IMPLEMENTASI METODE *PERVASIVE* MENGGUNAKAN PROTOKOL UDP PADA RASPBERRY PI DAN MYRIO

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
Joniar Dimas Wicaksono
NIM: 145150300111106

PROGRAM STUDI TEKNIK KOMPUTER
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

IMPLEMENTASI METODE *PERVASIVE* MENGGUNAKAN PROTOKOL UDP PADA RASPBERRY PI DAN MYRIO

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh :
Joniar Dimas Wicaksono
NIM: 145150300111106

Skripsi ini telah diuji dan dinyatakan lulus pada
01 Agustus 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Wijaya Kurniawan, S.T., M.T.	Mochammad Hannats Hanafi Ichsan, S.ST., M.T.
NIP: 19820125 201504 1 002	NIK: 201405 881229 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D.
NIP: 19710518 200312 1 001

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 pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat 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, 29 Maret 2018

Joniar Dimas Wicaksono

NIM: 145150300111106

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa, karena rahmat-Nya penulis dapat menyelesaikan penelitian dan laporan skripsi ini yang berjudul **IMPLEMENTASI METODE PERVASIVE MENGGUNAKAN PROTOKOL UDP PADA RASPBERRY PI DAN MYRIO.**

Penulis menyadari bahwa dalam penyelesaian penelitian dan penyusunan skripsi ini tidak akan terwujud tanpa adanya bantuan dan dorongan dari berbagai pihak. Oleh karena itu pada kesempatan ini penulis menyampaikan ucapan terima kasih kepada yang terhormat :

1. Kepada kedua Orang Tua, serta keluarga penulis yang telah memberikan motivasi, dukungan, support, doa yang tiada hentinya kepada penulis.
2. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D. selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang.
3. Bapak Heru Nurwarsito, Ir., M.Kom selaku Wakil Dekan Bidang Akademik Fakultas Ilmu Komputer Universitas Brawijaya Malang.
4. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D. selaku Ketua Jurusan Teknik Informatika Universitas Brawijaya Malang.
5. Bapak Sabriansyah Rizqika Akbar, S.T, M.Eng. selaku Ketua Program Studi Teknik Komputer Universitas Brawijaya Malang.
6. Bapak Wijaya Kurniawan, S.T, M.T selaku pembimbing I yang telah memberi arahan untuk menyelesaikan skripsi ini.
7. Bapak Mochammad Hannats Hanafi Ichsan, S.ST, M.T selaku pembimbing II yang telah memberi arahan untuk penyelesaian laporan skripsi ini.
8. Teman-teman penulis yang tidak bisa disebutkan satu-persatu yang telah memberi tekanan, dan memotivasi penulis untuk segera menyelesaikan skripsi.
9. Dan pihak-pihak yang tidak dapat disebutkan penulis.

Akhir kata penulis, penulis menyadari bahwa banyak kekurangan yang dilakukan pada penelitian ini. Penulis berharap kritik dan saran agar penulis menjadi lebih baik lagi. Semoga penelitian ini dapat bermanfaat untuk semua pembaca.

Malang, 29 Maret 2018

Penulis

dwdimas@gmail.com

ABSTRAK

Rumah cerdas atau biasa disebut dengan *smart home* merupakan lingkungan yang di dalamnya terdapat peralatan yang dapat saling berkomunikasi satu sama lain dan dapat dipantau atau dikendalikan secara jarak jauh oleh kita melalui internet. Dengan ini, akan memudahkan manusia untuk mengendalikan dan memantau peralatan yang ada dirumah. Kemudahan ini masih dirasa sulit untuk orang-orang yang masih awam akan teknologi. Karena saat ini, penggunaan peralatan rumah yang dapat menjadi *smart home* masih membutuhkan konfigurasi yang lumayan dirasa sulit. Metode *pervasive* merupakan metode memudahkan manusia untuk tidak perlu repot men-konfigurasi perangkat. Sistem kerja dari metode *pervasive* ini dengan cara menyebarkan informasi dirinya dan menyimpan informasi perangkat lain jika ada yang membalas informasi tersebut. Untuk meningkatkan performa dan efisiensi dari sistem ini, protokol pengiriman yang digunakan yaitu protokol UDP. Dimana protokol ini merupakan protokol yang ringan dan tidak memerlukan memori yang besar. Sistem pada penelitian kali ini diimplementasikan pada perangkat Raspberry Pi 3 dimana sebagai host dan NI myRIO sebagai client. Perangkat ini diprogram oleh LabVIEW dengan cara men-deploy program pada perangkat. Program dibuat sesuai alur sistem yaitu dengan *state machine* dimana pada masing-masing perangkat dapat melakukan discovery agar perangkat saling kenal dan mengetahui informasinya masing-masing tanpa konfigurasi manual. Dari hasil pengujian pada penelitian kali ini, untuk *discovery* perangkat pada bagian *host* yaitu 56,417ms, sedangkan proses *discovery* pada *client* didapatkan hasil yaitu 251,067ms. Sehingga untuk proses *discovery* seluruhnya yaitu 313,417ms. Sedangkan jika *host* mengalami kegagalan, waktu yang dibutuhkan *client* untuk terhubung kembali yaitu 10384,23ms. Sedangkan untuk pengiriman dan penerimaan data antar *host* dan client, untuk pengiriman data sensor membutuhkan waktu 58,263ms, untuk pengendalian LED membutuhkan waktu 5350,926ms, dan untuk *push button* membutuhkan waktu 255,696ms.

Kata kunci: *smart home, pervasive,* protokol UDP, *state machine,* LabVIEW, NI MyRIO, Raspberry Pi 3

ABSTRACT

Smart home environment is an environment in which there are equipment that can communicate with each other and can be monitored or controlled remotely through the internet. With this, it will make it easier for humans to control and monitor the equipment at home. This ease is still difficult for people who still lay in the technology. Because at this time, the use of home appliances that can be a smart home still requires difficult configuration. Pervasive is a method of making it easy for human to not have to bother configure device. The working system of this pervasive method by broadcast information itself and storing information other devices if there is a reply the others information. To improve the performance and efficiency of this system, the protocol will be used is UDP protocol. Where this protocol is a lightweight protocol and does not require large memory. The system in this research is implemented on Raspberry Pi 3 where as host and NI myRIO as client. This device is programmed by LabVIEW by deploying programs on the device. Program will be made according to the system flow that is with the state machine where on each device can do discovery for the devices know each other and know their respective information without manual configuration. From the results of testing in this study, for discovery device on the host that is 56,417ms, while the discovery process on the client obtained the result 251.067ms. So for the whole discovery process that is 313,417ms. Whereas if the host fails, the time it takes the client to reconnect is 10384,23ms. As for sending and receiving data between host and client, for data transmission sensor takes 58.263ms, for LED control takes 5350,926ms, and for push button takes 255,696ms.

Keywords: *smart home, pervasive,* UDP protocol, *state machine,* LabVIEW, NI MyRIO, Raspberry Pi 3

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Rumah cerdas atau biasa disebut dengan *smart home* merupakan lingkungan yang di dalamnya terdapat peralatan yang dapat saling berkomunikasi satu sama lain dan dapat dipantau atau dikendalikan secara jarak jauh oleh kita melalui internet. Tujuannya untuk memudahkan kita dalam mengawasi dan mengendalikan berbagai peralatan yang ada di rumah seperti peralatan listrik, suhu ruangan, keamanan rumah, kamera pengawas, dan sebagainya. Sehingga di masa yang akan datang, rumah cerdas merupakan sebuah pilihan yang salah satunya dapat memudahkan manusia dengan bantuan teknologi.

Saat ini telah dikembangkan metode *pervasive computing* untuk mempermudah kita menghubungkan antar perangkat secara otomatis tanpa konfigurasi manual oleh kita. Dimana *pervasive computing* merupakan suatu konsep di dalam *software engineering* dan *computer science* bahwa komputasi dapat diimplementasikan di mana saja. Lingkungan yang didukung dengan teknologi ini diciptakan untuk mempermudah pekerjaan dalam mencapai tujuan dan kebutuhan kita. Integrasi dari teknologi ini memungkinkan kita untuk dapat menikmati setiap layanan teknologi perangkat yang saling terhubung antara satu sama lain. Setiap pekerjaan atau proses yang dibutuhkan akan menjadi lebih mudah, cepat dan efisien karena pekerjaan-pekerjaan tersebut sudah diproses secara otomatis (Zulfy, et al., 2018). Otomatis yang dimaksudkan yaitu perangkat dapat saling terhubung dan berkomunikasi tanpa melakukan konfigurasi di setiap perangkatnya.

Protokol jaringan yang biasa digunakan untuk menghubungkan antar perangkat yaitu *Transmission Control Protocol* (TCP) dan *User Datagram Protocol* (UDP). Kedua protokol ini mempunyai kelebihan dan kekurangan masing-masing tergantung tujuan yang ingin dicapai, dimana kelebihan TCP yaitu *reliable* pada datanya, sedangkan UDP merupakan protokol yang ringan *(lightweight)* dimana dapat menghemat sumber daya memori dan prosesor. Pada lingkungan rumah cerdas, data yang dikirimkan tidak terlalu besar sehingga cocok jika menggunakan protokol UDP. Jika menggunakan TCP, dibutuhkan adanya proses *three-way-handshaking* dimana keadaan ini dapat menyebabkan kepadatan *traffic* dan proses akan terjadi lebih lama. Kelebihan lainnya menggunakan UDP yaitu tidak perlu menunggu penerimaan *(acknowledgement)* atau menyimpan data dalam memori sampai data tersebut diterima. Ini berarti, aplikasi UDP tidak diperlambat oleh proses penerimaan dan memori dapat dibebaskan lebih cepat (Kurniawan, et al., 2017). Serta protokol UDP dapat mengirimkan data secara *broadcast* dibandingkan TCP secara *direct*.

NI MyRIO merupakan sebuah kontroler yang dapat terkoneksi dengan banyak perangkat keras. Sistem operasi pada NI MyRIO menggunakan sistem operasi *real-time* berbasis linux. Pada kontroler MyRIO telah tersedia sensor *accelerometer* untuk menentukan posisi akselerasi kontroler dan *WiFi* untuk menghubungkan

antar perangkat lainnya (Ryzkiansyah, et al., 2017). Sedangkan Raspberry Pi 3 merupakan sebuah mini komputer dengan berbasis sistem operasi raspbian. Pada mini komputer Raspberry Pi 3 juga terdapat *WiFi* untuk menghubungkan antar perangkat.

Perangkat NI myRIO dan Raspberry Pi 3 merupakan perangkat *portable* yang dapat diletakkan dimana saja. Kelebihan dari kedua perangkat ini diantaranya NI myRIO yang sudah memiliki sensor *accelerometer* dan kemudahan untuk menambahkan sensor eksternal lainnya sesuai kebutuhan, sedangkan Raspberry Pi 3 merupakan mini komputer yang tidak memerlukan daya listrik yang besar. Pada pengujian kali ini, NI MyRIO akan bertindak sebagai node sensor dan Raspberry Pi 3 sebagai perangkat perantara yang berfungsi untuk menyimpan informasi node sensor yang terdeteksi di sekitarnya dan dapat memantau serta mengendalikan node sensor.

Berdasarkan penjelasan diatas, dibutuhkan adanya teknologi yang memudahkan manusia untuk mengkonfigurasi antar perangkat agar saling mengenal satu sama lain tanpa konfigurasi manual oleh manusia. Maka penulis terinspirasi untuk melakukan penelitian metode *pervasive* agar dapat diimplementasikan pada perangkat. Dimana perangkat tersebut yaitu Raspberry Pi 3 dapat mengenali perangkat node sensor NI MyRIO yang aktif disekitarnya. Dimana seluruh perangkat ini terhubung melalui *WiFi* di jaringan lokal menggunakan protokol komunikasi UDP. Pemrograman yang digunakan adalah *dataflow programming* yaitu LabVIEW dengan metode *state machine* yang nantinya diimplementasikan pada perangkat Raspberry Pi 3 dan NI MyRIO. Selain itu, dibutuhkan adanya *Library LINX Labview* agar kode program LabVIEW dapat diimplementasikan pada perangkat Raspberry Pi 3.

1.2 Rumusan masalah

Berdasarkan penjelasan pada latar belakang, maka dapat dirumuskan rumusan masalah sebagai berikut:

1. Bagaimana cara merancang sistem yang dapat mengenali perangkat sekitarnya tanpa konfigurasi dari pengguna?
2. Bagaimana cara mengimplementasikan sistem tersebut pada NI MyRIO dan Raspberry Pi 3 sebagai perangkat?
3. Bagaimana analisis hasil sistem tersebut yang diimplementasikan pada NI MyRIO dan Raspberry Pi 3 sebagai perangkat?

1.3 Tujuan

Adapun maksud dan tujuan dari penelitian ini adalah sebagai berikut:

1. Sistem dapat mengenali perangkat sekitarnya tanpa konfigurasi dari pengguna.
2. Sistem dapat diimplementasikan pada NI MyRIO dan Raspberry Pi 3 sebagai perangkat.

3. Mendapatkan hasil analisis sistem tersebut yang diimplementasikan pada NI MyRIO dan Raspberry Pi 3 sebagai perangkat.

1.4 Manfaat

Digarapkan manfaat dari penelitian ini bagi penulis yaitu dapat memahami komputasi *pervasive* menggunakan protokol UDP yang diimplementasikan pada perangkat NI MyRIO dan Raspberry Pi 3 menggunakan bahasa pemrograman *dataflow programming* yaitu LabVIEW. Serta memberi wawasan baru untuk pembaca tentang adanya komputasi *pervasive* menggunakan protokol UDP.

1.5 Batasan masalah

Adapun batasan masalah pada penelitian kali ini yaitu Raspberry Pi 3 atau bisa disebut sebagai *host* dimana perangkat perantara yang berfungsi untuk menyimpan informasi node sensor yang terdeteksi di sekitarnya dan dapat memantau serta mengendalikan node sensor dan NI MyRIO atau bisa disebut sebagai *client* sebagai node sensor. Sensor yang digunakan adalah sensor *accelerometer* yang telah tersedia pada NI MyRIO.

1.6 Sistematika pembahasan

Sistematika pembahasan dalam skripsi ini adalah sebagai berikut:

BAB 1 PENDAHULUAN

Menjelaskan latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika pembahasan dari "Implementasi Protokol UDP *Pervasive* Menggunakan Raspberry Pi Dan MyRIO Pada Lingkungan *Smarthome*"

BAB 2 LANDASAN KEPUSTAKAAN

Menjelaskan tinjauan pustakan dan dasar teori yang terkait dengan penelitian, dimana tinjauan pustaka akan menjelaskan penelitian yang terkait diantaranya yang berhubungan dengan *Smart Home* dan *Pervasive UDP* menggunakan LabVIEW. Sedangkan dasar teori akan menjelaskan pengertian-pengertian umum dari komponen penelitian yang akan dilakukan.

