:47.798	18:48:48.677	879
10	18:48:53.028	18:48:53.677	649
11	18:48:58.141	18:48:58.677	536
12	18:49:03.581	18:49:04.677	1096
13	18:49:08.734	18:49:09.677	943
14	18:49:13.974	18:49:14.678	704
15	18:49:19.112	18:49:19.678	566
Rata-rata Delay			806

Tabel diatas merupakan hasil pengujian *delay* pada penelitian ini dengan menggunakan protokol *MQTT*. Terdapat 15 paket yang dilakukan pengujian *delay* dengan *delay* tertinggi yaitu 1096 ms dan *delay* terendah yaitu 438 ms.

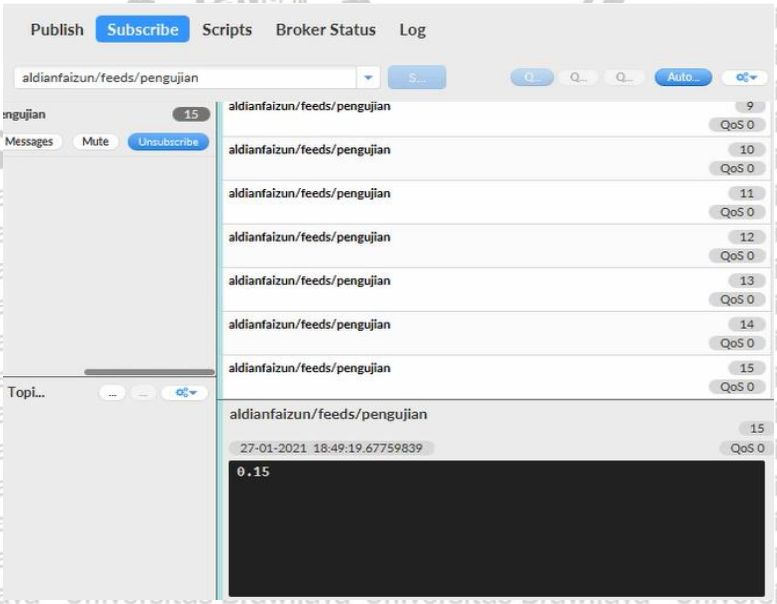
4.2.3.2 Packet Loss

Skenario pengujian *packet loss* dilakukan sebanyak 15 kali percobaan dengan setiap percobaan berjumlah 15 paket, namun untuk sampling data maka hanya 15 paket pertama yang diambil. Dari beberapa data tersebut dihitung berapa paket yang mengalami kegagalan pada saat transmisi paket mulai dari *publisher* hingga menuju *subscriber*. Pada Gambar 4.29 adalah tampilan *publisher* saat mengirimkan paket.

```
18:48:05.577    Location: Latitude=-7.443225, Longitude=112.699966
18:48:05.655    flowrate: 3.7L/min  Current Liquid Flowing: 0.06L/detik  Output Liquid Quantity: 0.06LPublish ok
18:48:10.773    Location: Latitude=-7.443225, Longitude=112.699966
18:48:10.834    flowrate: 8.9L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 0.21LPublish ok
18:48:15.994    Location: Latitude=-7.443225, Longitude=112.699966
18:48:16.104    flowrate: 9.0L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 0.37LPublish ok
18:48:21.154    Location: Latitude=-7.443225, Longitude=112.699966
18:48:21.239    flowrate: 8.9L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 0.51LPublish ok
18:48:26.515    Location: Latitude=-7.443218, Longitude=112.699996
18:48:26.603    flowrate: 9.3L/min  Current Liquid Flowing: 0.16L/detik  Output Liquid Quantity: 0.67LPublish ok
18:48:31.621    Location: Latitude=-7.443218, Longitude=112.699996
18:48:31.708    flowrate: 8.8L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 0.82LPublish ok
18:48:36.840    Location: Latitude=-7.443218, Longitude=112.699996
18:48:36.919    flowrate: 8.9L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 0.97LPublish ok
18:48:42.495    Location: Latitude=-7.443207, Longitude=112.699996
18:48:42.637    flowrate: 9.9L/min  Current Liquid Flowing: 0.17L/detik  Output Liquid Quantity: 1.13LPublish ok
18:48:47.672    Location: Latitude=-7.443207, Longitude=112.699996
18:48:47.798    flowrate: 8.9L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 1.28LPublish ok
18:48:52.951    Location: Latitude=-7.443207, Longitude=112.699996
18:48:53.028    flowrate: 9.1L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 1.43LPublish ok
18:48:58.078    Location: Latitude=-7.443207, Longitude=112.699996
18:48:58.141    flowrate: 8.7L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 1.58LPublish ok
18:49:03.502    Location: Latitude=-7.443214, Longitude=112.699989
18:49:03.581    flowrate: 9.4L/min  Current Liquid Flowing: 0.16L/detik  Output Liquid Quantity: 1.74LPublish ok
18:49:08.625    Location: Latitude=-7.443214, Longitude=112.699989
18:49:08.734    flowrate: 8.7L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 1.88LPublish ok
18:49:13.895    Location: Latitude=-7.443214, Longitude=112.699989
18:49:13.974    flowrate: 9.1L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 2.04LPublish ok
18:49:19.021    Location: Latitude=-7.443214, Longitude=112.699989
18:49:19.112    flowrate: 8.8L/min  Current Liquid Flowing: 0.15L/detik  Output Liquid Quantity: 2.18LPublish ok
```

