ANTIVIRUS: Jurnal Ilmiah Teknik Informatika (p – ISSN: 1978 – 5232; e – ISSN: 2527 – 337X)
Vol. 15 No. 2 November 2021, pp. 147-162

PENGEMBANGAN *DISCOVERABILITY* PADA SISTEM DETEKSI BANJIR KIRIMAN MENGGUNAKAN PROTOKOL *EDDYSTONE* URL BERBASIS *WEB OF THINGS*

Geraldhi Aditya Putra Mahayadnya [1)], Gusti Rafi Afkariansyah[2)], Franciscus Faust Hartanto[3)], Marselino Laksmana Syahjaya[4)] dan Harianto[5)]
[1, 2), 3), 4), 5)] Teknik Komputer, Universitas Dinamika
e-mail: geraldiaditya1111@gmail.com[1)], afkar.rafi@gmail.com[2)], franshartanto111@gmail.com[3)]
marselino115@gmail.com[4)], hari@dinamika.ac.id[5)]

Abstrak **:** *Teknologi* Internet of Things *pada sistem penanganan banjir kiriman di Indonesia sudah banyak diterapkan. Beberapa penelitian sebelumnya telah menunjukkan beberapa pendekatan dalam membuat sistem deteksi banjir kiriman diantaranya menggunakan SMS dan menggunakan Web yang memakai protokol MQTT. Namun di antara sistem deteksi yang sudah ada belum terdapat sebuah sistem* discoverability *untuk memungkinkan pengguna berinteraksi secara cepat dengan sistem tersebut. Pada penelitian ini, dibuat sebuah sistem deteksi banjir kiriman berbasis* Web of Things *dengan protokol komunikasi MQTT dan protokol* discoverability Eddystone URL. *Aplikasi* web *yang digunakan pada* Web of Things *menggunakan* Progressive Web App *untuk memungkinkan kompabilitas dengan perangkat mobile. Hasil pengujian menunjukkan bahwa penggunaan MQTT pada sistem deteksi banjir kiriman memiliki* latency *yang kurang dari 200ms, pemakaian* Eddystone URL *sebagai protokol* discoverability *pada sistem deteksi banjir menunjukkan respon sistem yang cukup baik dengan keberhasilan sebesar 90% pada sepuluh kali uji coba pada protokol* Eddystone URL*, dan pengujian aplikasi web dengan* Google Lighthouse *menunjukkan nilai rata-rata sebanyak 91 untuk aplikasi* web mobile *dan nilai rata-rata sebanyak 96,75 untuk aplikasi* web desktop.

Kata Kunci—Web Of Things, Eddystone URL, Progressive Web App, Physical Web, MQTT

Abstract **:** *Internet of Things technology in post-flood handling systems in Indonesia has been widely applied. Several previous studies have shown several approaches in creating a post-flood detection system, including using SMS and using the Web that using the MQTT protocol. However, among the existing detection systems there is not yet a discoverability system to allow users to interact quickly with the system. In this study, a Web of Things-based post-flood detection system was created with the MQTT communication protocol and the Eddystone URL discoverability protocol. Web applications that used on the Web of Things use a Progressive Web App to enable compatibility with mobile devices. The test results show that the use of MQTT in the post flood detection system has a latency of less than 200ms, the use of Eddystone URL as a discoverability protocol in the flood detection system shows a fairly good system response with 90% success in ten trials on the Eddystone URL protocol, and web application testing with Google Lighthouse showed an average score of 91 for mobile web applications and an average value of 96.75 for desktop web applications.*

Keywords— Web Of Things, Eddystone URL, Progressive Web App, Physical Web, MQTT

I. PENDAHULUAN

BANJIR adalah sebuah bentuk bencana alam yang disebabkan oleh luapan air berlebih pada sumber air, seperti sungai, danau, dan waduk sehingga menyebabkan genangan air pada daerah di sekitarnya. Salah satu jenis banjir yang perlu diwaspadai adalah jenis banjir kiriman. Banjir kiriman merupakan bentuk banjir dimana sumber luapan air berasal dari daerah lain dan mengalir ke daerah terdampak [1] [2]. Pada umumnya daerah-daerah yang sering dihampiri oleh bencana banjir sudah menyiapkan rencana-rencana mitigasi bencana banjir. Namun, bagi beberapa daerah yang sering mendapat kiriman luapan air dari daerah langganan banjir memiliki masalah dalam mitigasi bencana banjir. Faktor utama dari masalah mitigasi adalah kurangnya kewaspadaan dari masyarakan daerah tersebut karena banjir kiriman merupakan hal yang tidak

DOI: https://doi.org/10.35457/antivirus.v12i2.514

mudah diprediksi [3]. Diperlukan sebuah sistem deteksi banjir atau peringatan dini akan bahaya banjir yang akan mengingatkan warga di sekitar area terdampak dengan pemberitahuan tertentu.

Perkembangan teknologi telah merambah sampai kepada implementasi *Internet of Things* (IoT) yang telah digunakan untuk memecahkan permasalahan dalam hal penanganan banjir dan peringatan dini akan bahaya banjir. Penelitian yang dilakukan oleh Danang, Suwardi dan Hidayat menggunakan metode *SMS Gateway* untuk membuat sistem dan peringatan dini bencana banjir [2]. Sedangkan penelitian yang dilakukan oleh Prasetyo dan Setyawan menggunakan *bot telegram* yang dipakai untuk mengirimkan data pengukuran ketinggian air yang diolah menggunakan *Raspberry Pi* dan mendapatkan hasil rata-rata total *error* pada pengukuran sensor ultrasonik mencapai 0,27% dari 7 buah data pengujian [3]. Di samping itu, penelitian yang dibuat oleh Sagita dan Prapanca berhasil menerapkan konsep pemrograman *desktop* untuk membuat aplikasi pendeteksi level air dengan bantuan mikrokontroler Arduino [4]. Kemudian penelitian yang dilakukan oleh Nainggolan, Najoan, dan Karouw berhasil menerapkan protokol MQTT dalam pembuatan prototipe peringatan dini banjir di kota Manado berbasis web [5]. Hal ini juga dikuatkan oleh penelitian yang dilakukan oleh Habibi dengan perancangan web untuk sistem deteksi dini kawasan rawan banjir dengan notifikasi email [6].

Sistem SMS *Gateway* cocok untuk digunakan pada perangkat ponsel genggam untuk menerima pesan SMS. Sedangkan penggunaan bot telegram dipandang lebih efektif dalam sisi sinkronisasi pesan karena proses pengiriman dan penerimaan data dilakukan oleh aplikasi telegram yang terhubung ke perangkat minikomputer Raspberry Pi. Aplikasi *desktop* menggunakan *visual basic* juga cocok digunakan untuk banjir, namun hanya terbatas digunakan pada laptop dengan sistem operasi Windows. Sedangkan sistem *monitoring* dan deteksi banjir menggunakan web dipandang efektif dari sisi *monitoring* karena aplikasi web dapat diakses oleh perangkat *desktop* maupun perangkat *mobile*. Penggunaan aplikasi web dengan MQTT mampu membuat sistem *monitoring* deteksi banjir lebih responsif dari sisi desain [5] [6]. Namun pada kedua penelitian yang menggunakan aplikasi web, terdapat kekurangan dari sisi *discoverability*, yaitu pengguna harus mencari URL yang cocok untuk mengakses aplikasi web tersebut. Pencarian URL yang tidak cocok menyebabkan pengguna kesulitan dalam mengakses aplikasi web untuk melakukan *monitoring* dan deteksi dini bencana banjir.

Dari hasil studi literatur dan observasi, penulis dapat memberikan solusi dengan menerapkan *discoverability* pada sistem deteksi banjir menggunakan protokol *Eddystone URL* yang akan ditransmisikan melalui *Bluetooth*. Dan mentransmisikan URL aplikasi web yang digunakan untuk deteksi dan *monitoring* banjir atau dapat disebut dengan *Web of Things*. Proses yang digunakan dalam *Web of Things* menggunakan sebuah web yang terhubung pada MQTT broker dimana data dari sensor *ultrasonic* dan sensor *water level* diterima langsung secara *real-time*. Penggunaan *Web of Things* juga memanfaatkan ReactJS yang menjadi dasar dari pembuatan *Progressive Web App* (PWA). Penggunaan ReactJS memungkinkan untuk membuat aplikasi web dengan *component* web yang dapat digunakan ulang sehingga pembuatan aplikasi web dapat menjadi lebih dinamis. Penelitian ini bertujuan untuk menerapkan *discoverability* pada sistem deteksi banjir kiriman menggunakan *Web of Things* menggunakan *Eddystone* URL

II. TINJAUAN PUSTAKA

A. *Internet of Things*

Internet of Things (IoT) adalah sebuah konsep dengan tujuan memperluas manfaat dari koneksi internet yang terubung secara non-stop, IoT memiliki kemampuan untuk melakukan transmisi data dengan menggunakan perantara jaringan internet. Implementasi pada perangkat sehari-hari yang memungkinkan untuk dikontrol dan di menggunakan perangkat yang memiliki kemampuan untuk terhubung ke jaringan internet [7]. Sensor pada IoT digunakan untuk mengambil data aktual pada lingkungan yang masih berupa data mentah dijadikan sinyal digital dan data tersebut dikirimkan ke pusat kontrol.