BAB 3 METODOLOGI

Bab ini membahas tentang langkah-langkah dalam melakukan penelitian ini, diantaranya melakukan studi literatur yang terkait, analisis kebutuhan sistem, perancangan sistem sesuai tujuan penelitian, implementasi sistem pada perangkat, pengujian dan analisis sistem, dan kesimpulan dari hasil pengujian penelitian.

BAB 4 REKAYASA KEBUTUHAN

Menguraikan secara rinci terkait gambaran umum sistem dimana akan menjelaskan keseluruhan gambaran terkait sistem yang akan dirancang,

analisis kebutuhan sistem yang berisi kebutuhan fungsional dan non-fungsional dari sebuah sistem, dan batasan desain sistem yang menjelaskan batasan-batasan perancangan sistem agar tidak terlalu luas.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Membahas mengenai proses perancangan sistem yang berupa alur *state machine* hingga implementasi kode program pada perangkat Raspberry Pi 3 dan NI MyRIO.

BAB 6 PENGUJIAN DAN ANALISA

Membahas skenario pengujian yang dilakukan terhadap sistem berdasarkan analisis kebutuhan sistem, perancangan sistem dan implementasi serta melakukan analisa terhadap hasil pengujian yang telah dilakukan.

BAB 7 PENUTUP

Mendapatkan dan menjelaskan kesimpulan dari hasil penelitian yang dilakukan berdasarkan pengujian dan analisa yang dilakukan sebelumnya, dan menyampaikan saran yang dapat digunakan untuk pengembangan sistem serupa kedepannya.

BAB 2 LANDASAN KEPUSTAKAAN

Landasan kepustakaan berisi uraian dan pembahasan tentang teori, konsep, model, metode, atau sistem dari literatur ilmiah, yang berkaitan dengan tema, masalah, atau pertanyaan penelitian. Dalam landasan kepustakaan terdapat landasan teori dari berbagai sumber pustaka yang terkait dengan teori dan metode yang digunakan dalam penelitian.

2.1 Tinjauan Pustaka

Tinjauan pustaka membahas dari beberapa penelitian sebelumnya yang terkait dengan penelitian yang diusulkan. Pada penelitian ini tinjauan pustaka diambil dari beberapa penelitian yang pernah dilakukan. Tujuan dari tinjauan pustaka adalah untuk mengkaji hasil penelitian sebelumnya dan dijadikan sebagai dasar dalam pelaksanaan penelitian.

Penelitian pertama dilakukan oleh Hamed (2012) yang berjudul *Design & Implementation of Smart House Control Using LabVIEW.* Penelitian ini menyajikan rumah cerdas yang dapat dikendalikan oleh LabVIEW dimana aplikasi ini sebagai pengendali utama dari keseluruhan sistem. Sistem ini dibagi menjadi lima bagian, bagian-bagian ini terhubung dengan aplikasi LabVIEW yang sebagai pengendali utama dari keseluruhan sistem. Sistem rumah cerdas ini memiliki dua antarmuka, antarmuka komputer dan antarmuka pengendali. Komputer yang terdapat aplikasi LabVIEW ini adalah unit pengendali utama untuk seluruh sistem di rumah. Komputer akan menerima data dari node sensor, memproses informasi dan memperbaruii data dari berbagai sistem, dan mengirimkan sinyal kendali pada sistem lainnya. Jadi, fungsi dari LabVIEW ini memiliki kemampuan untuk memantau operasi pada sistem kepada pengguna untuk memberikan informasi apa saja perubahan yang terjadi pada sistem. Pengguna dapat mengendalikan berbagai kemampuan sistemnya, dan memilih kebutuhan yang baik untuk sistem. Jadi antarmuka LabVIEW pada rumah cerdas ini, antarmuka pengendali dapat mengendalikan beberapa sistem yang ada dirumah, dan terhubung dengan aplikasi LabVIEW untuk pengaplikasian lainnya (Hamed, 2012).

Penelitian selanjutnya dilakukan oleh Kurniawan dkk. (2013) yang berjudul *Lightweight UDP Pervasive Protocol in Smart Home Environment Based on LabVIEW.* Pada penelitian ini merupakan simulasi *pervasive computing* pada rumah cerdas menggunakan protokol UDP berbasis aplikasi LabVIEW. Peneliti berhasil membuat sistem ini sebagai prototipe, dan sistem ini bisa disebut sebagai *lightweight UDP* pada LabVIEW. Sistem ini memiliki dua perangkat yaitu *client dan host.* Dimana penelitian ini *client* dapat terhubung dengan *host* secara *pervasive*. Penelitian ini menitikberatkan pada layanan *discovery* yang di lakukan oleh *host,* dimana hasilnya *client* dapat berhasil teridentifikasi pada *host. Host* dapat mengendalikan segala layanan yang disediakan oleh *client*. Simulasi ini dilakukan pada dua buah komputer dengan asumsi salah satu komputer bertindak sebagai *client* dan *host* pada perangkat rumah (Kurniawan, et al., 2017).

2.2 Dasar Teori

2.2.1 User Datagram Protocol (UDP)

Protokol UDP merupakan protokol transport yang ringan, dan menyediakan minimal pelayanan. UDP merupakan *conectionless*, dimana tidak adanya proses *handshaking* sebelum kedua proses memulai untuk berkomunikasi. UDP menyediakan layanan pengiriman data yang *unreliable*, dimana dalam proses pengiriman sebuah pesan melalui soket UDP, UDP tidak menjamin bahwa pesan akan tersampaikan pada penerima. Kemudian, pesan yang tersampaikan bisa dimungkinkan rusak. UDP tidak menyertakan mekanisme kontrol kemacetan, jadi pada sisi pengirim dapat memompa data ke lapisan di bawah (*layer network*) pada tingkat yang diinginkan (Kurose & Ross, 2013).

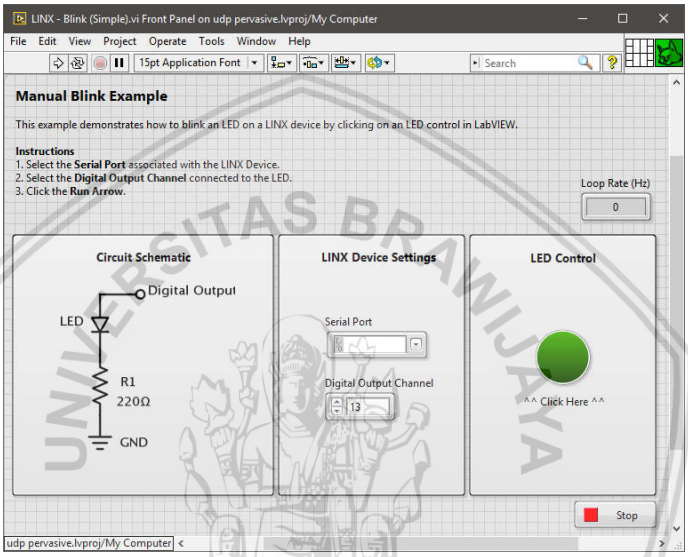
2.2.2 Pervasive Computing

Komputasi *pervasive* menawarkan cakupan yang luas untuk cakupan komputasi, mobilitas dan kegunaan. Sejatinya, pervasive model ini harus bisa mengenali dan mengakomodasi setiap perangkat yang terhubung. Misalkan perangkat yang mengumpulkan informasi dari pembacaan sensor, transfer data dan mengambil keputusan sendiri ini dapat dikategorikan sebagai perangkat pervasive. Dalam jaringan komputer, protokol *pervasive* dapat mengenali perangkat lain dan dapat menkonfigurasi layanan komunikasi secara otomatis.

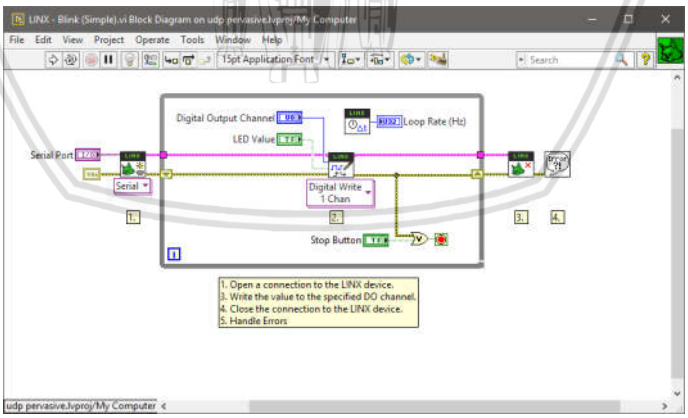
Komputasi pervasif, juga dikenal sebagai *Internet of Things*, atau *Ubiquitous Computing*. *Ubiquitous computing* menggambarkan tren yang muncul dari integrasi komputasi yang mulus ke dalam dunia fisik sehari-hari. Contoh sistem komputasi yang meliputi: *self-driving cars*, *smart home*, sistem navigasi untuk orang disabilitas, dan sistem pemantauan lingkungan. Fitur penting dari sistem ini adalah kesadaran konteksnya, yang berarti bahwa mereka menentukan keadaan (waktu, lokasi, suhu lingkungan, emosi, aktivitas seismik, dll.). Untuk mengekstraksi konteks ini, sistem komputasi harus dapat memproses sinyal dari lingkungan sekitarnya dan kemudian memprosesnya. Konsekuensinya, berdasarkan konteks yang disimpulkan dan model proses internal, sistem komputasi yang meresap mengambil keputusan cerdas dan bertindak pada lingkungan yang sama. Konteks menyimpulkan, alasan, dan pengambilan keputusan adalah tugas agen perangkat lunak, atau pengendali Kontroler. Dalam banyak kasus, lebih banyak sistem komputer bekerja bersama di belakang layar, bertukar informasi dengan cara transparan, untuk memberikan layanan kepada pengguna. Untuk meringkas, sistem komputasi menyeluruh meliputi sensor, aktuator, agen perangkat lunak (pengendali), dan modul komunikasi. Sifat intinya adalah konteks-kesadaran, interaksi manusia-komputer implisit, jaringan tanpa batas, otonomi, dan kecerdasan. Namun, cita-cita tertinggi dalam membuat komputasi yang tertanam dalam kehidupan kita secara alami tidak dapat dicapai hanya dengan memecahkan masalah teknis. Penerimaan dan kepercayaan komputasi pervasif juga membutuhkan solusi untuk tantangan hukum, etika, dan kegunaan (Silvis-Cividjian, 2017).

2.2.3 LabVIEW

LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench), merupakan sebuah software komputer yang berfungsi sebagai pemroses dan visualisasi data dalam bidang akuisisi data, yang dikembangkan oleh National Instruments. LabVIEW merupakan bahasa pemrograman yang berbasiskan blok diagram, yang dapat disebut *dataflow programming*, dimana eksekusi dari sebuah instruksi mengikuti dari alur blok diagram. Tampilan utama pada LabVIEW terdapat dua jendela yang pertama *Front Panel* yaitu antarmuka interaksi dengan pengguna pada Gambar 2.1 dan *Block Diagram* dimana tempat kita akan memprogram yang ada pada Gambar 2.2.



Gambar 2.1 *Front Panel* LabVIEW

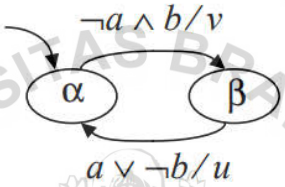


Gambar 2.2 *Block Diagram* LabVIEW

Program dieksekusi mengikuti jalur konektor yang terhubung antara node dengan node. Masing-masing fungsi disimpan sebagai *virtual instrument* (VI) dimana memiliki tiga komponen: *front panel, block diagram,* dan *connector pane* yang merupakan antarmuka seperti VI tetapi tertanam yang disebut sub-VI (Elliott, et al., 2017).

2.2.4 Finite State Machine

Finite State Machine atau bisa disebut FSM digunakan sebagai permodelan perilaku sistem dari berbagai macam pengaplikasian pada *engineering* dan *scientific*. *State* merupakan keadaan suatu sistem yang didefinisikan sebagai kondisi pada suatu titik tertentu; *state machine* merupakan sistem yang keluarannya tidak hanya bergantung pada masukan, tetapi juga keadaan sekarang *(current state)* dari sistem tersebut. Keadaan dari sebuah sistem merupakan sebuah kesimpulan dari seluruh kebutuhan sistem yang diketahui dari masukan sebelumnya sehingga menghasilkan sebuah keluaran. Variabel *state* dapat dipresentasikan sebagai $s \in \sum$, dimana $\sum$ merupakan keseluruhan kemungkinan dari sistem. Finite state machine (FSM) adalah state machine dengan $\sum$ set yang terbatas. pada finite state machine, keadaan sistem dimodelkan dari beberapa set keadaan dan aturan yang mengatur transisi diantara keadaan-keadaan tersebut (Ptolemaeus, 2014).



Gambar 2.3 FSM Dasar

FSM terdiri dari sekumpulan peristiwa input, seperangkat peristiwa output, seperangkat negara bagian, keadaan awal dan serangkaian transisi. Pertimbangkan contoh FSM, ditunjukkan pada Gambar 2.3, dengan peristiwa masukan {a, b} dan peristiwa keluaran {u, v}, di mana suatu peristiwa adalah variabel bernama yang hadir atau tidak ada. Setiap node elips mewakili suatu keadaan dan setiap busur mewakili suatu transisi. Busur tanpa status sumber menunjuk ke keadaan awal, yaitu menyatakan α. Setiap transisi menghubungkan negara sumber dengan negara tujuan, dan diberi label oleh "penjaga / tindakan" atau "penjaga" (yaitu tindakan dihilangkan). Penjaga adalah ekspresi boolean atas peristiwa input. Evaluasi suatu peristiwa adalah benar atau salah ketika acara itu hadir atau tidak ada. Operator ¬, ∨ dan ∧ dalam penjaga sesuai dengan operator boolean tidak, atau dan dan, masing-masing. Suatu tindakan daftar subset dari peristiwa output.

Dalam satu reaksi dari FSM, subset dari peristiwa input hadir. Satu transisi dipicu saat penjaga benar di bawah peristiwa masukan saat ini. FSM pergi ke negara tujuan dari transisi yang dipicu, dan memancarkan setiap peristiwa output dalam aksi transisi yang dipicu, membuat peristiwa keluaran ini hadir. Jika tindakan tersebut dihilangkan, itu berarti tidak ada peristiwa keluaran yang diemisikan. Suatu tindakan hanya mencantumkan peristiwa keluaran yang akan dipancarkan, dan dengan demikian semua peristiwa keluaran lainnya tidak ada (Lee & Lee, 1998).