Gambar 4.29 *Publisher* Mengirim Paket

Data yang dikirm oleh *publisher* akan diterima oleh *subscriber* menggunakan aplikasi *MQTT*.fx. Berikut adalah hasil data yang berhasil diterima oleh *subscriber* yang dapat dilihat pada Gambar 4.30.



Gambar 4.30 Penerimaan Paket Pada *MQTT*.fx

Dapat dilihat bahwa data yang dikirim oleh *publisher* telah diterima dengan sangat baik oleh *subscriber* karena dari 15 paket yang dikirim oleh *publisher* tidak ditemukannya paket yang gagal atau tidak diterima oleh *subscriber* sehingga tidak ada *packet loss* pada penelitian ini. Sebagai referensi perbandingan *packet loss* antara paket yang dikirim dengan paket yang diterima maka dapat dilihat pada Tabel 4.13 dibawah ini.

Tabel 4.13 Hasil Pengujian Nilai Packet Loss

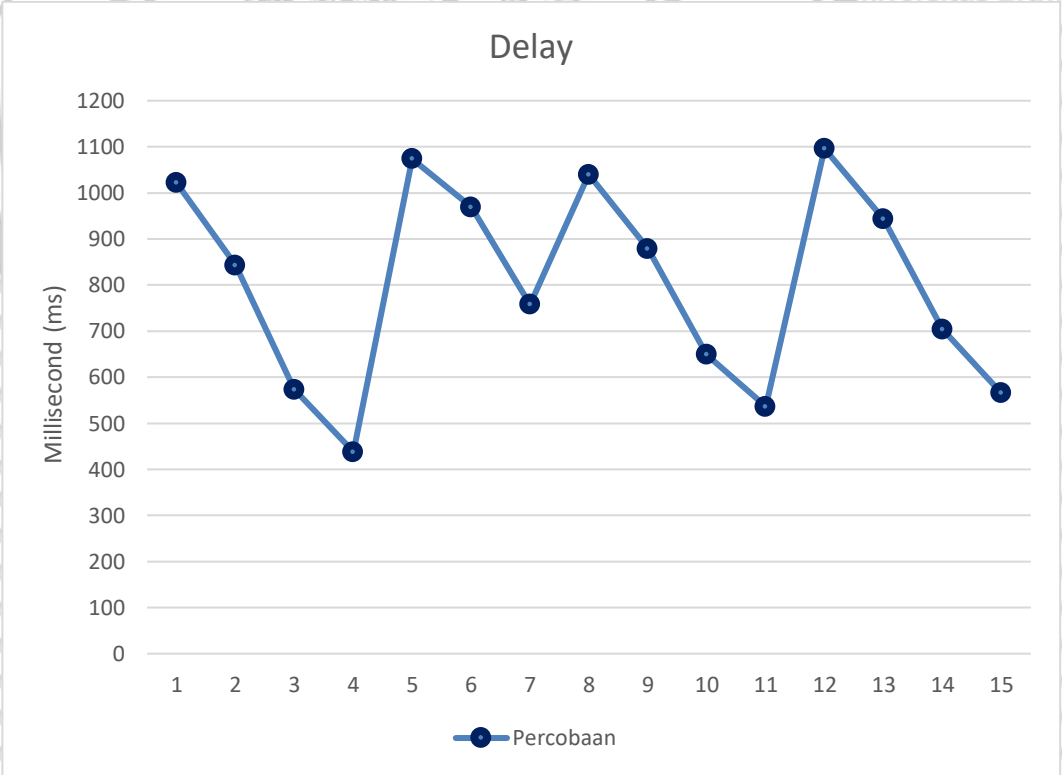
No	Debit Air *Publisher* (L)	Debit Air *Subscriber* (L)	Status
1	0.06	0.06	Berhasil
2	0.15	0.15	Berhasil
3	0.15	0.15	Berhasil
4	0.15	0.15	Berhasil
5	0.16	0.16	Berhasil
6	0.15	0.15	Berhasil
7	0.15	0.15	Berhasil
8	0.17	0.17	Berhasil
9	0.15	0.15	Berhasil
10	0.15	0.15	Berhasil
11	0.15	0.15	Berhasil
12	0.16	0.16	Berhasil
13	0.15	0.15	Berhasil
14	0.15	0.15	Berhasil
15	0.15	0.15	Berhasil