DOI: https://doi.org/10.35457/antivirus.v15i2.1611

B. NodeMCU

NodeMCU adalah sebuah platform *Internet of Things* (IoT) yang bersifat *open source*. NodeMCU terdiri dari *hardware* berupa *System On Chip* (SOC) ESP8266 dan *firmware* yang digunakan menggunakan bahasa pemrograman C/C++. NodeMCU ini memiliki antarmuka jaringan Wi-Fi dengan frekuensi 2.4Ghz.

Board NodeMCU memiliki berbagai fitur mikrokontroler yang memiliki kapabilitas akses *wi-fi* dan juga memiliki *chip* komunikasi USB to serial. Sehingga supaya NodeMCU dapat terhubung dengan komputer atau laptop hanya memerlukan kabel data type-B yang biasa digunakan untuk melakukan *charging* pada *smartphone.*

C. Water level Sensor

Water level Sensor adalah kesatuan perangkat untuk melakukan pemantauan tingkat kedalaman air dan sebagainya. *Water level Sensor* dapat melakukan pemantauan secara *real-time* dengan akurat. Dengan dilengkapi sensor yang terintegrasi dan alat yang akan digunakan dapat membantu sensor dalam proses pemantauan. *Water level Sensor* yang dipakai memiliki rentang pembacaan antara 0-4 cm pada sisi permukaannya.

D. Sensor Ultrasonik

Sensor Ultrasonik adalah sebuah sensor yang memanfaatkan pantulan gelombang. Gelombang pantulan tersebut dapat dimanfaatkan sebagai pengukur kedalaman air dengan menggunakan rumus tertentu yang menyesuaikan dengan tingkat kedalaman lokasi tempat yang akan diukur. Sensor ultrasonik yang dipakai bertipe HC-SR04.

E. Eddystone URL

Eddystone URL merupakan salah satu format pengiriman dari protokol *Eddystone* sebagai format pengiriman data pada perangkat *Bluetooth Low Energy* (BLE). *Eddystone* URL bekerja dengan cara mengirimkan URL dengan panjang maksimal 17 bit [8]. URL tersebut akan dikompresi dan dikirimkan Bersama *Eddystone Frame Header* dan diterima oleh perangkat yang berada di dekatnya. Ketika sebuah Frame *Eddystone* URL berhasil diterima dan diolah, maka URL yang telah dikirimkan dapat dikunjungi oleh pengguna. Konsep dari *Eddystone* URL sendiri sangat diperlukan dalam keberlangsungan dari konsep *Physical Web*.

Gambar 1. Format pesan *Eddystone* URL

F. Physical Web

Physical Web adalah sebuah konsep *Discoverability* dimana seseorang dapat berinteraksi secara cepat dan mudah dengan perangkat di dekatnya melalui web. Hal ini dapat dicapai dengan menggunakan URL *Broadcaster* yang mentransmisikan URL melalui media transmisi nirkabel [9]. Transmisi nirkabel yang dimaksud adalah melalui *Bluetooth* dengan perangkat *Bluetooth Low Energy* (BLE). Transmisi ini menggunakan *Bluetooth* karena konsep dari *Physical Web* ini berkonsep untuk mengenali sesuatu yang berada dekat dengan pengguna. Konsep *Physical Web* bekerja dengan cara menerima pesan URL yang disebarkan melalui perangkat yang dinamakan *Beacon*, kemudian mengolah pesan tersebut dan menampilkannya pada aplikasi Android. Dengan demikian, pengguna hanya perlu membuka aplikasi *Physical Web*, mendapatkan daftar URL yang dituju, dan membuka web tersebut untuk berinteraksi dengan perangkat fisik yang diinginkan. *Physical Web* juga membutuhkan protokol *Eddystone* URL untuk berkomunikasi dengan *Advertising interval* tidak lebih dari 700ms per detik.

DOI: https://doi.org/10.35457/antivirus.v15i2.1611

G. Web of Things

Web of Things merupakan sebuah penerapan *Internet of Things* yang menggunakan web sebagai media untuk berinteraksi antara pengguna dengan obyek yang dikontrol melalui *Internet of Things* [10]. Kelebihan dari model *Web of Things* ini adalah pengguna tidak perlu menginstall aplikasi tambahan untuk menjalankan proses *Internet of Things* seperti *monitoring* atau kontroling terhadap obyek yang dikontrol. Selain itu, *Web of Things* juga memungkinkan untuk terintegrasi dengan Ponsel Android dengan konsep *Progressive Web App*.

H. Progressive Web App

Progressive Web App (PWA) adalah sebuah teknologi web modern yang digagas oleh Google pada Tahun 2015. Konsep web modern ini dibuat untuk mengatasi permasalahan aplikasi web yang dibuat secara *native* pada Android [11]. Artinya, bahwa konsep PWA mampu membuat sebuah aplikasi web bertindak layaknya aplikasi Android secara bawaan dan mampu diluncurkan dari *home* menggunakan sebuah *shortcut* yang berasal dari browser *Chrome*. Kelebihan dari PWA ini adalah mampu dijalankan secara *offline*, mempunyai *push notifications*, dan mempunyai *splash screen* yang akan muncul setiap aplikasi dijalankan.

I. Message Queuing Telemetry Transport (MQTT)

Message Queuing Telemetry Transport (MQTT) adalah protokol *messaging* yang digunakan untuk komunikasi dalam *Internet of Things* (IoT). Sistem kerja MQTT adalah dengan menerapkan sistem *publish* dan *subscribe. Publish* memiliki tugas untuk melakukan pengiriman data yang akan diterima oleh *subscriber* melalui sebuah *broker* [5]. Berikut ini adalah penjelasan detail tentang *publisher*, *subscriber*, dan *broker*:

1. *Publisher*, memiliki tugas untuk mengirimkan data dan *publisher* diintegrasikan dengan berbagai macam sensor, data sensor yang diperoleh dikirimkan ke *broker*.
2. *Subscriber*, memiliki tugas untuk menerima data yang dikirimkan oleh *publisher*.
3. *Broker*, berperan dalam menampung seluruh data yang dikirimkan oleh *publisher* dan data tersebut akan dikirimkan pada *subscriber* dengan *topic* sama dengan *topic* yang diberikan oleh *publisher*.

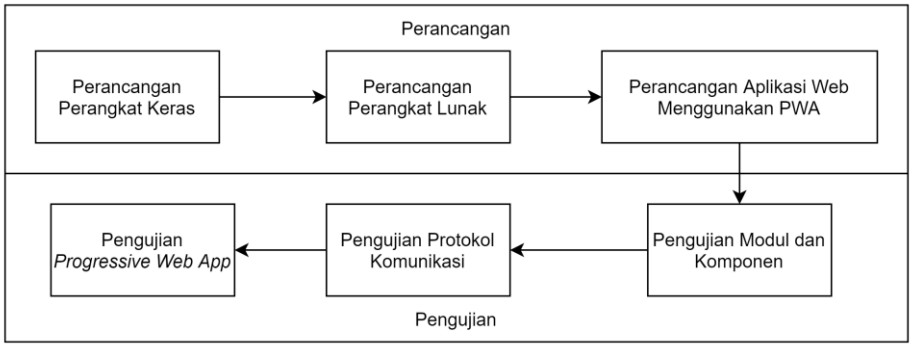
III. METODE PENELITIAN

A. Alur Penelitian

Alur penelitian ini menggunakan model perancangan dan pengujian. Perancangan pada penelitian ini meliputi perancangan perangkat keras, perangkat lunak, dan aplikasi web. Perancangan pada perangkat keras akan mendeskripsikan perancangan sistem pada sisi *hardware*. Sedangkan perancangan pada perangkat lunak akan mendeskripsikan perancangan sistem pada sisi *software* terutama pada pengembangan algoritma pada sistem deteksi banjir kiriman. Perancangan aplikasi web akan menjelaskan perancangan aplikasi web yang berfungsi untuk melakukan *monitoring* pada sistem deteksi banjir kiriman.