2.2.5 NI MyRIO

MyRIO merupakan sebuah *real-time embedded board* yang dibuat oleh National Instruments. Perangkat ini digunakan untuk mengembangkan aplikasi yang memanfaatkan *FPGA (Field-Programmable Gate Array)* dan *microprosessor*. NI MyRIO memiliki fitur I/O pada dua sisi dari perangkat dalam bentuk konektor MXP dan MSP. Itu termasuk *input* analog, *output* analog, I/O digital, LED, *push button*, *accelerometer onboard,* Xilinx FPGA, dan prosesor dual-core ARM Cortex-A9. Beberapa model lainnya memiliki WiFi. Perangkat NI MyRIO dapat diprogram dengan LabVIEW atau C (Instruments, 2018). Gambar 2.4 merupakan bentuk fisik dari NI MyRIO.

Gambar 2.4 National Instrument MyRIO

Sumber: http://www.ni.com

2.2.6 Raspberry Pi 3

Raspberry Pi 3 merupakan sebuah *single board computer* dimana fitur yang dimilikinya komplit dan dapat digunakan seperti *personal computer* pada umumnya, tetapi dengan ukuran kecil. Raspberry Pi dapat ditenagai hanya dengan *charger* handphone, berjalan dengan sistem operasi Linux, dan dapat menggunakan berbagai macam bahasa pemrograman. Gambar 2.5 merupakan bentuk fisik dari Raspberry Pi 3 Model B.

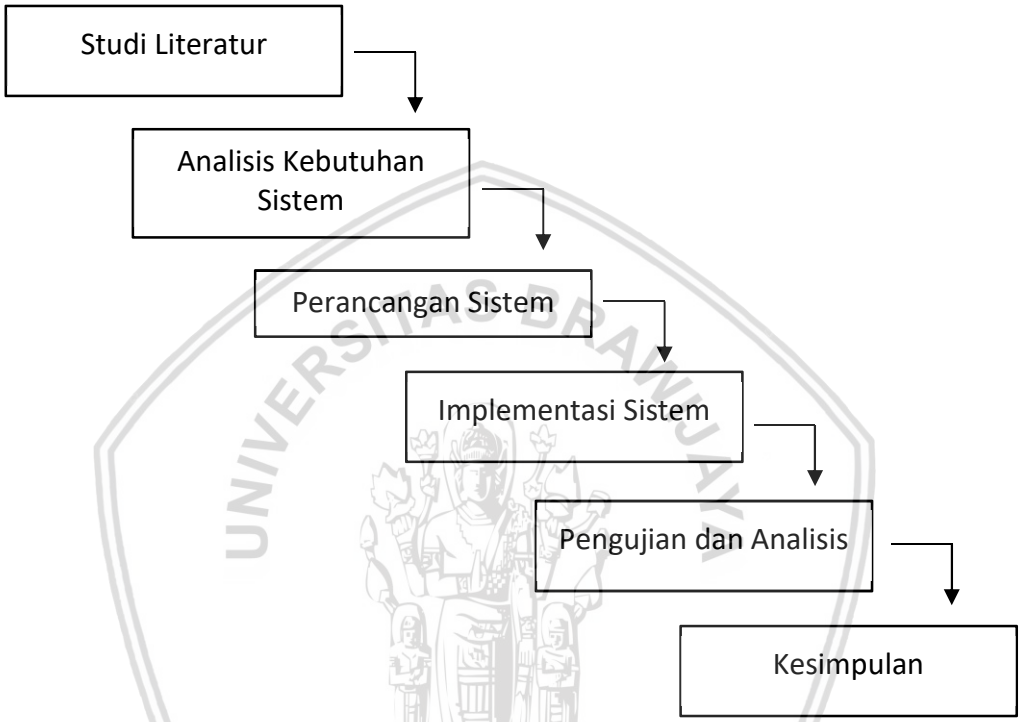
Gambar 2.5 Raspberry Pi 3

Sumber: https://www.raspberrypi.org

Spesifikasi yang ditawarkan oleh Raspberry Pi 3 Model B yaitu ditenagai oleh Prosesor Quad-core 1.2GHz Broadcom BCM2837 64bit CPU, dengan RAM 1GB, dimana terdapat memiliki beberapa fitur diantaranya: LAN dan *Bluetooth Low Energy* (BLE), 100 Base Ethernet, 40-pin extended GPIO, 4 USB port 2.0, 4 pole stereo output dan video port komposit, full size HDMI, port kamera CSI, port display DSI, dan slot Micro SD (Foundation, 2018).

BAB 3 METODOLOGI

Penelitian diawali dengan melakukan studi literatur terkait kajian pustaka dan dasar teori. Penelitian ini bersifat implementatif. Penentuan alur metode penelitian sebagai langkah yang ditempuh untuk menyelesaikan penelitian secara sistematis. Alur metode penelitian yang dilakukan dapat dilihat diagram alir pada Gambar 3.1.



Gambar 3.1 Diagram Alir Alur Penelitian

3.1 Studi Literatur

Pada perancangan dan implementasi penelitian ini, perlu diadakan studi literatur. Literatur digunakan sebagai teori penguat dan landasan dasar dalam penelitian. Teori pendukung tersebut didapat dari buku, jurnal, paper dan internet. Berikut merupakan dasar teori yang akan dipelajari sebagai bahan studi:

1. LabVIEW

   Mempelajari bahasa pemrograman yang dibawakan oleh LabVIEW agar dapat membuat dan mengimplementasikan kode program pada perangkat NI MyRIO dan Raspberry Pi 3.

2. State Machine

   Mempelajari alur dari sistem yang diinginkan, sehingga dapat dibuat alur menggunakan state machine.

3. NI MyRIO

   Mempelajari fitur yang dibawakan oleh perangkat NI MyRIO agar dapat berfungsi dengan tujuan penelitian.

4. Raspberry Pi 3

   Mempelajari fitur yang dibawakan oleh perangkat Raspberry Pi 3 agar dapat berfungsi dengan tujuan penelitian.

5. Protokol UDP

   Mempelajari protokol UDP yang dibawakan oleh LabVIEW agar dapat berfungsi dengan tujuan penelitian

6. Library LINX LabVIEW

   Mempelajari *library* LINX LabVIEW agar dapat di gunakan dan di implementasikan pada perangkat Raspberry Pi 3.

3.2 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem bertujuan untuk menganalisa semua kebutuhan yang diperlukan oleh sistem yang akan dibangun. Analisis kebutuhan sistem dilakukan dengan mengidentifikasi kebutuhan sistem. Analisis kebutuhan tersebut, meliputi kebutuhan perangkat lunak dan kebutuhan perangkat keras. Berikut merupakan kebutuhan perangkat lunak yang dibutuhkan pada penelitian kali ini:

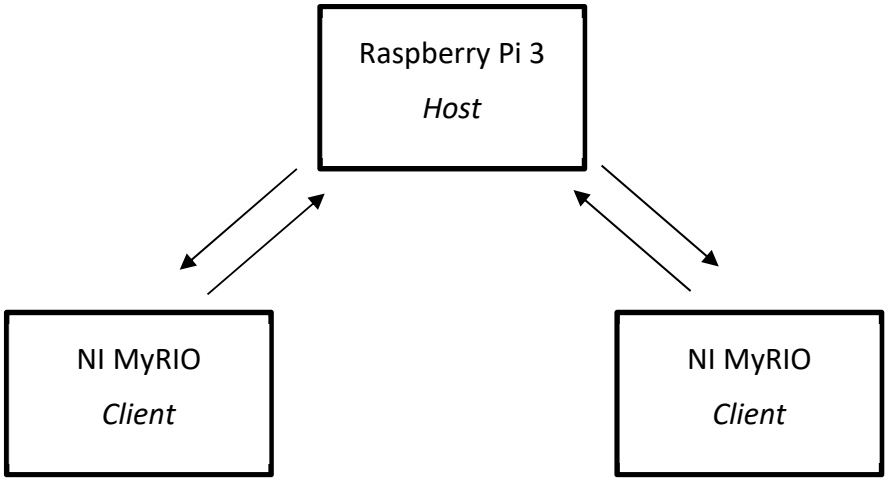
1. LabVIEW
2. Library Linx LabVIEW

Dan berikut merupakan kebutuhan perangkat keras yang dibutuhkan pada penelitian kali ini:

1. NI MyRIO
2. Raspberry Pi 3

3.3 Perancangan Sistem

Perancangan sistem dilakukan jika tahap analisis kebutuhan sistem terpenuhi. Pada kali ini, arsitektur sistem dapat dilihat pada Gambar 3.2. Pada gambar tersebut terdapat tiga buah perangkat, perangkat Raspberry Pi 3 dan Perangkat NI MyRIO. Pada arsitektur tersebut perangkat Raspberry Pi 3 dapat disebut sebagai *host* dan NI MyRIO dapat disebut *client*. *Host* dan client akan saling terhubung melalui *WiFi* pada jaringan hotspot lokal.



Gambar 3.2 Arsitektur Sistem

3.4 Implementasi Sistem

Setelah perancangan sistem terpenuhi, selanjutnya dilakukan implementasi sistem dimana kode program akan diimplementasikan pada perangkat NI MyRIO dan Raspberry Pi 3. Pada *client* fitur yang diimplementasikan diantaranya fitur pembacaan sensor accelerometer, penggunaan LED, pengiriman data menuju *host*, proses pencarian dan penyimpanan informasi dari *host*, proses menghubungkan ulang jika terjadi putus koneksi. Sedangkan fitur pada perangkat *host* yaitu proses pencarian dan penyimpanan informasi yang didapat dari *client*, proses terhubung dan memutus koneksi dengan *client*, pemantauan dan pengendalian *client*.

3.5 Pengujian dan Analisis

Pada tahap ini akan dilakukan pengujian dan analisis hasil dari sistem yang telah dibuat. Pengujian dan analisis dilakukan agar sistem dapat berjalan dengan tujuan penelitian yang diharapkan. Pengujian dan analisis yang dilakukan akan berdasarkan kebutuhan fungsional dan non-fungsional yang telah ditentukan. Pengujian tersebut diantaranya fungsional dari perangkat yang digunakan, alur *state machine* yang diimplementasikan, dan waktu *delay* yang dikerjakan oleh sistem.

3.6 Kesimpulan

Pada tahap kesimpulan, akan dilakukan jika seluruh tahapan sebelumnya telah dilakukan dan sesuai dengan yang diharapkan. Kesimpulan akan berisi dari hasil pengujian dan analisis sistem, dan saran-saran yang dibutuhkan untuk penelitian selanjutnya. Diharapkan pada kesimpulan ditulis secara jelas agar pembaca paham hasil yang didapatkan dari penelitian ini.

BAB 4 REKAYASA KEBUTUHAN

Pada bab ini akan menjelaskan tentang gambaran umum sistem, kebutuhan pada sistem, dan kebutuhan fungsional dan non-fungsional pada sistem yang akan diharapkan.

4.1 Gambaran Umum Sistem

Pada penelitian kali ini, tujuan yang diharapkan yaitu perangkat pada lingkungan rumah cerdas dapat terhubung secara *pervasive* sehingga tidak diperlukan konfigurasi manual oleh manusia. Perangkat yang digunakan yaitu NI MyRIO yang bisa disebut *client* dan Raspberry Pi 3 yang bisa disebut *host*. Pada *client* akan bertindak sebagai node sensor dimana perangkat ini akan mengirimkan data sensor *accelerometer*, *state* LED dan *state push button* kepada *host*. LED pada *client* nantinya akan dapat dikendalikan oleh *host*. Sedangkan *host* akan bertindak sebagai koordinator dari node sensor yang dapat memantau dan mengendalikan apa saja fitur yang ada pada *client*.

Kedua perangkat ini akan terhubung melalui *WiFi* dengan metode *pervasive.* Dimana *client* akan mengirimkan pesan *broadcast* pada jaringan yang terhubung berisi data informasi yang dimiliki oleh *client* seperti IP, dan sensor yang tersedia. Setelah pesan di-*broadcast,* maka pesan tersebut akan diterima oleh *host* dan akan menyimpan informasi yang dimiliki oleh *client*. Setelah penyimpanan informasi selesai, maka *host* akan membalas pesan kepada IP *client* yang berisi informasi data *host*. Proses pengiriman ini memanfaatkan protokol UDP yang disediakan oleh LabVIEW.

4.2 Analisis Kebutuhan Sistem

Pada bab ini, akan dijelaskan keseluruhan kebutuhan sistem pada penelitian kali ini. Dalam melakukan analisis kebutuhan terdapat beberapa kebutuhan yang perlu dijelaskan yaitu kebutuhan sistem yang terdiri dari kebutuhan perangkat keras dan perangkat lunak, kebutuhan fungsional dan non fungsional sistem.

4.2.1 Kebutuhan Fungsional

Pada bagian ini, akan dijelaskan kebutuhan fungsional agar sistem berjalan seperti yang diharapkan, diantaranya:

1. *Host* dapat mendeteksi dan menyimpan informasi *client* yang baru aktif.

   Host akan berada pada *state Listen*, dimana perangkat ini akan mendengarkan segala pesan *broadcast* yang dikirimkan oleh *client* pada *state broadcast* pada jaringan lokal dengan format yang sudah ditentukan. Jika ada pesan *broadcast* yang dikirimkan oleh *client*, maka *host* akan menerima pesan tersebut dan berpindah *state Check HW* dimana perangkat ini akan menyimpan informasi dari *client* yang berisi IP dan fitur yang dimiliki oleh perangkat tersebut. Setelah *host* menyimpan informasi

tersebut, selanjutnya *host* berpindah ke *state Send ACK* dimana akan membalas pesan tersebut berisi informasi data *host*.

2. *Client* dapat menyimpan data informasi *host* yang terdeteksi.

   Setelah *host* di *state Send ACK* dimana mengirimkan pesan ACK pada *client*, maka *client* akan menyimpan informasi *host* dan akan berpindah pada *state Wait Command* dimana *client* akan menunggu perintah dari *host*.

3. *Client* dapat mengirimkan data sensor kepada *host*.

   Disaat *client* pada *state Wait Command*, *host* akan mengirimkan perintah untuk ingin terkoneksi dengan *client* agar bisa dipantau dan dikendalikan. Maka *client* akan berpindah pada *state Request*, dimana *client* akan mengirimkan data sensor *accelerometer* dan *host* dapat mengendalikan LED dari *client*.

4. *Host* dapat memantau dan mengendalikan fitur yang ada pada *client*.

   Disaat *host* mengirimkan perintah kepada *client*, *state Listen* akan berpindah pada *state Send Request.* Dimana pada *state* ini *host* dapat memantau sensor data *accelerometer* dan dapat mengendalikan LED *client*.