4.2.4 Hasil Analisis Pengujian Kinerja Pada Protokol *MQTT*

Setelah dilakukan pengujian menggunakan dua parameter yaitu *delay* dan *packet loss* maka Langkah selanjutnya yaitu membuat analisis. Hasil pengujian kinerja protokol *MQTT* berjalan dengan baik karena masing-masing parameter dapat dihitung nilainya sehingga dapat dijadikan sebagai bahan analisis.

4.2.4.1 Delay

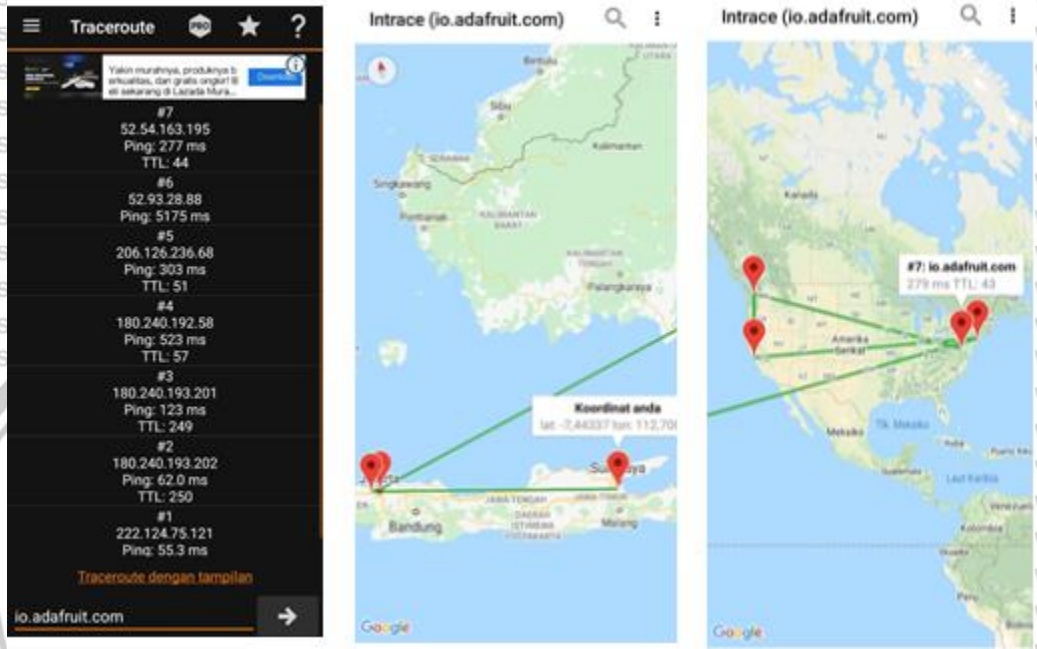
Delay adalah mengukur waktu yang dibutuhkan pada saat transmisi. Delay dapat dihitung dari perhitungan selisih dari waktu paket diterima dengan waktu paket dikirim. Parameter *delay* pada scenario ini menggunakan satuan ms (*millisecond*). Pada penelitian ini dilakukan pengujian *delay* sebanyak 15 kali percobaan sebagai bahan *sampling* uji. Dapat dilihat pada Gambar 4.31 bahwa nilai delay pada masing-masing paketnya bervariasi dan naik turun. Setiap paket memiliki perbedaan nilai *delay* yang berbeda dengan kisaran nilai yang tidak terpaut jauh. Naik turunya *delay* disebabkan oleh tidak stabilnya konektivitas internet 3G milik telkomsel serta performa antena modul yang kurang memadai. Terdapat juga faktor lain yang mempengaruhi yaitu banyaknya *noise* pada area pengujian yang menyebabkan komunikasi data sedikit terganggu. Nilai *delay* tertinggi pada pengujian ini yaitu pada paket nomor 12 dengan nilai 1096 ms sedangkan nilai terendahnya adalah 438 ms milik paket nomor 4. Jika mengacu pada standar TIPHON maka hanya terdapat satu paket yang masuk pada kategori sedang yaitu paket nomor 4 dengan nilai *delay* diantara 300 – 450 ms, sedangkan paket lainnya termasuk kategori jelek karena nilai *delay* lebih dari 450 ms. Dari 15 paket yang telah dilakukan pengujian *delay* setiap paketnya maka memiliki nilai rata-rata yaitu 806 ms.