Pengujian pada penelitian ini mencakup tiga hal. Tiga hal tersebut meliputi pengujian modul dan komponen, pengujian protokol komunikasi, dan pengujian *Progressive Web App*. Pengujian modul dan komponen akan menguji seberapa akurat sensor dan modul yang dipakai dalam sistem deteksi banjir kiriman. Sedangkan pengujian protokol komunikasi akan menguji kehandalan komunikasi pada protokol MQTT dan *Eddystone* URL dalam pengiriman data. Kemudian Pengujian *Progressive Web App* akan menguji performa aplikasi web yang menggunakan PWA menggunakan Google *Lighthouse*. Gambar 2 mendeskripsikan alur penelitian pada segi perancangan dan pengujian pada sistem deteksi banjir kiriman.

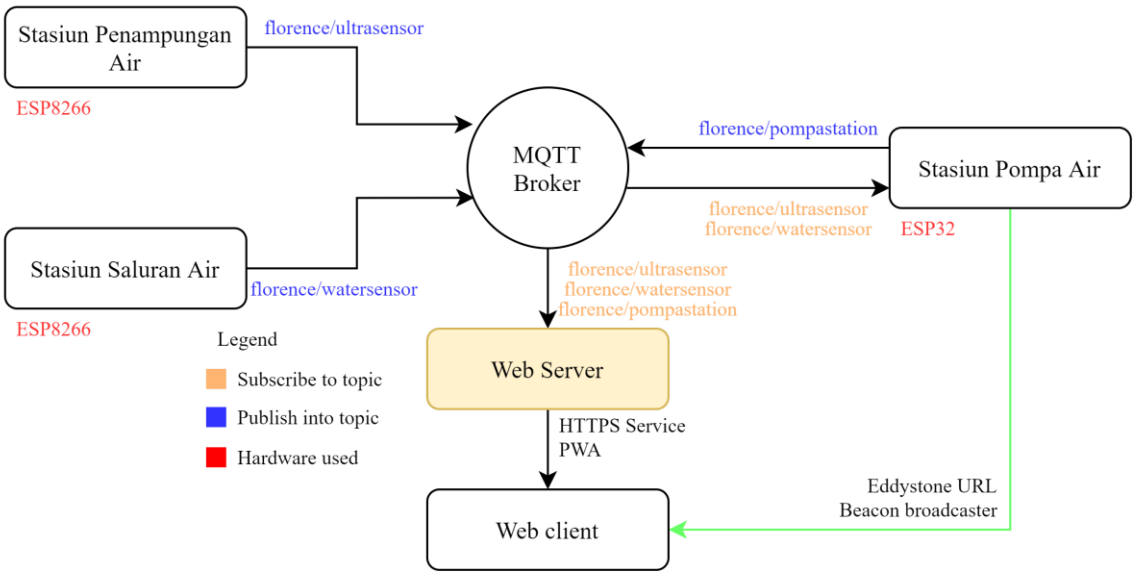
DOI: https://doi.org/10.35457/antivirus.v15i2.1611



Gambar 2. Alur penelitian pada sistem deteksi banjir kiriman

B. Gambaran Umum Sistem

Gambaran umum sistem menjelaskan tentang bagaimana sebuah sistem dapat saling terhubung dan berkomunikasi satu sama lain dan bertukar informasi. Pada penelitian ini blok diagram dibuat dengan menggunakan tiga stasiun yang berbeda. Stasiun ini menyimbolkan beberapa titik sensor dan aktuator yang akan ditempatkan. Tiga stasiun yang dimaksud antara lain: stasiun penampungan air, stasiun saluran air, dan stasiun pompa. Setiap stasiun memiliki topik MQTT yang berbeda untuk bertukar informasi. Keberadaan protokol MQTT pada sistem ini sangat penting untuk menunjang komunikasi dan mengatur aktuator berdasarkan data yang telah dikirim oleh sensor. Setiap stasiun akan berbagi informasi melalui MQTT *broker* yang berada di internet. MQTT *broker* yang dipilih oleh penulis adalah CloudMQTT. Detail tentang topik-topik yang akan dikirim dapat dilihat pada Gambar 3.



Gambar 3. Blok perancangan pengembangan protokol *Eddystone* URL terhadap sistem deteksi banjir kiriman

Sesuai dengan Gambar 3, pada stasiun pompa air akan disiapkan sebuah *beacon* yang akan memancarkan paket dengan protokol *Eddystone* URL. Paket yang dipancarkan berisi URL untuk membuka sebuah situs web yang akan disediakan oleh *web server*. Kemudian *web client* akan membuka URL yang telah dipancarkan dengan protokol *Eddystone* URL menggunakan aplikasi "*Physical Web*". Setelah itu *web server* akan memulai koneksi menuju MQTT *broker* dan melakukan *subscribe* pada topik-topik yang telah ditentukan sebelumnya. Kemudian web server akan membaca data yang telah didapat dari MQTT dan menyajikannya kepada *web client* berupa *Progressive Web App* yang telah dibuat oleh ReactJS.

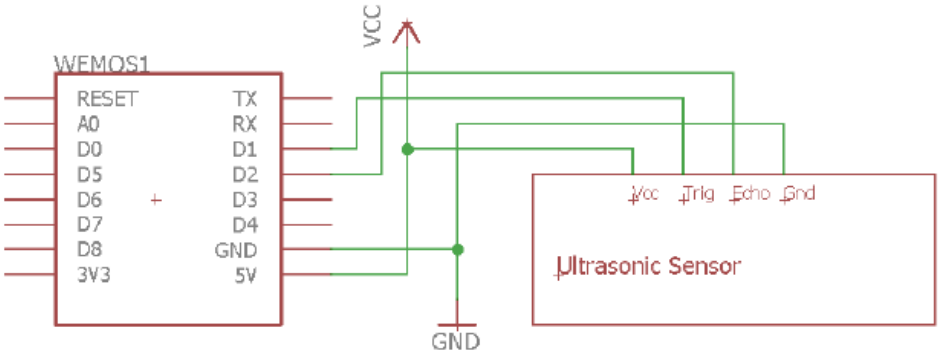
DOI: https://doi.org/10.35457/antivirus.v15i2.1611

C. Perancangan Perangkat Keras

Perancangan perangkat keras pada pengembangan protokol *Eddystone* URL meliputi perancangan perangkat keras pada Stasiun Penampungan Air, Stasiun Saluran Air, dan Stasiun Pompa Air. Berikut ini adalah penjelasan detail mengenai perancangan perangkat keras tersebut.

1) Diagram Skematik Stasiun Penampungan Air

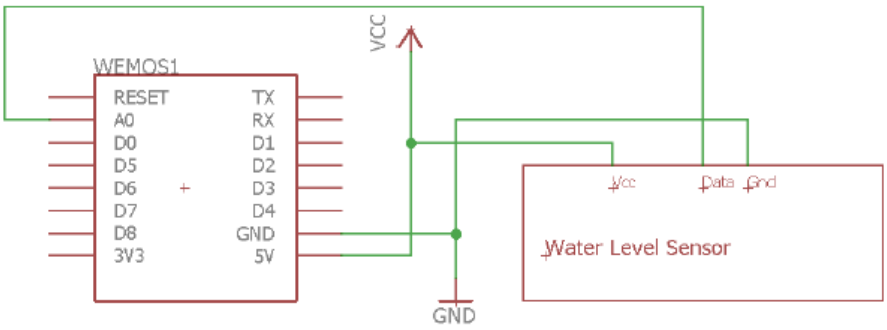
Pada sensor *ultrasonic* dipasang pada ESP8266 dengan sumber VCC berada pada tegangan 5V. Pin *trigger* dan pin *echo* dipasang pada ESP8266 masing-masing berada pada pin D1 dan D2. Sensor *ultrasonic* dipakai untuk mendeteksi kedatangan banjir kiriman dari daerah pantauan. Detail skematik dapat dilihat pada Gambar 4.



Gambar 4. Diagram Skematik Stasiun Penampungan Air

2) Diagram Skematik Stasiun Saluran Air

Pada sensor *water level* dipasang pada ESP8266 dengan sumber VCC berada pada tegangan 5V. Pin *data* dipasang pada ESP8266 di pin A0. Sensor *water level* dipakai untuk mendeteksi debit air pada saluran air untuk memantau *threshold* dari kedalaman air. Detail skematik dapat dilihat pada Gambar 5.