5. *Host* dapat memberitahu jika *client* terputus dari koneksi maupun ingin melakukan koneksi.

   Disaat *client* terputus koneksi dengan *host* pada *state Send Request*, maka pada antarmuka *host* akan memberi tahu jika koneksi telah terputus dan akan kembali pada *state Listen*, disaat itu juga informasi *client* yang terputus akan dihapus dari list. Keadaan ini juga berlaku disaat *host* mengirimkan perintah pada *client* tetapi perangkat tersebut sudah tidak terhubung.

6. *Client* dapat kembali ke *state* pencarian *host* jika terputus koneksi.

   Disaat *host* terputus koneksi dari *client*, maka *client* akan menunggu waktu hingga *host* aktif kembali. Jika pada waktu yang sudah ditentukan *host* tidak aktif, maka *client* akan berpindah *state broadcast* untuk mencari *host* yang baru.

4.2.2 Kebutuhan Non-Fungsional

Pada bagian ini, kebutuhan non-fungsional yang dibutuhkan pada penelitian ini agar sistem ini berjalan seperti yang diharapkan yaitu diantaranya:

1. *State machine* berjalan sesuai dengan perancangan.
2. Waktu d*elay* yang dibutuhkan dalam transisi antar *state*, pengiriman data, dan pengendalian fitur.

4.3 Batasan Desain

Batasan desain sistem dibuat agar pembuatan sistem dari penelitian ini tidak terlalu luas cakupannya. Maka berikut batasan desain yang akan membatasi penelitian ini diantaranya:

1. Program akan di *deploy* secara manual pada perangkat. Perangkat tidak akan otomatis menjalankan programnya di saat awal pengaktifan perangkat.
2. Proses pemantauan dan pengendalian dilakukan pada aplikasi LabVIEW yang terinstall pada laptop.
3. Jaringan lokal yang digunakan merupakan jaringan *hotspot* pada *smartphone*. Seluruh perangkat akan terhubung pada *hotspot* yang tersedia oleh *smartphone*.
4. Dibutuhkan internet untuk menyinkronkan waktu pada masing-masing perangkat agar dapat menentukan *delay* transmisi antar perangkat.

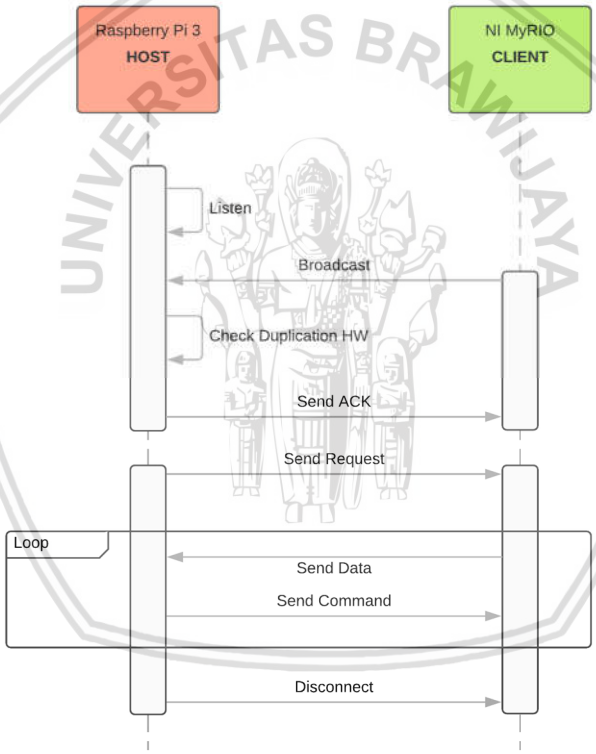
BAB 5 PERANCANGAN DAN IMPLEMENTASI

Pada bab ini, akan dijelaskan perancangan dan implementasi sistem berdasarkan tujuan penelitian. Perancangan sistem akan memuat seluruh alur dari sebuah sistem. Sedangkan implementasi sistem akan memuat fitur yang diimplementasikan pada perangkat NI MyRIO *(client)* dan Raspberry Pi 3 *(host)*.

5.1 Perancangan Sistem

5.1.1 Perancangan State Machine

Pada bagian ini, akan dijelaskan perancangan sistem yang dibutuhkan. Perancangan sistem ini yaitu berupa desain sistem yang mencakup seluruh alur dari perlakuan sistem. Pada Gambar 5.1 merupakan diagram interaksi sistem antara *host* dan *client.*



Gambar 5.1 Diagram Interaksi Sistem

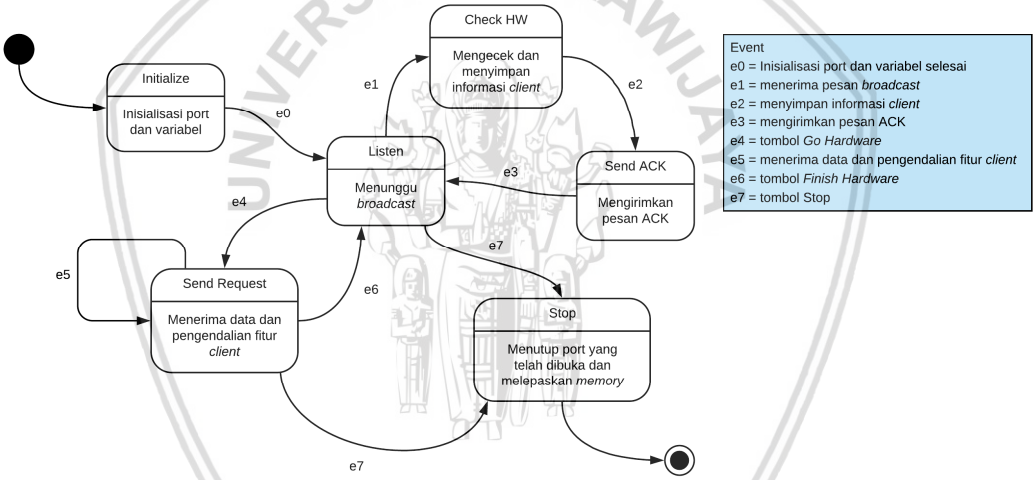
Pada keadaan pertama, *host* akan berada pada kondisi *listen* dimana *host* akan mendengarkan pesan *broadcast* yang dikirimkan oleh *client*. Disaat *client* mengirimkan pesan *broadcast*, *host* akan mengecek informasi *client* apakah telah tersimpan. Jika informasi *client* termasuk baru, maka *host* akan menyimpan informasi tersebut, dimana isi dari informasi yaitu IP, nama *client*, dan fitur yang disediakan. Setelah menyimpan informasi tersebut, *host* akan mengirimkan pesan balasan pada IP *client* saat *state Send ACK* yang berisi informasi IP dari *host*.

Setelah *host* dan *client* saling mengetahui informasi yang dimiliki, maka *host* dapat memantau dan mengendalikan fitur *client*. *Host* akan mengirimkan pesan *request* pada *client* dan *client* akan memulai mengirimkan data sensor dan menunggu perintah yang dikirimkan oleh *host*. *Host* akan menerima data sensor tersebut dan dapat mengendalikan fitur *client* dimana fitur tersebut adalah LED yang tersedia pada *client.* Seluruh dari interaksi antar *host* dan *client* memerlukan perantara protokol pengiriman agar data dapat sampai pada tujuan, dimana protokol pengiriman yang digunakan yaitu protokol UDP.

Pada sub-sub-bab selanjutnya akan lebih dijelaskan alur dari sistem yang berupa *state machine* pada *host* dan *client*.

5.1.1.1 State Machine Host

Pada sub-bab kali ini, akan dijelaskan alur sistem dari Raspberry Pi 3 sebagai *host*. Dapat dilihat pada Gambar 5.2 merupakan perancangan *state machine* pada *host*. Pada gambar tersebut, terdapat beberapa kondisi *state, event,* dan *action* yang akan terjadi pada kondisi saat itu.



Gambar 5.2 *State machine* pada *host*

Pada kondisi awal, *host* akan masuk kondisi *Initialize* dimana *host* akan mengisialisasi variabel yang dibutuhkan dan membuka *port* protokol UDP. Kemudian terjadi *event e0* dimana inisialisasi port dan variabel selesai, berpindah menuju state *Listen*. *State Listen* merupakan kondisi *host* disaat mendengarkan pesan *broadcast* yang dikirimkan oleh *client*.

Jika *host* menerima pesan *broadcast* dari *client*, akan terjadi *event e1* dan berpindah *state Check HW.* Pada *state Check HW, host* akan mengecek informasi yang diterimanya dengan informasi yang telah disimpan. Jika informasi *client* masih baru maka informasi akan disimpan oleh *host*, sedangkan jika informasi tersebut telah tersimpan sebelumnya maka informasi sebelumnya akan dihapus dan akan diganti dengan yang baru. Setelah penyimpanan informasi selesai, akan memicu *event e2* dan berpindah *state Send ACK*. Pada *state* ini akan mengirimkan

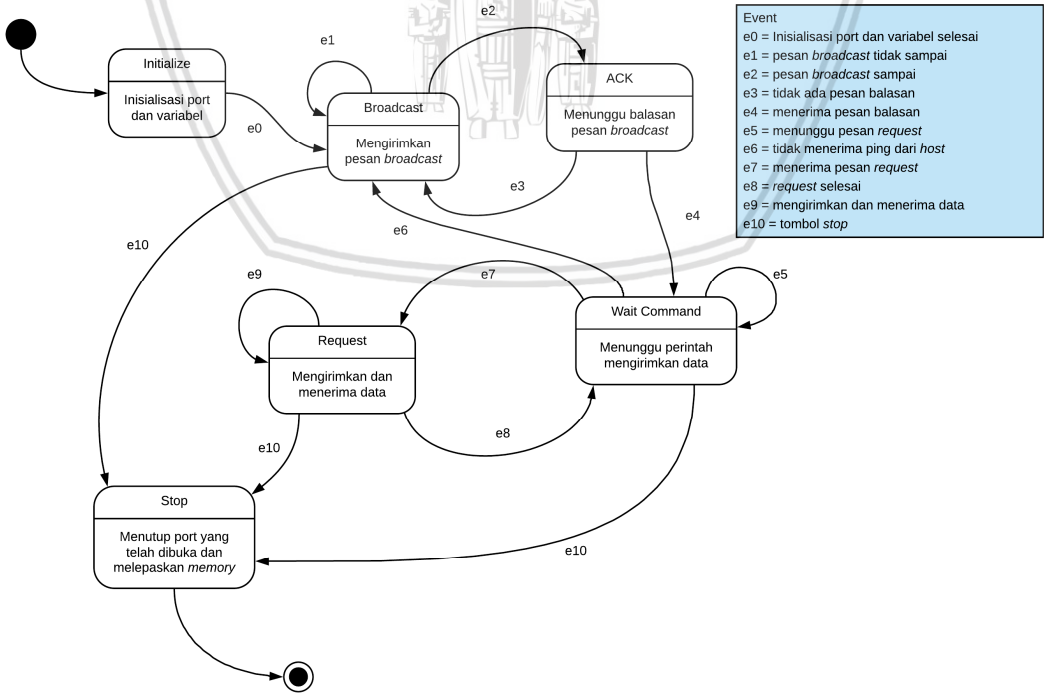
pesan balasan berupa informasi milik *host* menuju *client*. Setelah mengirimkan pesan balasan, akan terjadi *event e3* dimana akan kembali pada *state Listen*.

Pada *state Listen*, *host* dapat mengendalikan dan memantau sensor yang dimiliki oleh *client* yang informasinya telah tersimpan sebelumnya. *Host* akan mengirimkan pesan *request* pada *client* dimana terjadi *event e4* dan menunggu respon apakah pesan tersebut diterima. Jika pesan tersebut diterima maka akan berpindah pada *state Send Request*, jika tidak maka akan kembali pada *state Listen*. Disaat *state Send Request*, *host* akan menerima data sensor *accelerometer* dan informasi *push button,* dan LED *client*. LED pada *client* dapat dikendalikan langsung oleh *host* sehingga apapun perubahan yang dilakukan oleh *host,* maka pada *client* juga akan berubah. Pada proses pengiriman dan penerimaan data akan terjadi *event e5* dimana akan mengulang *state Send Request*. Jika telah selesai, *host* dapat menekan tombol *Finish Hardware* dan memicu *event e6* dimana akan berpindah *state Listen* kembali.

Pada keadaan *state Listen* dan *Send Request*, *host* dapat dihentikan dengan menekan tombol *Stop*. Disaat tombol *stop* ditekan, akan terjadi *event e7* dimana akan berpindah pada *state Stop*. Pada *state stop, host* akan menutup *port* yang dibuka disaat awal inisialisasi dan melepaskan memori yang telah disimpan.

5.1.1.2 State Machine Client

Pada sub-bab kali ini, akan dijelaskan alur sistem dari NI MyRIO sebagai *clliient*. Dapat dilihat pada Gambar 5.3 merupakan perancangan *state machine* pada *client*. Pada gambar tersebut, terdapat beberapa kondisi *state, event,* dan *action* yang akan terjadi pada kondisi saat itu.



Gambar 5.3 *State machine* pada *client*

Pada kondisi awal, *client* akan masuk pada *state Initialize* dimana *client* akan mengisialisasi variabel yang dibutuhkan dan membuka *port* protokol UDP. Kemudian terjadi *event e0* dimana inisialisasi port dan variabel selesai, berpindah menuju state *Broadcast*. *State Broadcast* merupakan kondisi *client* mengirimkan pesan *broadcast* pada IP *broadcast*. Jika pesan tersebut tidak ada balasan maka akan memicu *event e1* dimana akan mengulang *state Broadcast*, sedangkan jika pesan tersebut diterima oleh *host,* maka akan memicu *event e2* dan berpindah pada *state ACK*. Pada *state ACK*, *client* akan menunggu balasan dari *host* terkait informasi yang dimiliki oleh *host*. Jika *client* tidak menerima balasan akan memicu *event e3* dimana akan kembali pada *state Broadcast*, sedangkan jika menerima balasan dari *host* akan memicu *event e4* dimana akan berpindah pada *state Wait Command*.

Pada *state Wait Command*, *client* akan menunggu pesan *request* yang dikirimkan oleh *host*, jika tidak ada kiriman pesan dengan waktu yang ditentukan maka akan memicu *event e5* dimana akan mengulang *state Wait Command*. Dan bila *client* tidak menerima *ping* yang dikirimkan oleh *host* pada waktu yang ditentukan, maka akan memicu *event e6* dimana *client* akan kembali pada *state Broadcast* untuk mencari *host* baru karena *host* sebelumnya sudah tidak aktif.