Gambar 4.31 Grafik Nilai *Delay*

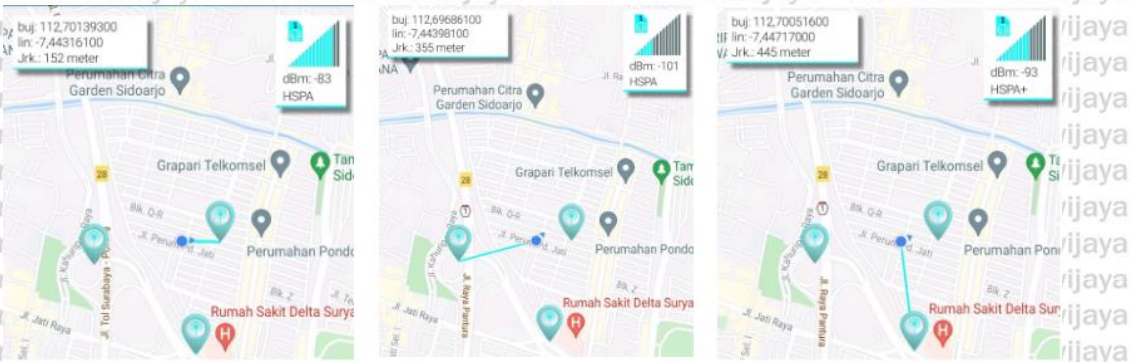
Terlihat bahwa *delay* pada penelitian ini cukup tinggi hal ini dikarenakan perjalanan saat transmisi data melalui banyak *hop* mulai dari *publisher* yang berlokasi di Indonesia lalu dikirim ke *broker* dengan lokasi server di Amerika yang kemudian data tersebut harus di *subscribe* dari Indonesia kembali. Untuk

mengetahui lokasi *broker* maka dapat dilihat menggunakan aplikasi Intrace seperti pada Gambar 4.32. Lokasi *broker* yang letaknya di Amerika adalah salah satu faktor besarnya nilai *delay* pada penelitian ini karena semakin jauh lokasi tujuan maka waktu yang diperlukan untuk tiba lebih besar serta jumlah *hop* juga berpengaruh karena jika semakin banyak jumlah *hop* maka proses perjalanan data semakin lama untuk bisa sampai ke tujuan. Untuk menguji lama waktu yang digunakan menuju *broker* maka menggunakan traceroute seperti pada Gambar 4.32 yang dimana masing-masing hop memiliki nilai ping yang cukup tinggi.



Gambar 4.32 *Traceroute* Lokasi

Tidak hanya jumlah hop count saja sebagai faktor besarnya *delay*, lemahnya jaringan 3G Telkomsel di lokasi *publisher* tidak terlalu banyak BTS yang mendukung konektivitas 3G dan antena pada modul SIM5360E kekuatannya tidak terlalu tinggi yaitu kurang lebih sekitar 3dBi sehingga sulit untuk menjangkau cakupan area dari BTS terdekat. Terlihat pada Gambar 4.33 bahwa di lokasi hanya terdapat 3 BTS terdekat di sekitar lingkup *publisher*. BTS terdekat dengan jarak 152 meter namun kuat sinyal yang didapat hanya -83dBm yang termasuk kategori buruk.