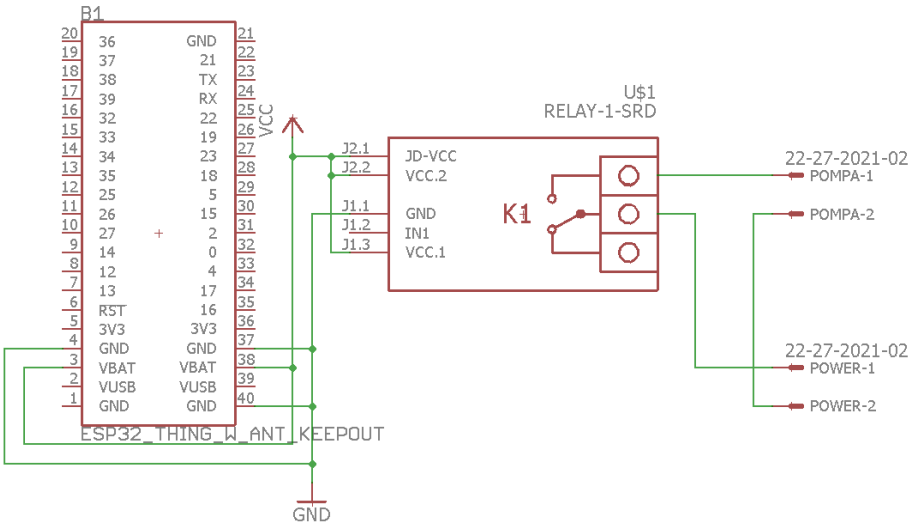


Gambar 5. Diagram Skematik Stasiun Saluran Air

3) Diagram Skematik Stasiun Pompa Air

Pada pompa air akan dipasang pada *relay module*, yang terpasang pada ESP32 dengan sumber VCC berada pada tegangan 5V. terdapat dua pin yang terpasang pada relay module untuk bagian *primary* dan cadangan. Pompa air akan digunakan untuk memindahkan air dari daerah hilir sungai menuju sumur resapan atau penampungan air. Setelah itu pada ESP32 akan dijalankan *Beacon Bluetooth* yang mengrim paket dengan protokol *Eddystone* URL. Detail skematik dapat dilihat pada Gambar 6.

DOI: https://doi.org/10.35457/antivirus.v15i2.1611

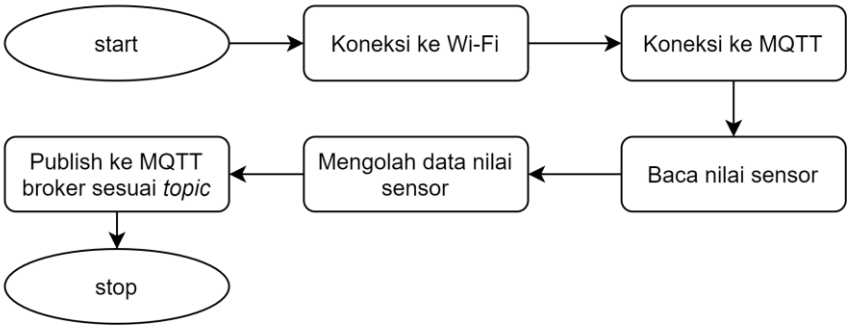


Gambar 6. Diagram Skematik Stasiun Pompa Air

D. Perancangan Perangkat Lunak

Perancangan perangkat lunak pada pengembangan protokol *Eddystone* URL meliputi perancangan perangkat lunak pada Stasiun Penampungan Air, Stasiun Saluran Air, dan Stasiun Pompa Air. Setelah itu pada bagian ini akan menjelaskan tentang perancangan *Web Server* pada sistem deteksi banjir kiriman yang terintegrasi dengan *Eddystone* URL dan perancangan pengiriman data menggunakan *Eddystone* URL itu sendiri. Berikut ini adalah penjelasan detail mengenai perancangan perangkat lunak tersebut.

1) *Flowchart* Program Stasiun Penampungan Air dan Stasiun Saluran Air



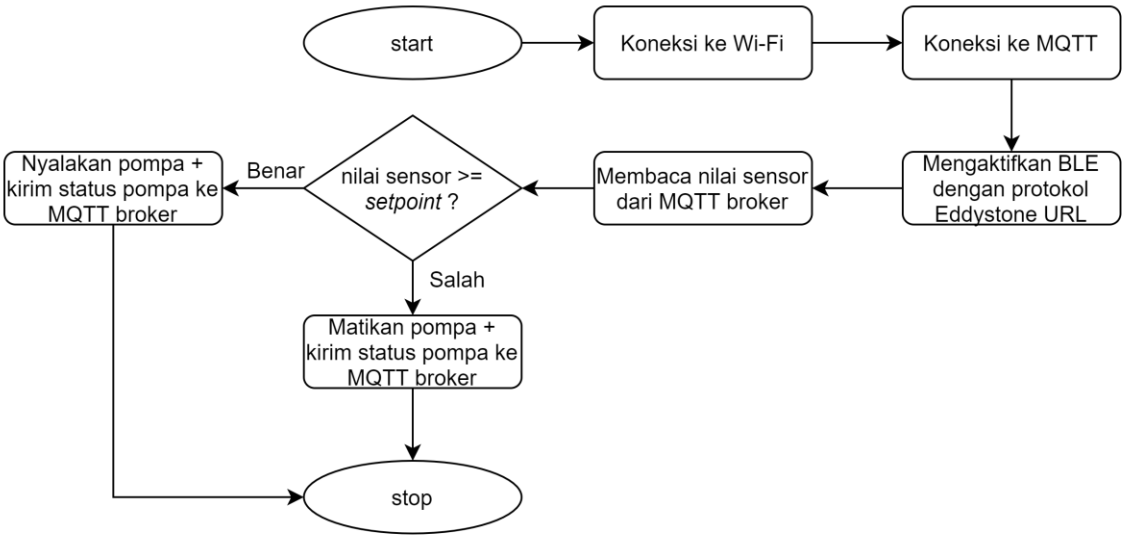
Gambar 7. *Flowchart* program stasiun penampungan air dan stasiun saluran air

Berdasarkan Gambar 7, Algoritma pada program stasiun penampungan air dan stasiun saluran air memiliki pola yang sama. Persamaan tersebut adalah pola pembacaan sensor dan pengiriman data hasil bacaan sensor melalui protokol MQTT dengan *topic* yang berbeda. Kemudian algoritma ini akan menjalankan proses untuk mengolah data sesuai dengan nilai sensor yang ada. Setelah itu MQTT broker akan menerima pesan tersebut dan menyalurkannya menuju *subscriber* yang telah siap menerima data dari MQTT broker.

2) Flowchart Program Stasiun Pompa Air

Algoritma pada program stasiun pompa air akan membaca nilai dari MQTT broker dan mengaktifkan *Bluetooth Low Energy* untuk mentransisikan data dengan protokol *Eddystone* URL. *Eddystone* URL akan mentransmisikan URL yang sudah didefinisikan di dalam algoritma tersebut. Setelah itu algoritma ini akan mengaktifkan kondisi apabila data nilai yang telah ditransmisikan sudah sesuai dengan *setpoint* yang sudah ditentukan sesuai dengan keadaan dari masing-masing stasiun. Diagram *flowchart* yang menjelaskan hal tersebut dapat dilihat pada Gambar 8.

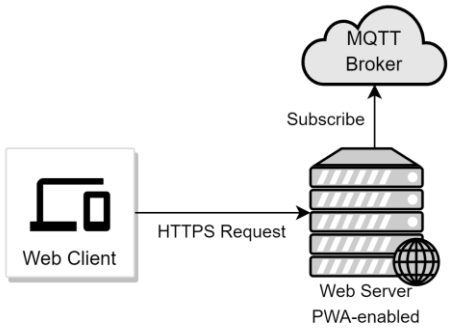
DOI: https://doi.org/10.35457/antivirus.v15i2.1611



Gambar 8. *Flowchart* program stasiun pompa air

3) Perancangan Web Server menggunakan PWA

Perancangan Web Server pada sistem pengembangan deteksi banjir kiriman ini terdapat dua tahapan dalam perancangannnya, antara lain perancangan aplikasi web server dengan ReactJS dan meluncurkan aplikasi tersebut dalam server hosting yang menggunakan protokol HTTPS. Perancangan aplikasi web server dengan ReactJS menggunakan NodeJS sebagai platform aplikasi tersebut. Aplikasi web server yang akna dibuat juga menggunakan bantuan *library* MQTT untuk mengambil data dari MQTT broker dan memasukkannya ke dalam aplikasi web untuk selanjutnya melakukan *rendering*. Gambar 9 menggambarkan sistem arsitektur yang dipakai pada aplikasi web yang menggunakan ReactJS.