Jika *client* saat keadaan *state Wait Command* menerima pesan *request*, maka akan memicu *event e7* dimana akan berpindah pada *state Request*. Pada *state Request*, *client* akan mengirimkan data sensor *accelerometer* dan informasi dari *push button,* dan LED yang dimiliki oleh *client*. Pada *state* ini akan memicu *event e9* dimana akan mengulang *state Request* dan tetap mengirimkan informasi pada *host*. Jika *host* telah selesai melakukan *request* data, maka akan memicu *event e7* dimana *client* akan berpindah pada *state Wait Command* kembali.

Pada keadaan *state Broadcast, Wait Command* dan *Request*, *client* dapat dihentikan dengan menekan tombol *Stop*. Disaat tombol *stop* ditekan, akan terjadi *event e10* dimana akan berpindah pada *state Stop*. Pada *state stop, client* akan menutup *port* yang dibuka disaat awal inisialisasi dan melepaskan memori yang telah disimpan.

5.1.2 Perancangan Koneksi Perangkat

Pada perancangan koneksi *wifi* perangkat, perangkat *host* dan *client* akan dikonfigurasi untuk terhubung dengan *hotspot*. *Hotspot* yang digunakan merupakan *hotspot* yang disediakan pada *smartphone*. Kelebihan dari *Hotspot smartphone* yaitu mudahnya untuk melihat perangkat yang telah terhubung dengan *hotspot.*

5.1.3 Perancangan LINX LabVIEW

LabVIEW merupakan pemrograman berbasis *dataflow programming*, dimana kode program yang dapat diimplementasikan hanya pada perangkat milik National Instrument salah satunya yaitu NI myRIO. Kode program LabVIEW juga dapat diimplementasikan pada perangkat lainnya dengan *library* yang dibutuhkan untuk

perangkat tersebut. *Library* yang dapat digunakan untuk perangkat Raspberry Pi 3 agar kode program dapat diimplementasikan yaitu *Library LINX LabVIEW.*

5.1.4 Perancangan Sensor *Accelerometer client*

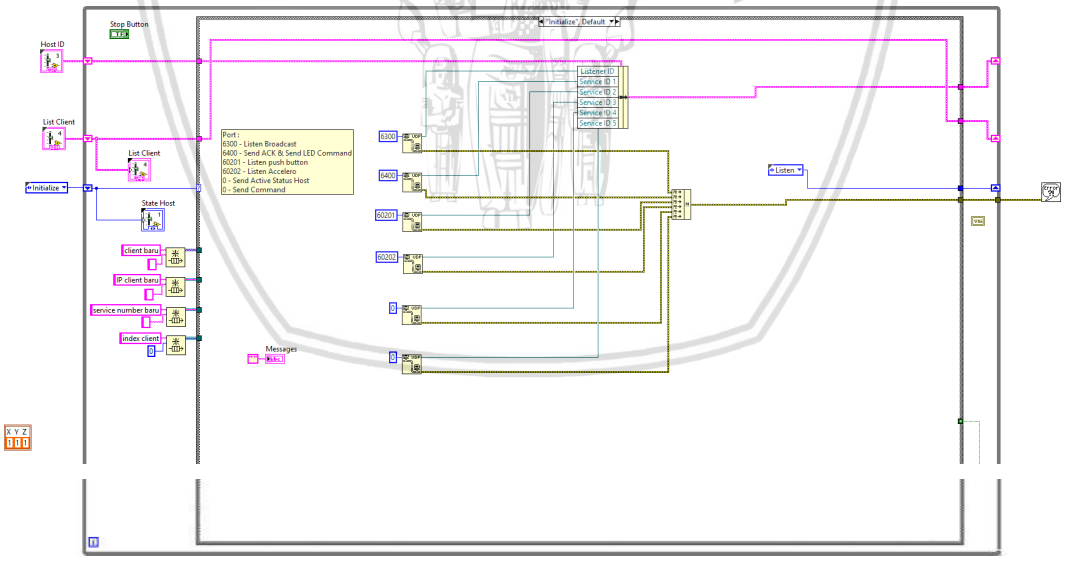
Perangkat NI myRIO memiliki beberapa fitur salah satunya yaitu sensor *accelerometer*. Sensor ini merupakan sensor yang telah disediakan myRIO. Pada perancangan ini, peneliti akan memanfaatkan sensor yang telah disediakan agar berfungsi sebagai mestinya.

5.2 Implementasi Sistem

Pada bagian ini akan dijelaskan implementasi sistem sesuai perancangan sistem sebelumnya. Implementasi sistem ini berupa kode program yang telah dibuat sesuai dengan desain sistem yang ada para perancangan sistem. Pada sub-sub-bab selanjutnya akan dijelaskan bentuk kode program yang akan diimplementasikan pada *host* dan *client*.

5.2.1 Implementasi *state machine* pada *Host*

Pada sub-bab kali ini akan dijelaskan kode program yang akan diimplementasikan pada Raspberry Pi 3 sebagai *host*. Kode program akan dibuat seperti alur yang dibutuhkan seperti pada perancangan *state machine host.* Pada Gambar 5.4 merupakan cuplikan keseluruhan kode program *host*. Kode program tiap *state* akan dijelaskan pada sub-sub-bab berikutnya.



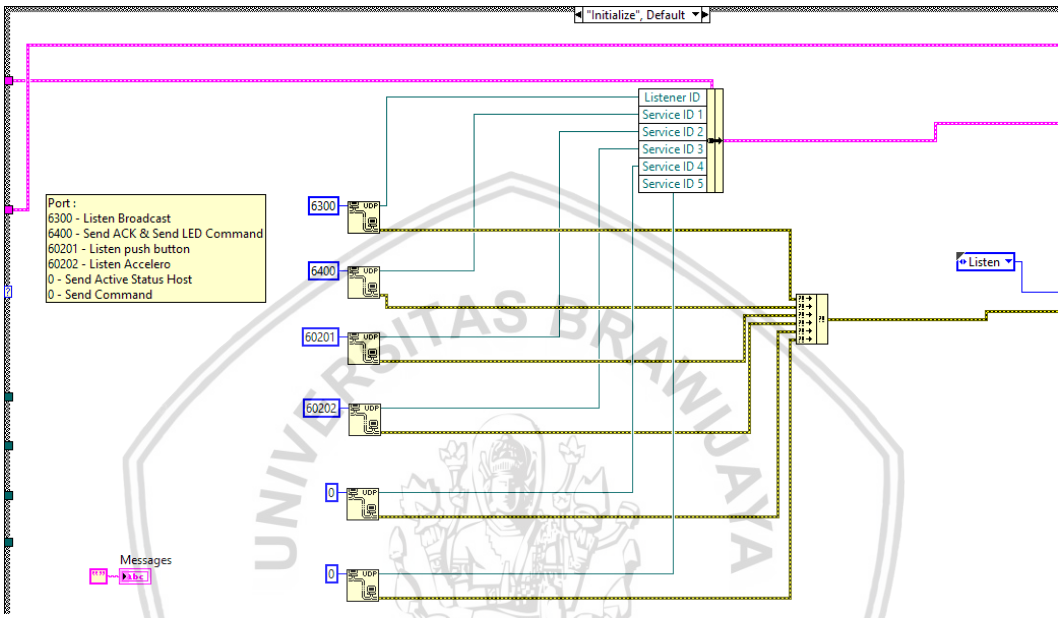
Gambar 5.4 Implementasi kode program *Host*

Pada Gambar 5.4 merupakan cuplikan keseluruhan dari kode program *host* pada *state Initialize*. Pada gambar tersebut terdapat beberapa inisialisasi diantaranya inisialisasi *state, service id (Host ID), List Client,* variabel XYZ yang berfungsi untuk menyimpan nilai sensor, dan *Queue* untuk menyimpan informasi sementara yang didapat dari *client*. Kemudian terdapat perulangan *while* pada

bagian luar yang akan menjalankan selalu program tersebut hingga dihentikan oleh pengguna.

5.2.1.1 Implementasi state Initialize

Pada Gambar 5.5 merupakan cuplikan kode program *state Initialize* pada *host*. Pada kode program tersebut dilakukan inisialisasi untuk membuka port UDP dan variabel *Messages*, dimana variabel ini untuk menampilkan pesan yang ingin disampaikan oleh sistem.

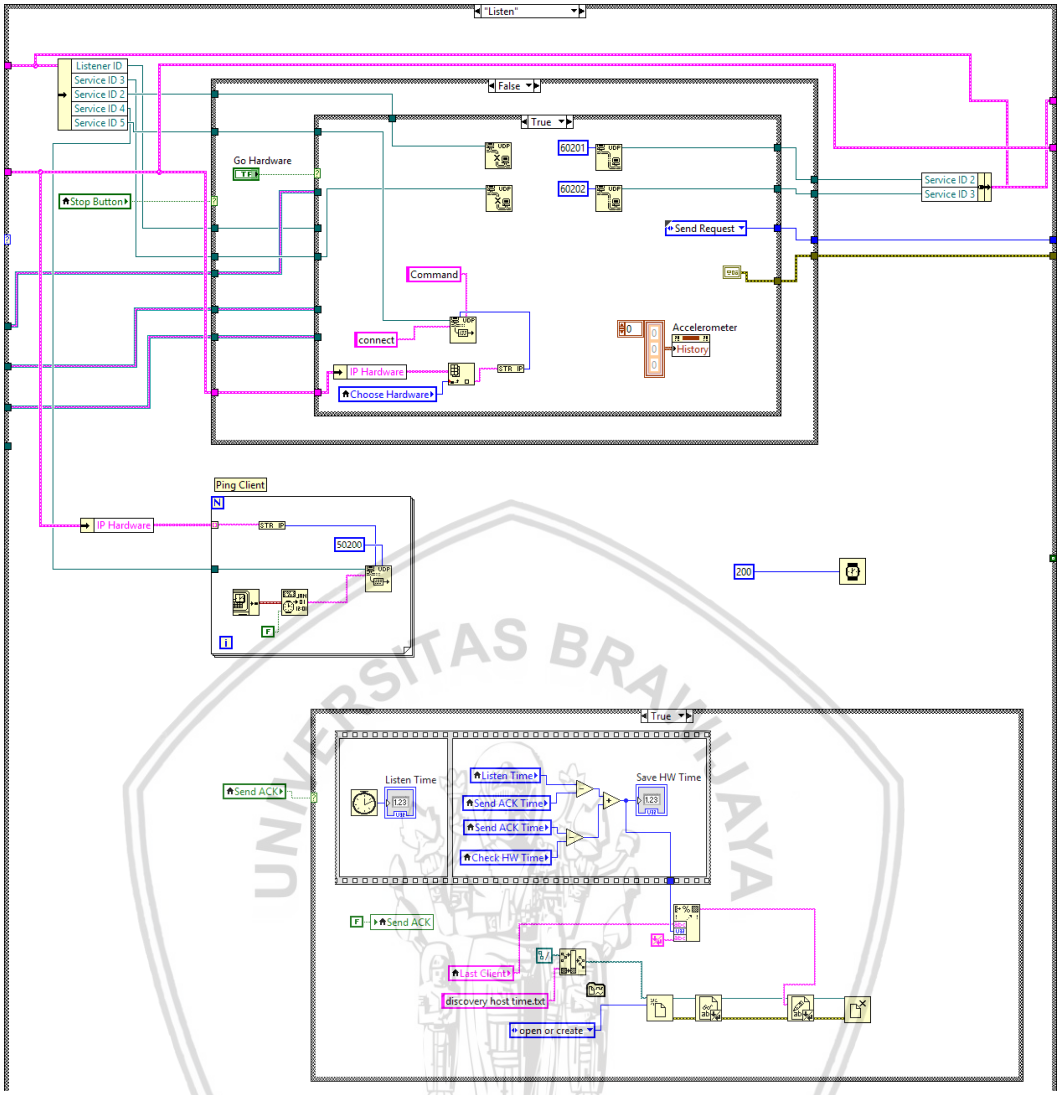


Gambar 5.5 Kode program *state Initialize host*

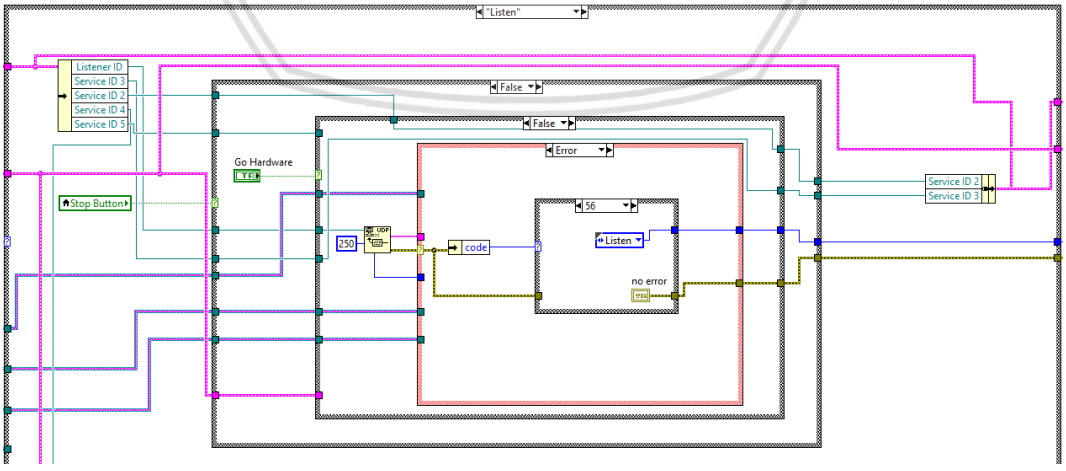
Untuk inisialisasi port UDP, port yang dibuka sebanyak enam dimana port ini memiliki fungsi masing-masing. Port tersebut diantaranya yaitu port 6300 *(Listener ID)* untuk mendengarkan *broadcast* dari *client*, port 6400 *(Service ID 1)* untuk mengirimkan pesan ACK dan mengirimkan perintah LED, port 60201 *(Service ID 2)* untuk menerima data *state push button,* port 60202 *(Service ID 3)* untuk menerima data sensor *Accelerometer*, dan dua port bebas *(Service ID 4 dan 5)* untuk mengirimkan status aktif host dan mengirimkan perintah untuk *Request* pada *client*.

5.2.1.2 Implementasi state Listen

Pada Gambar 5.6 merupakan cuplikan kode program *state Listen.* Pada *state* ini terdapat beberapa fungsi yang dijalankan secara bersamaan. Fungsi tersebut diantaranya, menerima pesan *broadcast* dari *client,* mengirimkan pesan aktif kepada *client*, mengirimkan pesan *request,* dan penyimpanan waktu *delay* pada file.