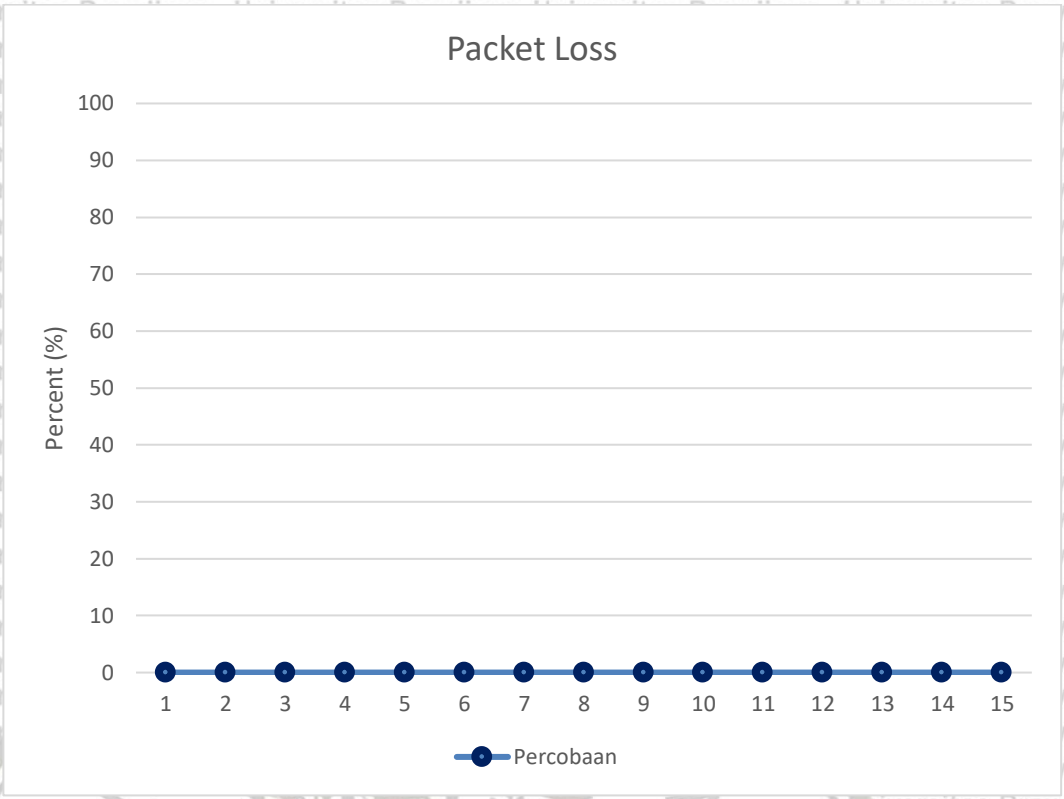


Gambar 4.33 Kuat Sinyal Terhadap Lokasi BTS

Berdasarkan hasil traceroute maka garis besar alasan mengapa nilai *delay* termasuk kategori jelek yaitu karena jumlah hop yang cukup banyak untuk dapat berkomunikasi dengan *broker* yaitu sebanyak 7 *hop*. Semakin banyak *hop* maka akan semakin lama karena setiap hop mengalami pengolahan data serta perpindahan ke *hop* selanjutnya juga membutuhkan waktu yang cukup lama. Lama waktu setiap hop juga bervariasi yang dimana waktu tertinggi jatuh pada *hop* ke 6 yaitu sebesar 5175 ms, namun besar atau kecilnya waktu setiap *hop* akan berubah-ubah bergantung pada kondisi jaringan saat proses transmisi data terjadi. Lokasi server *broker* Adafruit telah ditemukan yaitu berada di Amerika sehingga data membutuhkan waktu tempuh yang cukup lama untuk bisa mencapai tujuan.

4.2.4.2 Packet Loss

Packet loss adalah mengukur tingkat keberhasilan suatu paket untuk tiba pada tujuan. Parameter *delay* pada skenario ini menggunakan satuan persen (%) Pada penelitian ini dilakukan pengujian *packet loss* sebanyak 15 kali percobaan sebagai bahan *sampling* uji. Dapat dilihat pada Gambar 4.34 bahwa dari 15 percobaan yang diuji nilai *packet loss* keseluruhannya adalah 0% sehingga dapat diartikan bahwa semua paket yang dikirim oleh *publisher* menuju *subscriber* tidak ada yang mengalami kegagalan atau berhasil terkirim. Dalam pengukuran kualitas *packet loss* maka semakin rendah nilai yang diberikan maka menunjukkan performa yang bagus, namun jika nilainya tinggi maka dalam suatu transmisi terdapat beberapa paket yang mengalami kegagalan atau tidak terkirim. Tingkat konsistensi paket mulai dari percobaan pertama hingga akhir sangat stabil dan tidak mengalami naik turun nilai packet loss dari masing-masing paket yang dikirimkan.