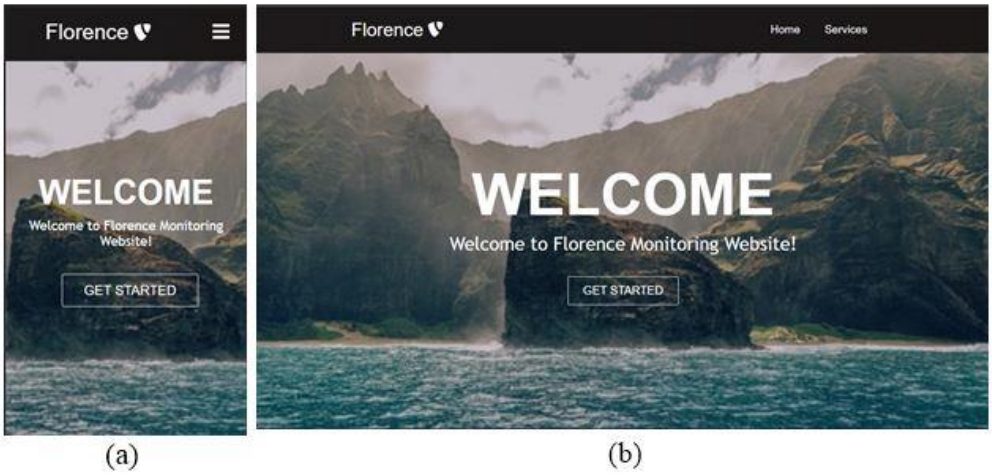
Gambar 9. Gambaran umum perancangan web server

Pada perancangan web server, aplikasi web dirancang menggunakan ReactJS yang merupakan salah satu *library* dari NodeJS. Selain itu aplikasi web ini juga akan memakai *library* MQTT untuk mendapatkan data dari MQTT broker melalui *Websocket*. *Websocket* yang aktif pada sisi MQTT broker memungkinkan untuk mengakses data yang mempunyai protokol MQTT melalui protokol *Websocket* yang dibungkus melalui HTTPS. URL untuk mengakses *Websocket* ini diawali dengan *wss://[url-broker]*.

Setelah itu web server dirancang dengan menggunakan konsep *Progressive Web App* dengan mengaktifkan *service worker* pada aplikasi web. Setelah itu melakukan penyuntingan pada file manifest.json yang akan menjadi identitas resmi dari aplikasi web yang akan dirancang. Kemudian melakukan optimasi pada file-file *source code* dari aplikasi web agar performa pada aplikasi web tersebut menjadi lebih baik. Gambar 10(a) menunjukkan desain aplikasi web yang telah dirancang dengan tampilan berbasis *mobile*. Aplikasi web yang telah menjadi Progressive web apps dapat memiliki jalan pintas *(shortcut)* yang dapat ditaruh pada beranda Android untuk selanjutnya dapat diakses secara langsung tanpa harus mengetik URL pada browser. Selain

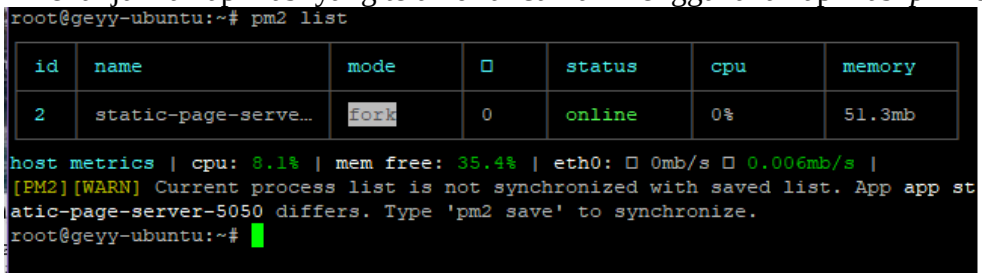
DOI: https://doi.org/10.35457/antivirus.v15i2.1611

itu, Gambar 10(b) menunjukkan bahwa desain aplikasi web juga mendukung tampilan berbasis *desktop* sehingga aplikasi web ini dapat berjalan secara *multi-platform*.



Gambar 10 (a). Desain web tampilan *mobile* (b). Desain web tampilan *desktop*

Kemudian aplikasi web yang telah dirancang diluncurkan pada server yang berada di Internet dengan konsep *Cloud Computing*. Aplikasi web tersebut berjalan di belakang *reverse proxy* yang dijalankan menggunakan aplikasi nginx. Tujuan dari konsep *reverse proxy* ini adalah untuk meminimalkan port yang terbuka di sisi server dan memperkecil dampak serangan akibat dari *port-scanning*. Setelah itu aplikasi web akan diluncurkan pada mode *production* menggunakan aplikasi *pm2* dan *serve*. Setelah itu server akan mengaktifkan HTTPS untuk memastikan bahwa aplikasi web sudah kompatibel terhadap konsep *Physical Web*. Gambar 11 menunjukkan aplikasi yang telah diluncurkan menggunakan aplikasi *pm2* dan *serve*.



Gambar 11. Meluncurkan aplikasi web menggunakan *pm2* dan *serve*

IV. HASIL DAN PEMBAHASAN

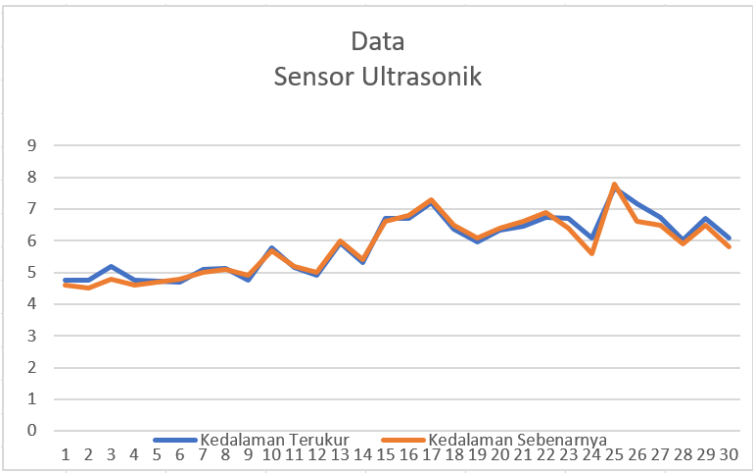
A. Pengujian Modul dan Komponen

Pengujian modul dan komponen ini akan menguji performa dari modul dan sensor yang akan digunakan untuk menjalankan keseluruhan sistem yang ada. Pengujian ini meliputi pengujian sensor ultrasonik dan pengujian sensor *water level*. Pengujian sensor ultrasonik ini akan menjamin bahwa sensor tersebut siap digunakan dalam sistem dengan *error* yang minimal. Begitu pula dengan sensor *water level* yang siap digunakan dalam stasiun saluran air.

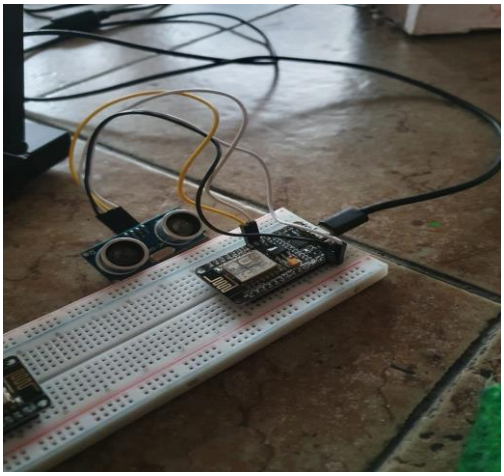
1) *Pengujian Sensor Ultrasonik*

Hasil pengujian yang didapat dari sensor ultrasonik yaitu dapat menentukan kedalaman sebenarnya. Kedalaman sebenarnya dapat digunakan oleh peneliti sebagai data untuk menentukan kondisi pompa aktif atau tidak aktif. Pada Gambar 12 dapat dilihat grafik dengan data kedalaman terukur dan kedalaman sebenarnya. Kedalaman terukur adalah kedalaman yang diukur oleh sensor dan kedalaman sebenarnya adalah kedalaman yang diukur secara manual. Hasil pengujian ini menunjukkan rata-rata *error* sebesar 3,5% dalam 30 sampel data uji pada sensor ultrasonik.