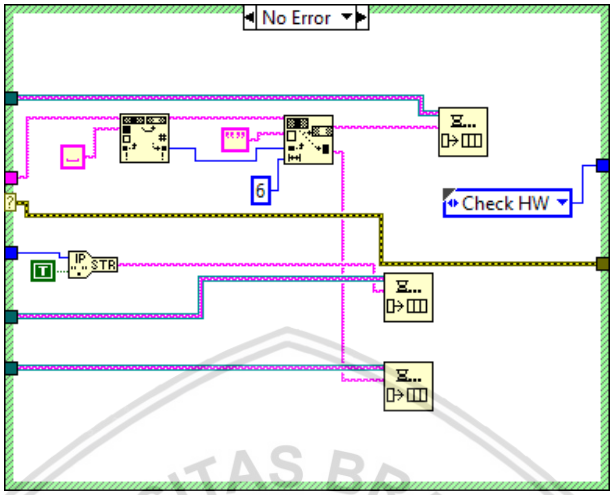
Gambar 5.6 Kode program *state Listen host*



Gambar 5.7 Kode program perulangan *state Listen*

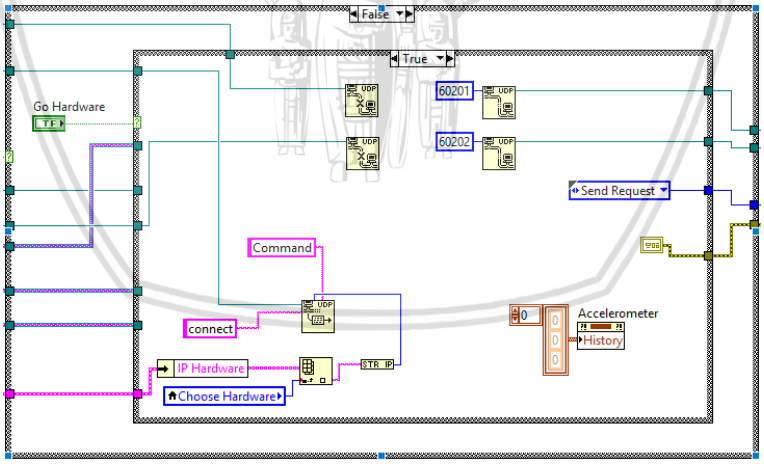
Pada Gambar 5.7 merupakan kode program *state listen* disaat *host* tidak menerima pesan *broadcast*. Pada *Listener ID*, akan mengecek apakah ada pesan

broadcast yang dikirimkan oleh *client*, jika tidak ada maka fungsi *UDP Read* akan menghasilkan *error 56,* dimana error ini menandakan bahwa tidak ada pesan yang masuk. Sehingga *state* selanjutnya setelah dilakukan pengecekan pesan *broadcast,* akan mengulang pada *state Listen* dan dianggap tidak ada *error*.



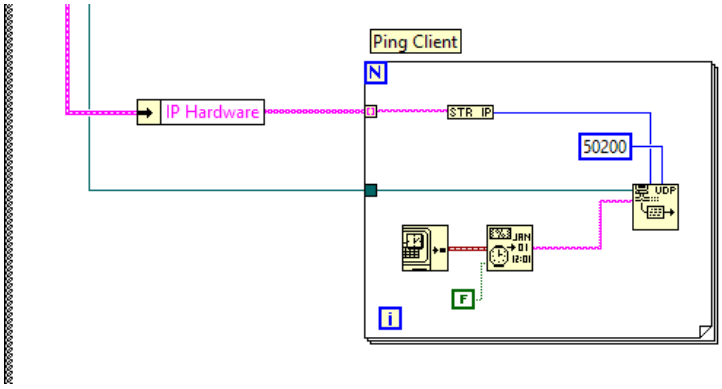
Gambar 5.8 Kode program penerimaan pesan *broadcast* sukses

Pada Gambar 5.8 merupakan kode program *state listen* disaat *host* menerima pesan *broadcast*. *Host* akan memecah data yang diterima hingga dihasilkan nama *client, IP client,* dan fitur yang dimiliki oleh *client.* Informasi tersebut dimasukkan pada *queue* yang akan diolah nantinya. Pada keadaan ini, *state* selanjutnya yaitu *state Check HW.*



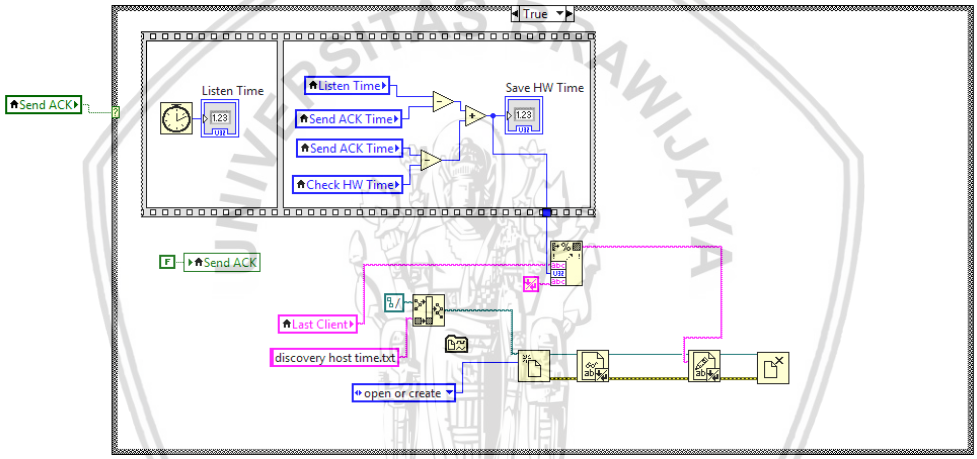
Gambar 5.9 Kode program *Go Hardware button*

Fungsi selanjutnya yaitu fungsi mengirimkan pesan *request* pada *client.* Pada Gambar 5.9 merupakan kode program *state Listen* jika tombol *Go Hardware* ditekan. Pada keadaan tersebut, *host* akan mengirimkan pesan *connect* pada *client* yang dipilih, dan dikirimkan pada *service name Command* yang ada pada *client*. Selain itu, data *Accelerometer* akan di-*reset* kemudian port *Service ID 2 (60201)* dan *Service ID 3 (60202)* akan dibuka ulang untuk menghindari adanya *buffering* data yang tersimpan. Pada keadaan ini *state* selanjutnya yaitu *state Send Request*.



Gambar 5.10 Kode program mengirim pesan aktif *host*

Fungsi selanjutnya yaitu fungsi mengirimkan pesan aktif *host* pada *client.* Pada Gambar 5.10 merupakan kode program *state Listen* untuk pengiriman pesan aktif *host* pada *client.* Pesan tersebut berisikan waktu *host* saat ini dan dikirimkan melalui melalui *Service ID 4* dan port 50200 pada *client*.

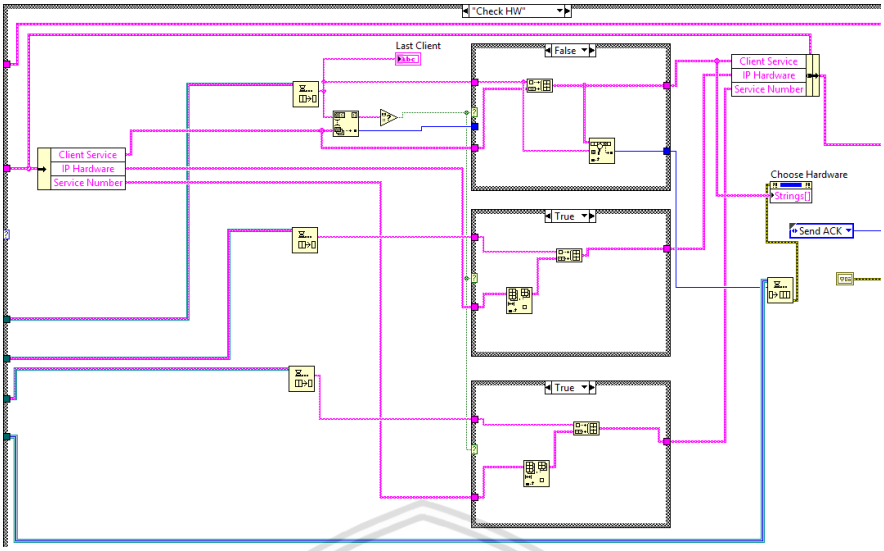


Gambar 5.11 Kode program penyimpanan waktu *delay* pada file

Fungsi selanjutnya yaitu penyimpanan waktu *delay* pada file. Pada Gambar 5.11 Kode program penyimpanan waktu *delay* pada file. Pada fungsi tersebut akan menghitung waktu yang dibutuhkan *host* pada *state Listen-Check HW-Send ACK* dan kembali lagi pada *state Listen,* yang kemudian data tersebut akan disimpan pada file lokal *host* dengan nama *discovery host time.txt*. Fungsi ini akan dijalankan jika *host* telah masuk *state Send ACK* sebelumnya.

5.2.1.3 Implementasi state Check HW

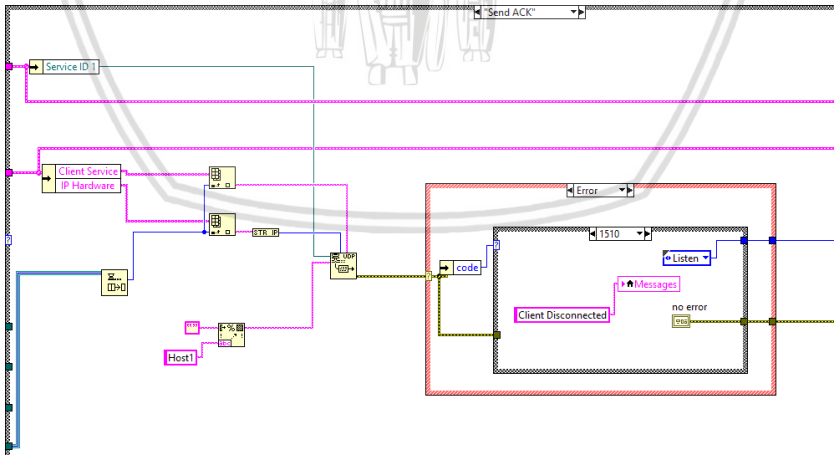
Pada Gambar 5.12 Kode program *state Check HW*. Pada *state* ini, *host* akan mengecek duplikasi nama *client.* Informasi diambil dari *queue* yang telah disimpan sebelumnya pada *state Listen.* Jika nama *client* telah ada sebelumnya, maka *array* yang mengandung informasi *client* tersebut akan dihapus, kemudian informasi *client* yang baru akan disimpan. Jika nama *client* tidak ada sebelumnya, maka informasi tersebut akan langsung disimpan. Informasi yang disimpan diantaranya nama *client,* IP *client,* dan pelayanan yang dimiliki. Pada keadaan ini, *host* akan mencatat waktu saat ini dan *state* selanjutnya yaitu *state Send ACK.*



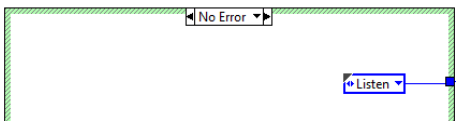
Gambar 5.12 Kode program *state Check HW host*

5.2.1.4 Implementasi state Send ACK

Pada Gambar 5.13 Kode program *state Send ACK*. Pada *state* ini, *host* akan mengirimkan pesan *ack* pada *client*. Pengiriman akan melalui *Service ID 1* pada *Client Service* dan *IP Hardware client* yang dituju. Pesan yang dikirimkan berisikan nama dari *host* yaitu *Host1*. Jika pesan tersampaikan pada *client,* maka *host* akan kembali pada *state Listen* yang dapat dilihat pada Gambar 5.14. Sedangkan jika pesan tidak sampai pada *client* dalam hal *client* telah mati, maka sistem akan mengalami *error* dengan kode 1510 dengan menampilkan pesan *Client Disconnected* dan akan kembali pada *state Listen* yang dapat dilihat pada Gambar 5.13. Pada keadaan ini, *host* akan mencatat waktu saat ini (masuk *state Send ACK).*



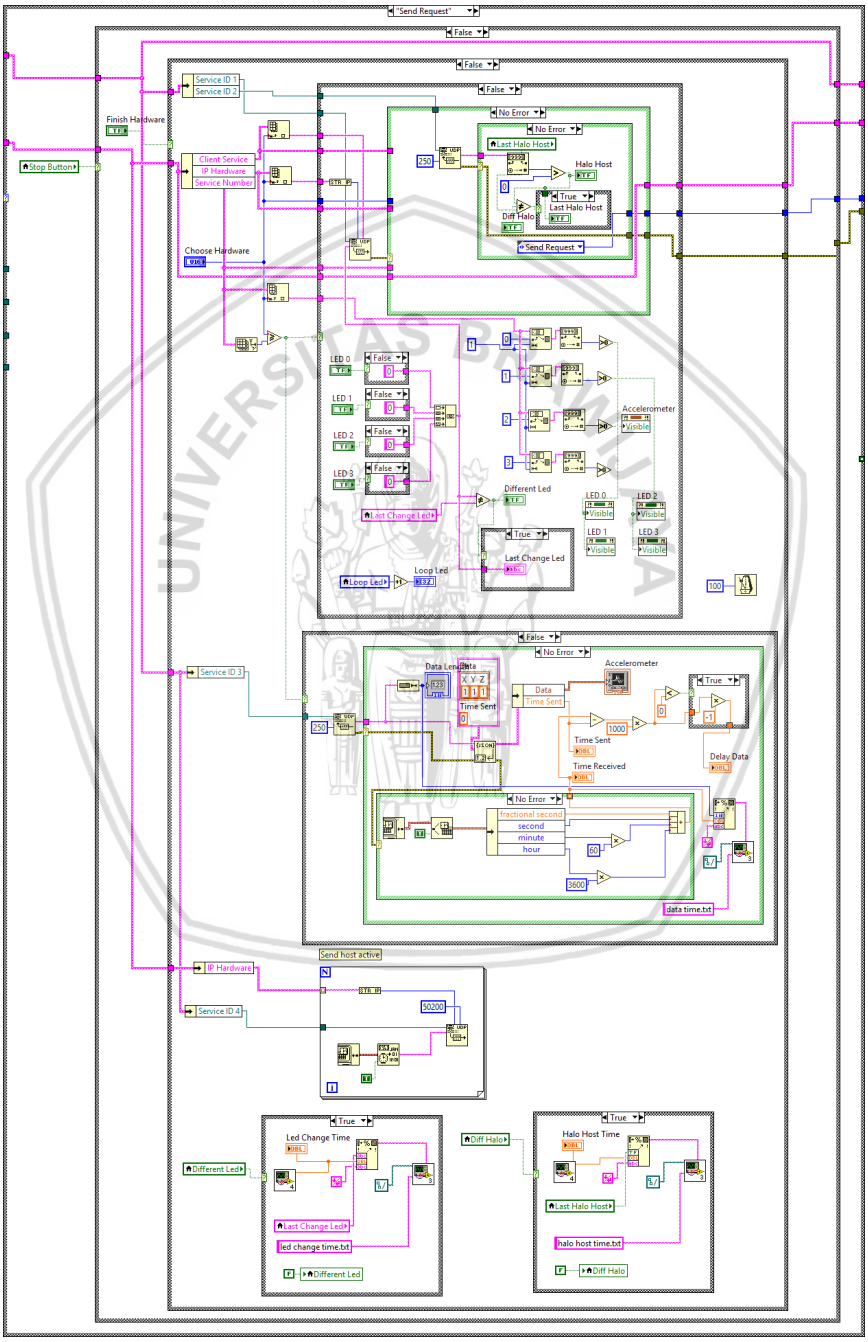
Gambar 5.13 Kode program *state Send ACK host*