Gambar 4.34 Grafik Nilai Packet Loss

Performa protokol *MQTT* yang berbasis TCP/IP menjamin akan keberhasilan paket yang dikirim pada suatu komunikasi. Telah diketahui sebelumnya dengan nilai *delay* yang cukup tinggi menurut standarisasi namun paket tersampaikan dengan sangat baik dengan persentase 0% *packet loss*. Semakin rendah nilai *packet loss* maka semakin bagus kualitasnya. Maka tingkat keberhasilan transmisi data pada penelitian ini hampir mencapai 100% karena rendah kemungkinan paket mengalami kegagalan. Jika menganut pada standarisasi TIPHON maka *packet loss* keseluruhan 15 paket termasuk dalam kategori sangat bagus sehingga menunjukkan bahwa tingkat akurasi data yang tinggi.

BAB 5 PENUTUP

5.1 Kesimpulan

Setelah mendapatkan hasil dari pengujian dan analisis maka pada penelitian ini akan ditarik kesimpulan seperti berikut:

1. Akurasi deteksi kebocoran diketahui melalui berubahnya warna status ikon pada halaman web yang semula berwarna hijau menjadi warna merah. Warna hijau adalah kondisi debit air normal, sedangkan warna merah adalah indikasi terjadi kebocoran.
2. Lama waktu yang dibutuhkan oleh sistem dalam mendeteksi kebocoran yaitu kurang lebih 5 detik.
3. Penggunaan algoritma pada *subscriber* yaitu dengan memberikan nilai batas ambang sebesar kurang lebih 30% yang kemudian dibandingkan dengan kondisi debit air pada saat kondisi normal. Perbandingan nilai tersebut akan menjadi tolak ukur kondisi jaringan pipa dalam keadaan normal atau terjadi kebocoran.
4. Kinerja protokol *MQTT* pada penelitian ini yaitu packet loss mendekati 0% dan *delay* rata-rata 806ms. Menurut standarisasi maka *packet loss* tergolong dalam kategori bagus, sedangkan *delay* termasuk dalam kategori jelek.

5.2 Saran

Untuk meningkatkan dan perbaikan pada penelitian ini agar menjadi lebih baik, maka diperlukan saran agar kedepannya sistem ini dapat berkembang yaitu sebagai berikut:

1. Dapat menggunakan modul internet dengan performa yang lebih baik untuk memaksimalkan kinerja *delay*.
2. Kedepannya dapat menggunakan jenis modul GPS yang lebih unggul untuk memudahkan saat proses implementasi dan data koordinat lebih stabil.
3. Informasi kebocoran dapat diinformasikan dengan cara yang lain seperti melalui sms, email, dan lain-lain.

DAFTAR PUSTAKA

Ahmady, H. (2018). Analisis Quality of Services Jaringan Berbasis Cisco di PT Kereta Api Indonesia [Institut Bisnis dan Informatika STIKOM]. In *Director*. https://www.uam.es/gruposinv/meva/publicaciones jesus/capitulos_espanyol_jesus/2005_motivacion para el aprendizaje Perspectiva alumnos.pdf%0Ahttps://www.researchgate.net/profile/Juan_Aparicio7/publication/253571379_Los_estudios_sobre_el_cambio_conceptual_

Alfeno, S., & Devi, R. E. C. (2017). Implementasi Global Positioning System (GPS) dan Location Based Service (LSB) pada Sistem Informasi Kereta Api untuk Wilayah Jabodetabek. *Sisfotek Global*, *7*(2), 27–33.

Andi, W. (2018). APLIKASI EXTRUDER MENGGUNAKAN SENSOR SUHU PADA ALAT PENCETAK AKRILIK TIGA DIMENSI. In *Politeknik Negeri Sriwijaya*.

Ardiansyah. (2016). Sistem Monitoring Air Layak Konsumsi Berbasis Arduino (Studi Kasus Pdam Patalassang). Skripsi. In *Makassar. UIN. Alauddin*. Universitas Islam Negeri Alauddin Makassar.

Arimbawa, I. W. A., Rahman, A. C., & Jatmika, A. H. (2019). Implementasi Internet of Things pada Sistem Informasi Pelacakan Kendaraan Bermotor Menggunakan GPS Berbasis Web. *Jurnal Teknologi Informasi, Komputer, Dan Aplikasinya (JTIKA)*, *1*(1), 121–130. https://doi.org/10.29303/jtika.v1i1.10

Arsyistawa, N., Rivai, M., & Suwito, S. (2017). Aplikasi Wireless Sensor Network Untuk Pembacaan Meteran Air. *Jurnal Teknik ITS*, *6*(2), 807–812. https://doi.org/10.12962/j23373539.v6i2.26648

Bagita, A. S. (2019). *Rancang Bangun Sistem Pendeteksi Kebocoran PDAM Menggunakan Sensor Tekanan Berbasis PID-Fuzzy*. Universitas Sriwijaya.