DOI: https://doi.org/10.35457/antivirus.v15i2.1611



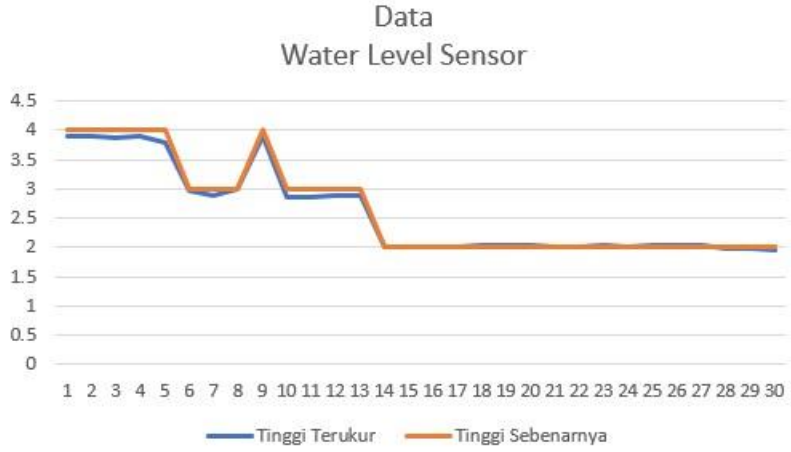
Gambar 12. Grafik data pengujian Sensor Ultrasonik



Gambar 13. ESP8266 dan Sensor Ultrasonik

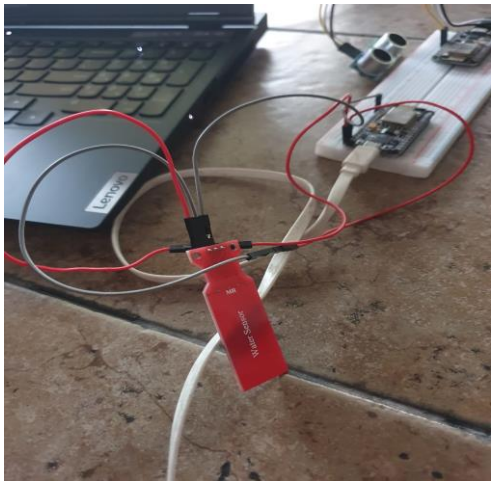
2) *Pengujian Sensor Water level*

Hasil pengujian dari sensor *water level* yaitu dapat digunakan sebagai pemindai yang akan digunakan untuk menentukan perkiraan penambahaan debit air pada sungai dalam estimasi waktu tertentu sehingga dapat diperkirakan kapan diperlukan pompa aktif. Hasil pengujian ini menunjukkan rata-rata *error* sebesar 2,5% pada 30 sampel data uji *Water level Sensor*. Pengujian ini didasarkan pada pembacaan ketinggian air pada permukaan *Water level Sensor*.



Gambar 14. Grafik data pengujian *Water level Sensor*

DOI: https://doi.org/10.35457/antivirus.v15i2.1611



Gambar 15. ESP8266 dan *Water level* Sensor

3) *Pengujian Pompa Air*

Hasil pengujian dari aktuator (pompa air) yaitu digunakan sebagai pengatur debit air. Aktuator ini terpasang pada stasiun pompa air untuk memindahkan air dari daerah terdampak banjir ke daerah penampungan air seperti waduk atau lautan. Tabel 1 menunjukkan hasil pengujian pada sepuluh data uji dengan keberhasilan 100%.

Tabel 1. Hasil pengujian aktuator pompa air

Jarak Aktual	Status	
	Keluaran Diharapkan	Keluaran Sebenarnya
11 Cm	Hidup	Hidup
12 Cm	Hidup	Hidup
13 Cm	Hidup	Hidup
14 Cm	Hidup	Hidup
15 Cm	Hidup	Hidup
16 Cm	Hidup	Hidup
17 Cm	Hidup	Hidup
18 Cm	Hidup	Hidup
19 Cm	Hidup	Hidup
20 Cm	Mati	Mati

B. Pengujian Protokol Komunikasi

Pengujian protokol komunikasi akan menguji performa protokol komunikasi digunakan untuk menjalankan keseluruhan sistem yang ada. Pengujian ini meliputi pengujian protokol komunikasi MQTT dan protokol komunikasi *Eddystone* URL yang menggunakan *Bluetooth*. Pengujian protokol komunikasi MQTT akan menguji bagaimana kinerja protokol MQTT dalam pengiriman data yang diukur dengan *latency*. Sedangkan pengujian protokol *Eddystone* URL akan menguji pengiriman data dengan model *discoverability* menggunakan *Bluetooth* dengan parameter jarak dari pengguna menuju pusat pemancar *Bluetooth*.

1) *Pengujian Protokol MQTT*

Pengujian protokol MQTT ini akan menguji *latency* dari protokol MQTT dalam melakukan *publishing* ke dalam MQTT broker. Pengujian ini dilakukan dengan mengirimkan data hasil olahan dari data sensor ultrasonik, sensor *water level*, dan status nyala pompa air melalui protokol MQTT dengan *topic* yang berbeda. Setelah itu protokol MQTT akan mengirimkan ketiga data tersebut kepada MQTT broker. Setelah itu MQTT broker akan menerima pesan tersebut dan menyalurkannya menuju *subscriber* yang telah siap menerima data

DOI: https://doi.org/10.35457/antivirus.v15i2.1611

dari MQTT broker. Tabel 2, Tabel 3, dan Tabel 4 menunjukkan *latency* pengiriman data dari sepuluh sampel data yang dikirimkan dari tiap data tersebut.

Tabel 2. Proses Pengiriman Sensor Ultrasonik

No	ESP8266	
	Latency	Pesan Sensor Ultrasonik
1	190ms	Terkirim
2	140ms	Terkirim
3	132ms	Terkirim
4	148ms	Terkirim
5	157ms	Terkirim
6	194ms	Terkirim
7	170ms	Terkirim
8	165ms	Terkirim
9	113ms	Terkirim
10	133ms	Terkirim

Tabel 3. Proses Pengiriman Sensor *Water level*

No	ESP8266	
	Latency	Pesan *Water level* Sensor
1	142ms	Terkirim
2	151ms	Terkirim
3	122ms	Terkirim
4	140ms	Terkirim
5	174ms	Terkirim
6	132ms	Terkirim
7	152ms	Terkirim
8	179ms	Terkirim
9	105ms	Terkirim
10	132ms	Terkirim

Tabel 4. Proses Pengiriman Pompa Station

No	ESP32	
	Latency	Pompa Station
1	148ms	Terkirim
2	157ms	Terkirim
3	194ms	Terkirim
4	170ms	Terkirim
5	165ms	Terkirim
6	113ms	Terkirim
7	133ms	Terkirim
8	121ms	Terkirim
9	142ms	Terkirim
10	151ms	Terkirim

Berdasarkan Tabel 2, Tabel 3, dan Tabel 4, *latency* terkecil dari seluruh data yang telah dikirimkan adalah 105ms. Sedangkan *latency* tertinggi dari seluruh data yang telah dikirimkan adalah 194ms. *Latency* protokol MQTT dalam melakukan publishing ini dipengaruhi oleh kecepatan internet dan besar paket yang dikirimkan. Pada kasus ini besar dari masing-masing paket yang dikirimkan melalui protokol MQTT tidak

DOI: https://doi.org/10.35457/antivirus.v15i2.1611

lebih dari 10 *bytes*. Sistem deteksi banjir kriman yang menggunakan MQTT ini mentolerir *latency* pada pengiriman data menggunakan protokol MQTT tidak lebih dari 500ms. *Latency* pada protokol MQTT juga akan mempengaruhi keterbacaan data pada situs web yang digunakan untuk *monitoring* pada sistem deteksi banjir kiriman. *Latency* yang terlalu tinggi dapat mengakibatkan data sulit terbaca pada situs web yang digunakan untuk *monitoring* sistem deteksi banjir kiriman. Selain itu *latency* ini juga dapat menyebabkan respon pada stasiun pompa air menjadi lebih lambat karena stasiun pompa air bergantung sepenuhnya kepada protokol MQTT untuk beroperasi.