Gambar 5.14 Kode program kembali pada state *Listen*

5.2.1.5 Implementasi state Send Request

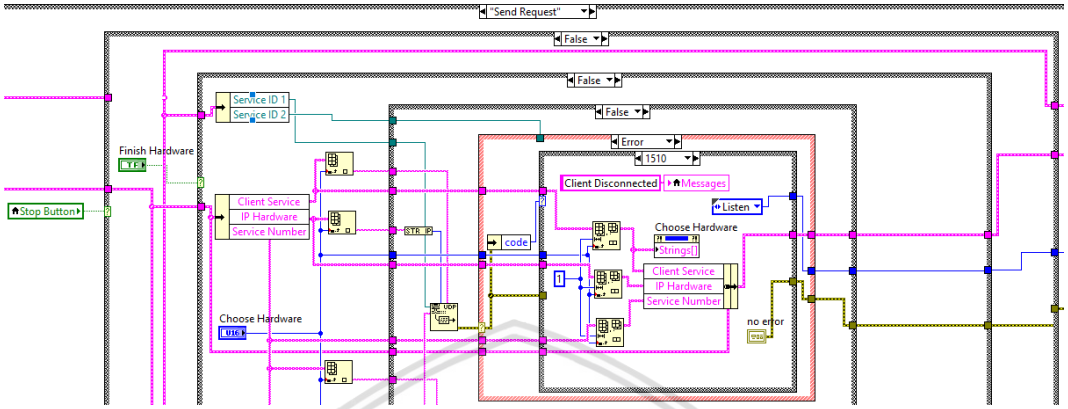
Pada Gambar 5.15 Kode program *state Send Request*. Pada *state* ini *host* dan *client* akan melakukan pertukaran data dimana *host* melakukan pemantauan dan pengendalian pada *client. State* ini terdapat beberapa fungsi yang diantaranya penerimaan data *state push button, client* terputus koneksi dari *host,* pengiriman *state LED,* jika *host* tidak menemukan informasi *client*, saat *host* telah selesai memantau dan mengendalikan *client,* dan saat *host* programnya dihentikan.



Gambar 5.15 Kode program *state Send Request host*

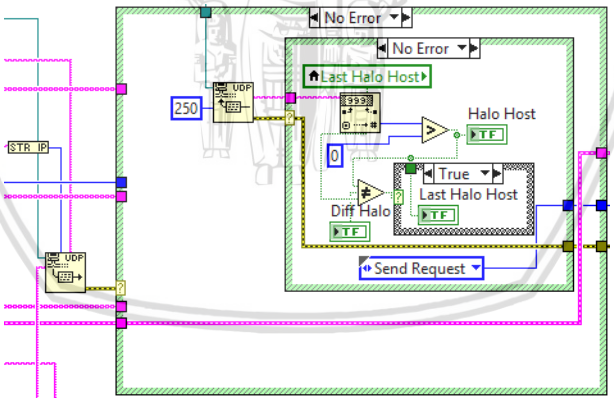
Fungsi yang pertama yaitu saat *client* terputus koneksi dari *host.* Pada Gambar 5.16 Kode program saat *client* terputus dari *host*. *Host* akan mengirimkan data

berupa *state LED* kepada *client* yang merupakan pengendalian *LED client* oleh *host* melalui *Service ID 1,* jika pengiriman error dengan kode 1510 maka *client* dianggap tidak aktif, kemudian *host* akan menghapus informasi *client* dan akan kembali pada *state Listen* dengan menampilkan pesan sistem *Client Disconnected.*

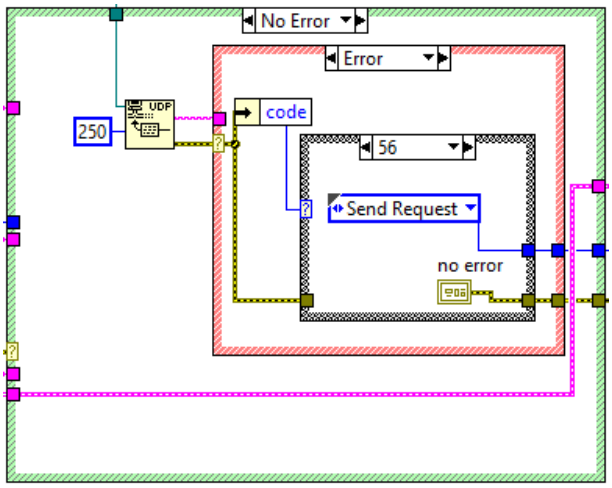


Gambar 5.16 Kode program saat *client* terputus dari *host*

Jika *host* sukses untuk mengirimkan data berupa *state LED* kepada *client,* maka selanjutnya *host* akan menerima data berupa *state push button client* melalui *Service ID 2.* Pada keadaan ini, *host* akan tetap pada keadaan *state Send Request* dan menyimpan keadaan jika terjadi perbedaan *state push button* dengan data sebelumnya, yang dapat dilihat pada Gambar 5.17. Jika *host* tidak menerima pesan dari *client*, maka *host* akan tetap pada keadaan *state Send Request* yang dapat dilihat pada Gambar 5.18.

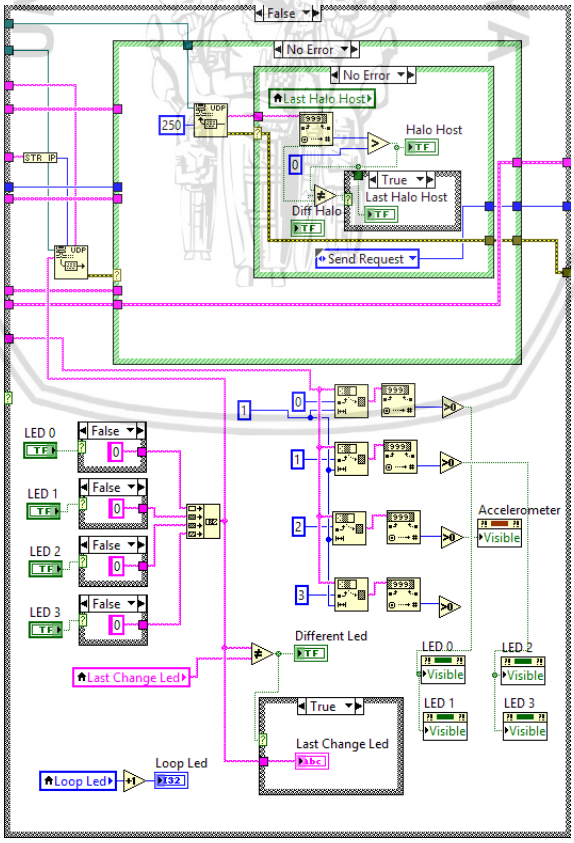


Gambar 5.17 Kode program penerimaan *state push button*



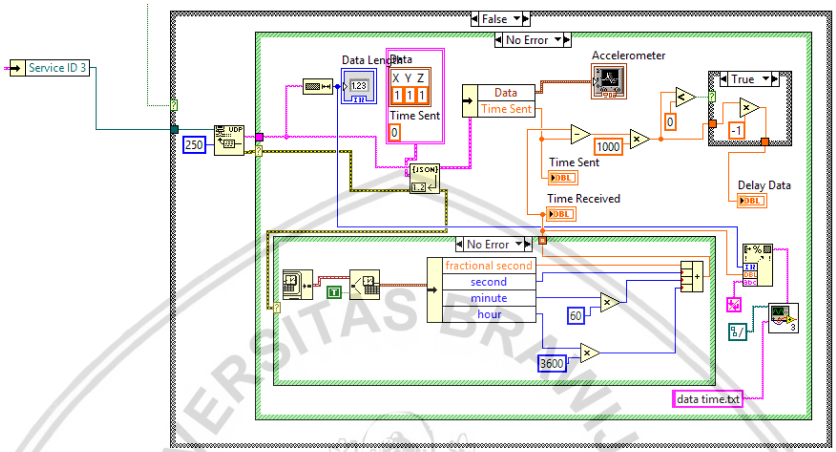
Gambar 5.18 Kode program gagal menerima pesan *state push button*

Fungsi selanjutnya yaitu pengiriman *state LED* pada *client.* Pada Gambar 5.19 Kode program pengiriman *state LED*. *State LED* akan diambil dari masukan keadaan *state* pada antarmuka LabVIEW, yang direpresentasikan dengan LED 0, LED 1, LED 2, LED 3. *State LED* tersebut akan digabungkan menjadi satu string dan dikirimkan melalui *Service ID 1.* Jika terjadi perbedaan dengan *state LED* sebelumnya, maka akan ditandai sebagai perbedaan dan menyimpan *state* tersebut.



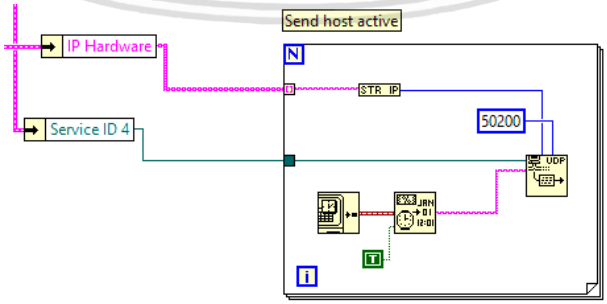
Gambar 5.19 Kode program pengiriman *state LED*

Pada Gambar 5.19 bagian kanan bawah, terdapat kode program untuk menampilkan fungsi LED dan grafik *accelerometer* pada antarmuka LabVIEW *host.* Jika *client* memiliki pelayanan yang lengkap, maka *host* akan menampilkan seluruh fitur yang dimiliki. Dimana perwakilan pelayanan tersebut yang dimiliki *client* diantaranya, pelayanan pertama dengan indeks 0 yaitu LED 0 dan LED 1, pelayanan kedua dengan indeks 1 yaitu LED 2 dan LED 3, pelayanan ketiga dengan indeks 2 yaitu grafik *accelerometer,* dan pelayanan keempat dengan indeks 3 tidak ada fitur.



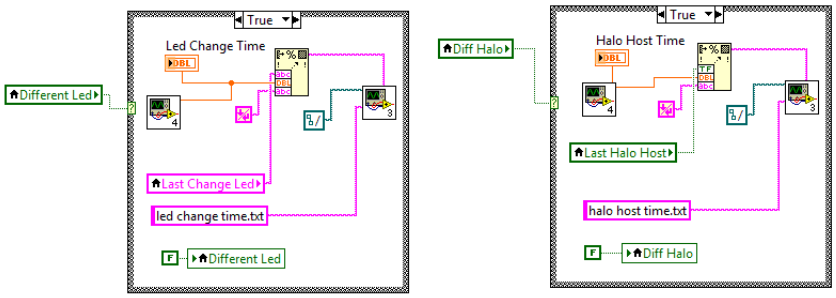
Gambar 5.20 Kode program penerimaan data sensor *accelerometer*

Fungsi selanjutnya yaitu penerimaan data sensor *accelerometer* dari *client* yang dapat dilihat kode programnya pada Gambar 5.20. *Host* akan menerima data melalui *Service ID 3.* Kemudian data tersebut diukur panjangnya menggunakan *Data Length,* dan dilakukan fungsi *Unflatten from JSON* agar data dapat digunakan kembali. Data tersebut kemudian ditampilkan pada grafik *Accelerometer.* Selain itu *host* akan mencatat waktu saat menerima data tersebut, kemudian mengurangi dengan waktu saat data tersebut dikirim oleh *client* sehingga menghasilkan waktu *delay* pengiriman antara *host* dan *client.* Kemudian setelah didapat panjang data dan waktu *delay* pengiriman, selanjutnya akan disimpan pada file yang bernama *data time.txt*.



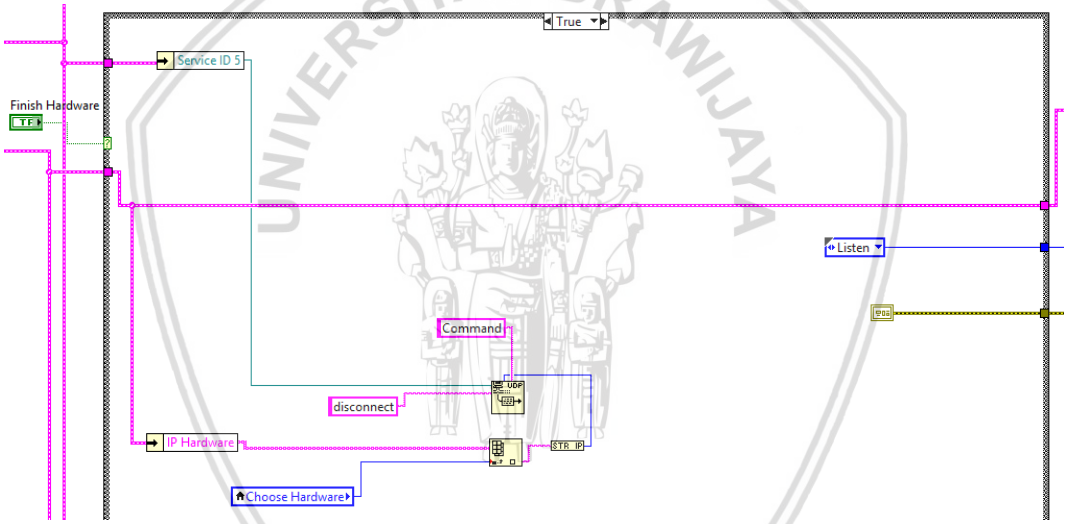
Gambar 5.21 Kode program mengirim pesan aktif *host*

Fungsi selanjutnya yaitu fungsi mengirimkan pesan aktif host pada client. Pada Gambar 5.21 merupakan kode program state Listen untuk pengiriman pesan aktif *host* pada *client*. Pesan tersebut berisikan waktu host saat ini dan dikirimkan melalui melalui *Service ID 4* dan port 50200 pada *client*.