Dewanto, E., & Yoseph, J. (2018). Tandon Air Otomatis Dengan Sistem Monitoring Melalui Android Berbasis Arduino Uno. *Autocracy*, *5*(1), 8–16. https://doi.org/10.21009/autocracy.05.1.2

Dewi, K. H., Koosdaryani, & Muttaqien, A. Y. (2015). Analisis Kehilangan Air Pada Pipa Jaringan Distribusi Air Bersih Pdam Kecamatan Baki , Kabupaten Sukoharjo. *Jurnal UNS*, *9*(1), 9–16.

Dharmaadi, I. P. A., Made, D., & Arsa, S. (2020). Studi Pustaka Sistem Pemantauan Jaringan Distribusi Air Publik berbasis Internet of Things (IoT). *JURNAL ILMIAH MERPATI*, *8*(1), 54–60.

Fauzi, A. (2019). *Rancang Bangun Sistem Kendali AC, TV dan Pintu Gerbang Berbasis Inframerah Menggunakan Aplikasi Gawai*. Universitas Lampung.

Gunastuti, D. A. (2018). Pengukuran Debit Air Pelanggan Air Bersih Berbasis IoT Menggunakan Raspberry Pi. *EPIC (Journal of Electrical Power,*

Instrumentation and Control), *1*(2), 1–9. http://www.openjournal.unpam.ac.id/index.php/jit/article/view/1528

Haq, M. M. N. (2017). *Sistem Monitoring Pengendalian Pendistribusian Air PDAM Berbasis ATMEGA 16 dan Visual Basic 6* [Universitas Jendral Achmad Yani]. http://www.ghbook.ir/index.php?name=مجموعه مقالات دومین هم اندیشی سراسری رسانه تلویزیون و سکولاریسم&option=com_dbook&task=readonline&book_id=13629&page=108&chkhashk=03C706812F&Itemid=218&lang=fa&tmpl=component

Hariyanto, D. (2016). *Perancangan Alat Deteksi Letak Kebocoran Pipa PVC Menggunakan Sensor Flowmeter Model FS300A Berbasis TCP/IP*. Universitas Lampung.

Hermanto, D., Yamato, & Machdi, A. R. (2016). Perancangan Sistem Keamanan Berkendara Roda Dua Menggunakan Arduino Uno Berbasis Sms. *Jurnal Online Mahasiswa (JOM) Bidang Teknik Elektro*, *1*(1), 1–10. http://jom.unpak.ac.id/index.php/teknikelektro/article/view/506

Hwang, K., Lee, J. M., & Jung, I. H. (2019). Desing and implementation of MQTT protocol analyzer. *International Journal of Advanced Science and Technology*, *28*(5), 92–99.

Junaidi, & Prabowo, Y. D. (2018). Project Sistem Kendali Elektronik Berbasis Arduino. In *CV Anugrah Utama Raharja*. https://docplayer.info/109709787-Project-sistem-kendali-elektronik-berbasis-arduino-dr-junaidi-s-si-m-sc-yuliyan-dwi-prabowo.html

Lewi, E. B., Sunarya, U., & Ramadhan, D. N. (2017). Sistem Monitoring Ketinggian Air Berbasis Internet of Things Menggunakan Google Firebase. *E-Proceeding of Applied Science*, *3*(2), 1–8.

Lin, C. E. (2014). *Advance Real Time Mobile Surveillance System*. *October*.

Mayub, A., Fahmizal, Shidiq, M., Oktiawati, U. Y., & Rosyid, N. R. (2019). Implementation smart home using internet of things. *Telkomnika (Telecommunication Computing Electronics and Control)*, *17*(6), 3126–3136. https://doi.org/10.12928/TELKOMNIKA.v17i6.11722

Muhammad, M. M. (2017). *Sistem Monitoring Kontainer Truk Menggunakan Mikrokontroller Berbasis Web*. Universitas Muhammadiyah Malang.