2) *Pengujian Protokol Eddystone URL*

Pengujian protokol *Eddystone* URL dilakukan dengan sepuluh kali uji dengan setiap kali uji jarak akan bertambah satu meter dari lokasi tempat *beacon Eddystone* URL ditempatkan. Ketika pengguna berhasil mendapatkan URL melalui *Eddystone* URL, maka pengujian dikatakan berhasil. Pengujian jarak dilakukan untuk mengetahui tingkat efektivitas protokol *Eddystone* URL yang menggunakan *Bluetooth* dalam mencakup sesuatu yang berada di dekat Pemancar *(Beacon)* tersebut. Pengujian ini akan berimplikasi terhadap pengalaman pengguna dalam menerima data menggunakan *Bluetooth*. Media transmisi *Bluetooth* dapat berjalan efektif jika pengguna berada pada jarak tertentu terhadap *Beacon*. Artinya transmisi menggunakan *Bluetooth* ditujukan untuk memperkenalkan pengguna kepada perangkat IoT yang berada di dekatnya menggunakan *Eddystone* URL. Pengujian ini akan menggunakan aplikasi Android yang khusus dipakai untuk menganalisa paket *Bluetooth* yang dikirimkan. Jarak uji minimal pada pengujian ini adalah 1 meter dan jarak uji maksimal pada pengujian ini adalah 5,5 m. Jarak uji ini didasarkan pada jarak optimal interaksi pengguna dengan perangkat IoT untuk memaksimalkan unsur *discoverability* pada perangkat IoT tersebut. Tabel 5 menunjukkan hasil pengujian dari protokol *Eddystone* URL terhadap jarak pengujian.

Tabel 5. Hasil pengujian pada pengiriman *Eddystone* URL dengan parameter jarak

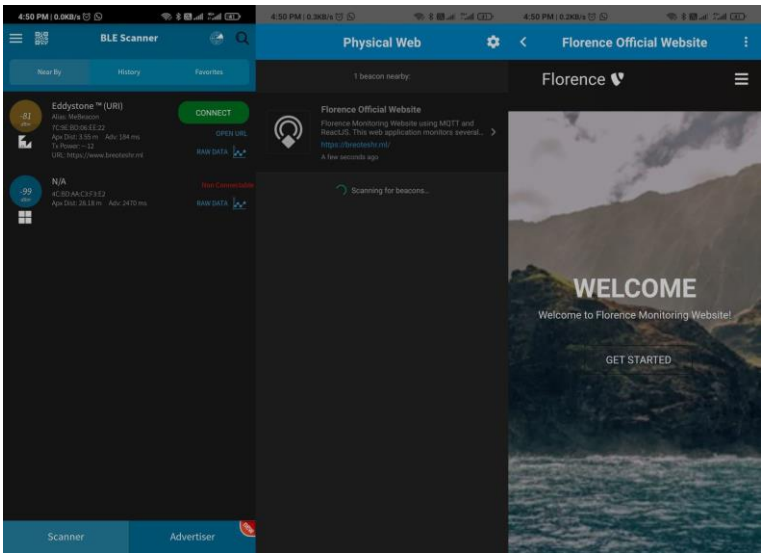
No	Jarak Pengujian (meter)	RSSI (dBm)	*Advertising interval* (ms)	URL Terkirim	Status
1	1	-58	128	https://breoteshr.ml	Berhasil
2	1,5	-64	153	https://breoteshr.ml	Berhasil
3	2	-67	186	https://breoteshr.ml	Berhasil
4	2,5	-65	215	https://breoteshr.ml	Berhasil
5	3	-71	352	https://breoteshr.ml	Berhasil
6	3,5	-71	100	https://breoteshr.ml	Berhasil
7	4	-74	115	https://breoteshr.ml	Berhasil
8	4,5	-76	175	https://breoteshr.ml	Berhasil
9	5	-81	168	https://breoteshr.ml	Berhasil
10	5,5	-80	231	-	Gagal

Berdasarkan Tabel 5, pengujian berhasil dengan 9 kali berhasil dan 1 kali gagal dengan persentase keberhasilan 90%. Hal ini didasarkan oleh jarak antara penguji dengan perangkat *beacon Eddystone* URL yang memiliki jarak yang bervariasi. Hasil pengujian RSSI menunjukkan nilai RSSI yang semakin tinggi pada jarak pengujian yang semakin dekat dengan *Beacon* dengan nilai tertinggi -58dBm dan nilai terendah -80dBm. Hasil pengujian ini juga menunjukkan bahwa *advertising interval* tertinggi pada data uji adalah 352ms. Sedangkan *advertising interval* terendah pada data uji adalah 100ms. Keseluruhan data uji menunjukkan tidak ada *advertising interval* yang melebihi 700ms sehingga sesuai dengan standar komunikasi pada *Physical Web*.

Pengujian protokol *Eddystone* URL juga menguji tentang bagaimana pengguna dapat mengakses aplikasi web melalui aplikasi *Physical Web*. Hasil pengujian menunjukkan bahwa perangkat pengguna mampu mengakses aplikasi web melalui sistem *Physical Web*. Gambar 16 menunjukkan tentang bagaimana pengguna dapat

DOI: https://doi.org/10.35457/antivirus.v15i2.1611

mengakses aplikasi web melalui *Physical Web* untuk melakukan *monitoring* terhadap sistem deteksi banjir kiriman.



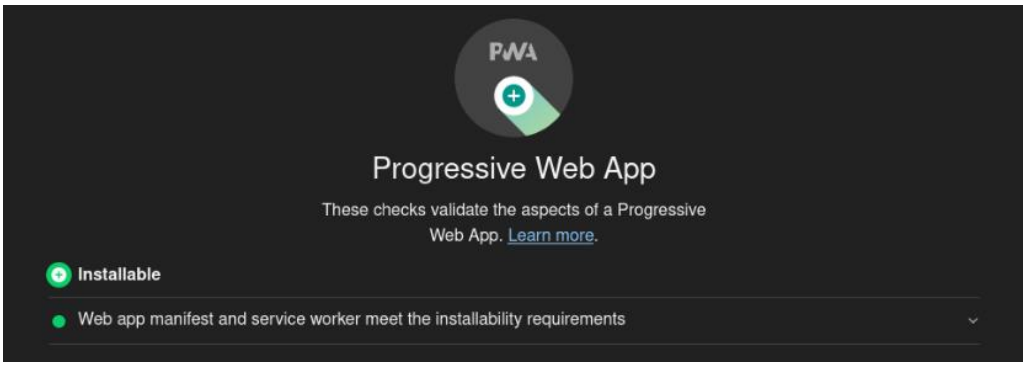
Gambar 16. Dukungan PWA pada aplikasi web PWA

C. Pengujian Progressive Web App

Pengujian *Progressive Web App* (PWA) akan menguji performa aplikasi web yang telah dirancang dengan mengunakan PWA. Dalam hal ini, sebuah aplikasi yang Bernama Google *Lighthouse* akan digunakan sebagai alat uji untuk mengukur kinerja aplikasi web tersebut. Pengujian ini meliputi empat hal, yaitu *Perfomance*, *Accesibility*, *Best Practices*, dan SEO. Tabel 6 menunjukkan hasil pengujian aplikasi web menggunakan PWA dengan menggunakan Google *Lighthouse*. Rata-rata nilai untuk aplikasi dengan tampilan *mobile* adalah 91 dan rata-rata nilai untuk aplikasi dengan tampilan mobile adalah 96,75. Gambar 17 menunjukkan tampilan aplikasi Google Lighthouse yang menujukkan dukungan PWA aplikasi web yang telah dibuat.

Tabel 6. Hasil pengujian pada aplikasi web menggunakan Google *Lighhouse*

No	Aspek yang diuji	Nilai (tampilan *mobile*)	Nilai (tampilan *desktop*)
1	Performance	71	94
2	Accesibility	100	100
3	Best Practices	93	93
4	SEO	100	100
	Rata-rata	91	96,75



Gambar 17. Dukungan PWA pada aplikasi web sistem deteksi banjir kiriman

DOI: https://doi.org/10.35457/antivirus.v15i2.1611

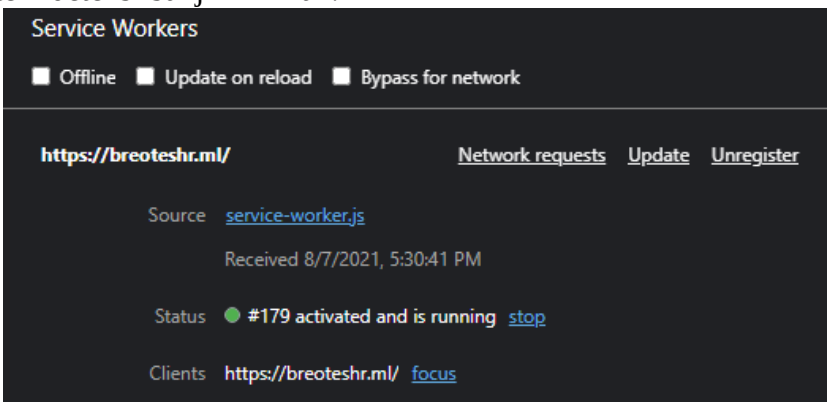
Pengujian dengan menggunakan Google Lighthouse juga menunjukkan bahwa aplikasi web sudah mendukung PWA. Hal ini ditunjukkan dengan adanya *service worker* dan *manifest.json* yang menjadi syarat terbentuknya aplikasi web berbasis PWA. File *manifest.json* adalah sebuah file yang menyediakan identitas pada situs web. Identitas yang dimaksud adalah nama, deskripsi, ikon aplikasi web, dan jalan pintas *(shortcut)* yang akan dipakai oleh situs web. Gambar 18 menunjukkan isi dari *manifest.json* pada aplikasi web yang digunakan untuk *monitoring* sistem deteksi banjir kiriman.

```
{
  "short-name": "Florence",
  "name": "Florence Monitoring Website",
  "icons": [{"src": "/android-icon-36x36.png","sizes": "36x36","type": "image/png","density":
"0.75"},
    {"src":  "/android-icon-192x192.png","sizes":  "512x512",”type":  "image/png",”density":
"4.0"}],
  "display":  "standalone","theme_color":  "#000000","background_color":  "#303030","scope":
"/",
  "shortcuts": [
    {"name": "About Florence Monitoring Website","short_name": "About Us","description": "View
About Florence Monitoring Website","url": "/services","icons": [
        {"src": "/android-icon-192x192.png","sizes": "192x192"}]
    },
    {"name": "Florence Monitoring Dasboard","short_name": "Dashboard","description": "View
Florence Monitoring Dashboard","url": "/","icons": [{"src": "/android-icon-192x192.png",
          "sizes": "192x192"}]}],
"description": "Florence Monitoring Website using MQTT and ReactJS. This web application
monitors several sensors and output state with MQTT."
}
```

Gambar 18. Deskripsi *manifest.json* pada aplikasi web sistem deteksi banjir kiriman.

Berdasarkan Gambar 18, pada file *manifest.json* menunjukkan nama dari situs web *monitoring* sistem deteksi banjir kirman, deskripsi, dan beberapa jalan pintas yang digunakan untuk mengakses situs web tersebut. Jalan pintas ini dapat diakses melalui *browser* seperti Google Chrome pada perangkat Android. File ini juga mendefinisikan ikon yang akan dipakai oleh aplikasi web sebagai ikon pada *title bar*.

Service worker juga berperan penting dalam implementasi *Progressive Web App* pada aplikasi web sistem deteksi banjir kiriman. *Service worker* memungkinkan aplikasi web untuk mengaktifkan *offline caching* sehingga aplikasi web dapat berjalan secara *offline*. Gambar 19 menunjukkan *service worker* yang telah aktif pada aplikasi web sistem deteksi banjir kiriman.



Gambar 19. *Service worker* pada aplikasi web sistem deteksi banjir kiriman.

DOI: https://doi.org/10.35457/antivirus.v15i2.1611

V. KESIMPULAN

Berdasarkan pengujian yang dilakukan sesuai dengan metode penelitian yang telah dilakukan dapat ditarik kesimpukan bahwa protokol *Eddystone* URL cocok digunakan sebagai unsur *discoverability* pada sistem deteksi banjir kiriman dengan menggunakan konsep *Web Of Things*. Perancangan aplikasi web dengan penggunakan *Progressive Web App* menujukkan hasil performa yang cukup baik pada aplikasi web *desktop* maupun aplikasi web *mobile*. Berikut ini adalah rangkuman kesimpulan dari hasil pengujian terebut:

1. Pengujian sensor ultrasonik menunjukkan bahwa sensor ultrasonik mampu digunakan sebagai media untuk mengukur kedalaman sebenarnya pada sungai dengan rata-rata *error* sebesar 3,5%. Sedangkan pengujian *water level* sensor menunjukkan bahwa debit air yang mengalir atau arus dapat dideteksi dengan memanfaatkan *water level* sensor, debit air yang dideteksi oleh *water level* sensor dapat digunakan sebagai perkiraan penambahan debit air yang ada pada hilir dengan rata-rata *error* sebesar 2,5%.
2. Penggunaan MQTT pada sistem deteksi banjir kiriman menunjukkan bahwa protokol MQTT mampu digunakan sebagai media komunikasi antar stasiun dengan masing-masing data uji pengiriman mempunyai *latency* yang kurang dari 200 ms.
3. Pemakaian *Eddystone* URL sebagai protokol *discoverability* pada sistem deteksi banjir menunjukkan respon sistem yang cukup baik dengan keberhasilan sebesar 90% pada sepuluh kali uji coba pada protokol *Eddystone* URL.
4. Pengujian aplikasi web dengan Google *Lighthouse* menunjukkan nilai rata-rata sebanyak 91 untuk aplikasi web *mobile* dan nilai rata-rata sebanyak 96,75 untuk aplikasi web *desktop*. Selain itu, pengujian aplikasi web pada Google Lighthouse juga menunjukkan kompabilitas terhadap *Progressive Web App*.

DAFTAR PUSTAKA

[1] S. M. Mardikaningsih, C. Muryani, dan S. Nugraha, "Studi Kerentanan dan Arahan Mitigasi Bencana Banjir di Kecamatan Puring Kabupaten Kebumen Tahun 2016," *J. Geo Eco*, vol. 3, no. 2, hal. 157–163, 2017.

[2] D. Danang, S. Suwardi, dan I. A. Hidayat, "Mitigasi Bencana Banjir dengan Sistem Informasi *Monitoring* dan Peringatan Dini Bencana menggunakan Microcontroller Arduino Berbasis IoT," *Teknik*, vol. 40, no. 1, hal. 55, 2019, doi: 10.14710/teknik.v40i1.23342.

[3] A. Prasetyo dan M. B. Setyawan, "Purwarupa Internet of Things Sistem Kewaspadaan Banjir Dengan Kendali Raspberry Pi," *Netw. Eng. Res. Oper.*, vol. 3, no. 3, hal. 201–205, 2018, [Daring]. Tersedia pada: http://nero.trunojoyo.ac.id/index.php/nero/article/view/97.

[4] R. B. Sagita dan A. Prapanca, "Rancang Bangun Prototype Sistem *Monitoring* Level Air Untuk Mendeteksi Banjir Berbasis Mikrokontroler Arduino Dan Visual Basic.Net," *J. Manaj. Inform.*, vol. 8, no. 2, hal. 98–104, 2018.

[5] J. P. Nainggolan, M. E. I. Najoan, dan S. D. S. Karouw, "Pengembangan Sistem Informasi Peringatan Dini Banjir Di Kota Manado Berbasis Internet of Things," *J. Tek. Inform.*, vol. 15, no. 1, hal. 65–74, 2020, doi: 10.35793/jti.15.1.2020.29064.

[6] M. F. Habibi, "Rancang Bangun Sistem *Monitoring* Deteksi Dini Untuk Kawasan Rawan Banjir Berbasis Arduino," *J. Mhs. Tek. Inform.*, vol. 2, no. 2, hal. 190–195, 2018.

[7] B. GF dan A. RV, "An IoT-Based Solution for Control and *Monitoring* of Additive Manufacturing Processes," *J. Powder Metall. Min.*, vol. 06, no. 01, hal. 1–7, 2017, doi: 10.4172/2168-9806.1000158.

[8] Google, "Eddystone." https://github.com/google/eddystone (diakses Jun 29, 2021).

[9] Google, "Physical Web." https://google.github.io/physical-web/ (diakses Jun 29, 2021).

[10] M. Ruta, S. Ieva, G. Loseto, dan E. Di Sciascio, "From the Physical Web to the Physical Semantic Web: knowledge discovery in the Internet of Things," *Tenth Int. Conf. Mob. Ubiquitous Comput. Syst. Serv. Technol.*, hal. 209–214, 2016.

[11] N. Nurwanto, "Penerapan Progressive Web Application (PWA) pada E-Commerce," *Techno.Com*, vol. 18, no. 3, hal. 227–235, 2019, doi: 10.33633/tc.v18i3.2400.

DOI: https://doi.org/10.35457/antivirus.v15i2.1611