Paksi, Y. E. E., & Prihartono, E. (2019). Sistem Monitoring Pemakaian Air Pdam Tirta Kencana Kota Samarinda Berbasis Arduino. *J I M P - Jurnal Informatika Merdeka Pasuruan*, *4*(2), 10–17. https://doi.org/10.37438/jimp.v4i2.203

Prasetya, A. D., Haryanto, H., & Wibisono, K. A. (2020). Rancang Bangun Sistem Monitoring dan Pendeteksi Lokasi Kebocoran Pipa Berdasarkan Analisis Debit Air Berbasis IoT. *Elektrika*, *12*(1), 39–47. https://doi.org/10.26623/elektrika.v12i1.2338

Pratama, R. Z., & Nurwarsito, H. (2019). Monitoring Penggunaan Daya Listrik

menggunakan Protokol MQTT berbasis Web. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *3*(11), 10820–10826.

Priangga, A. A., & Budisusanto, Y. (2015). PENGEMBANGAN SISTEM INFORMASI GEOGRAFIS BERBASIS MOBILE ANDROID UNTUK STUDI KEBOCORAN JARINGAN PIPA PDAM (Studi Kasus : Sub Zona 109 , Kota Surabaya). *Geoid*, *11*(1), 102–110.

Putra, N. D. (2018). *Wireless Smart Tag Device Sebagai Sistem Keamanan Rumah Sistem Keamanan Rumah*. Universitas Islam Indonesia.

Rachman, H., & Iskandar, I. (2017). *Analisis Kualitas Layanan Jaringan Komunikasi VoIP (Voice over Internet Protocol) Menggunakan Elastix Server Di Universitas Islam Negeri Sultan Syarif Kasim Riau*. 243–249.

Risna, R., & Pradana, H. A. (2014). Rancang Bangun Aplikasi Monitoring Penggunaan Air PDAM Berbasis Mikrokontroler Arduino Uno. *Jurnal Sisfokom (Sistem Informasi Dan Komputer)*, *3*(1), 60–66. https://doi.org/10.32736/sisfokom.v3i1.212

Rochman, H. A., Primananda, R., & Nurwasito, H. (2017). Sistem Kendali Berbasis Mikrokontroler Menggunakan Protokol MQTT pada Smarthome. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *1*(6), 445–455. http://j-ptiik.ub.ac.id

Sasmoko, D., Rasminto, H., & Rahmadani, A. (2019). Rancang Bangun Sistem Monitoring Kekeruhan Air Berbasis IoT pada Tandon Air Warga. *Jurnal Informatika Upgris*, *5*(1), 25–34. https://doi.org/10.26877/jiu.v5i1.2993

Setiadi, D., & Abdul Muhaemin, M. N. (2018). PENERAPAN INTERNET OF THINGS (IoT) PADA SISTEM MONITORING IRIGASI (SMART IRIGASI). *Infotronik : Jurnal Teknologi Informasi Dan Elektronika*, *3*(2), 95–102. https://doi.org/10.32897/infotronik.2018.3.2.108

Sugeng, W., Istiyanto, J. E., Mustofa, K., & Ashari, A. (2015). The Impact of QoS Changes towards Network Performance. *International Jurnal of Computer Networks and Communications Security*, *3*(2), 48–53. http://www.ijcncs.org/published/volume3/issue2/p5_3-2.pdf

Suharjono, A., Rahayu, L. N., & Afwah, R. (2015). Aplikasi Sensor Flow Water Untuk Mengukur Penggunaan Air Pelanggan Secara Digital Serta Pengiriman Data Secara Otomatis Pada PDAM Kota Semarang. *Teknik Elektro, Politeknik Negeri Semarang*, *13*(1), 7–12.

Sutiono, M. A. (2016). Rancang Bangun Smart Rice Cooker Menggunakan Protokol Komunikasi Wi-Fi Dan Protokol Pertukaran Pesan MQTT. In *Fakultas Teknik Dan Informatika Universitas Multimedia Nusantara*. Universitas Multimedia Nusantara.

Syaefudin. (2018). *Implementasi Algoritme A* pada Prototipe Robot Tempat Sampah*. Universitas Muhammadiyah Purwokerto.

Waworundeng, J. M. S., Tombeng, M., Maria, R., Komputer, F. I., & Klabat, U. (2019). *E-Water System : Prototipe Pemantauan Debit Air Berbasis Android E-Water System : Prototype of Monitoring Water Discharge Based on Android*. *5*(2), 280–293.

Yahdiani, A. (2020). *Analisis Kualitas Jaringan Internet Di Negeri Semarang Berdasarkan*. Universitas Negeri Semarang.