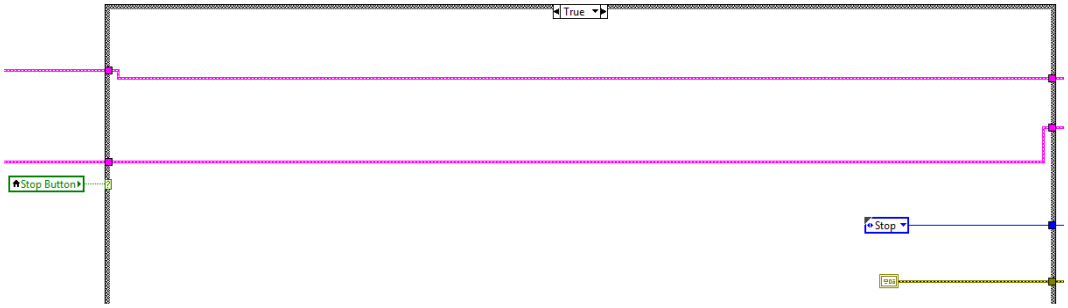
Gambar 5.22 Kode program pencatatan waktu

Fungsi selanjutnya yaitu pencatatan waktu perubahan *state LED* dan *state push button* yang dapat dilihat pada Gambar 5.22. Pada bagian kiri gambar, jika terjadi perbedaan *state LED* dengan keadaan sebelumnya, maka akan dicatat waktunya dan disimpan dengan nama *led change time.txt* dan memberi nilai *false* kepada *Different Led.* Sedangkan pada bagian kanan gambar, jika terjadi perbedaan *state push button* (dengan nama *Diff Halo)* maka akan dicatat waktu dan disimpan dengan nama *halo host time.txt dan* memberi nilai *false* kepada *Diff Halo.*



Gambar 5.23 Kode program tombol *Finish Hardware*

Fungsi selanjutnya yaitu memutus koneksi dengan *client* jika *host* telah selesai melakukan pengendalian dan pemantauan *client.* Jika pada antarmuka LabVIEW *host* menekan tombol *Finish Hardware,* maka akan menjalankan kode program yang dapat dilihat pada Gambar 5.23. Pada keadaan ini, *host* akan mengirimkan pesan *disconnect* kepada *client* dengan *Service name Command* melalui *Service ID 5* dan selanjutnya *host* akan kembali pada *state Listen.*

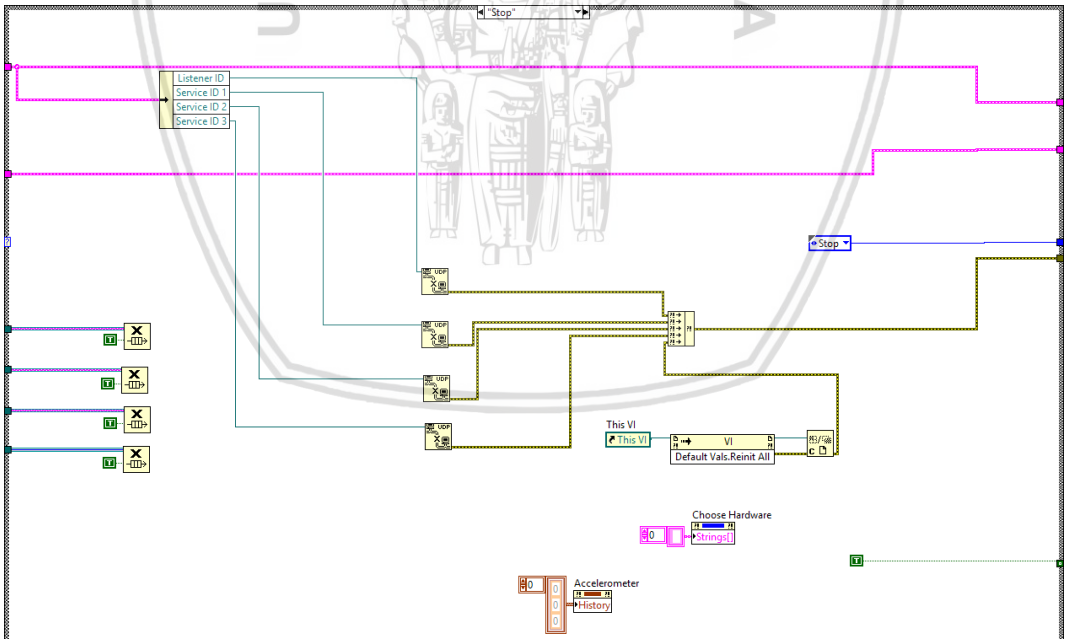


Gambar 5.24 Kode program tombol *stop*

Fungsi selanjutnya yaitu jika pada antarmuka LabVIEW *host* menekan tombol *Stop. Host* akan berpindah pada *state Stop* jika tombol *stop* ditekan. Kode program dapat dilihat pada Gambar 5.24.

5.2.1.6 Implementasi state Stop

State machine host selanjutnya yaitu *state Stop.* Pada keadaan ini *host* akan menutup semua port yang telah dibuka pada awal inisialisasi diantaranya *Listener ID, Service ID 1, Service ID 2, Service ID 3.* Kemudian akan melepaskan memori yang disimpan jika ada *queue* yang belum dikeluarkan, me-*reset* nilai yang disimpan pada *Accelerometer* dan akan me-*reset* tampilan antarmuka seperti awal. Kode program *state Stop* dapat dilihat pada Gambar 5.25. Setelah itu akan mengirimkan nilai *True* untuk menghentikan perulangan pada perulangan *while*.

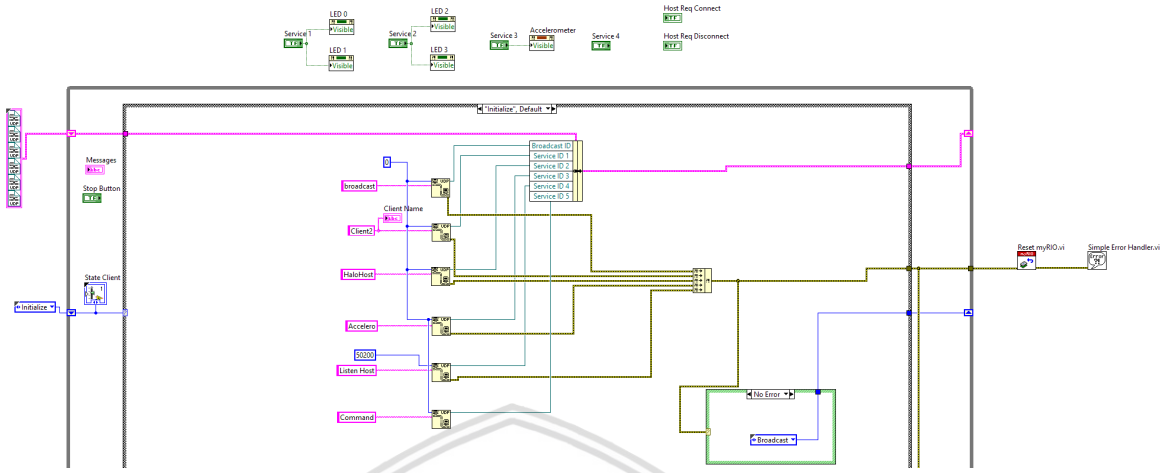


Gambar 5.25 Kode program *state Stop host*

5.2.2 Implementasi *state machine* pada *Client*

Pada sub-bab kali ini akan dijelaskan kode program yang akan diimplementasikan pada NI myRIO sebagai *client*. Kode program akan dibuat seperti alur yang dibutuhkan seperti pada perancangan *state machine client.* Pada

Gambar 5.26 merupakan cuplikan keseluruhan kode program *client*. Kode program tiap *state* akan dijelaskan pada sub-sub-bab berikutnya.

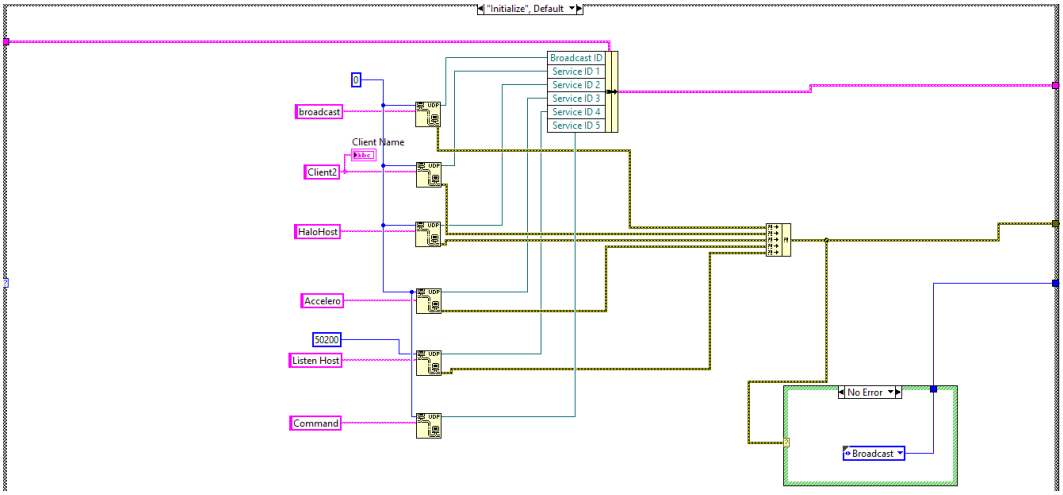


Gambar 5.26 Implementasi kode program *client*

Pada Gambar 5.26 Implementasi kode program *client* pada *state Initialize.* Pada gambar tersebut terdapat beberapa inisialisasi diantaranya inisialisasi *state, service id,* variabel *Messages* dan *Stop Button,* pelayanan yang disediakan diantaranya *Service 1* yaitu *LED 0* dan *LED 1, Service 2* yaitu *LED 2* dan *LED 3, Service 3* yaitu *Accelerometer, Service 4* dengan fitur kosong, dan variabel *Host Req Connect* dan *Host Req Disconnect*. Kemudian terdapat perulangan *while* pada bagian luar yang akan menjalankan selalu program tersebut hingga dihentikan oleh pengguna. Setelah perulangan dihentikan, maka myRIO akan di-*reset* dan menampilkan *error* jika terjadi kegagalan sistem.

5.2.2.1 Implementasi state Initialize

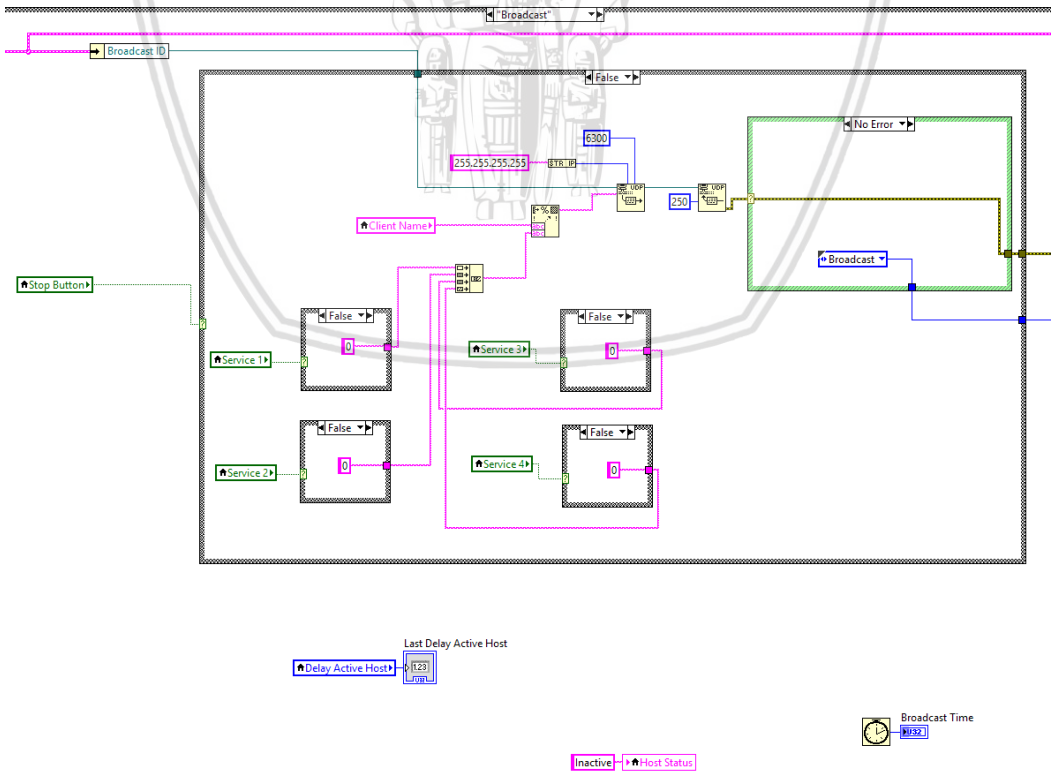
State machine client pertama yaitu *state Initialize,* dimana pada keadaan ini *client* akan membuka port UDP. Port UDP yang dibuka diantaranya port bebas dengan *service name broadcast* dengan nama *Broadcast ID,* nama *client (Client2)* dengan nama *Service ID 1, HaloHost (push button)* dengan nama *Service ID 2, Accelero* dengan nama *Service ID 3 dan Command* dengan nama *Service ID 5,* sedangkan *service name Listen Host* menggunakan port 50200 dengan nama *Service ID 4*. Kode program dapat dilihat pada Gambar 5.27. Pada keadaan ini, *state client* selanjutnya yaitu *state Broadcast.*



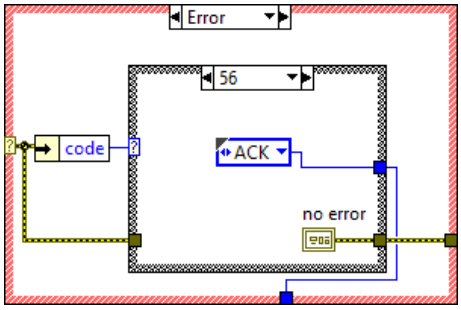
Gambar 5.27 Kode program *state Initialize client*

5.2.2.2 Implementasi state Broadcast

Pada Gambar 5.28 Kode program *state Broadcast client*. Pada keadaan ini, *client* akan mengirimkan pesan *broadcast* yang berisi nama *client* dan layanan yang ada pada *client*. Pesan ini dikirimkan melalui *Broadcast ID* pada IP *broadcast* yaitu 255.255.255.255 dengan port 6300. Jika tidak ada error, maka *state* akan tetap pada *state Broadcast,* sedangkan jika ada *error* dengan kode 56 maka akan dilanjutkan pada *state ACK* dan dianggap tidak ada *error* yang dapat dilihat pada Gambar 5.29.

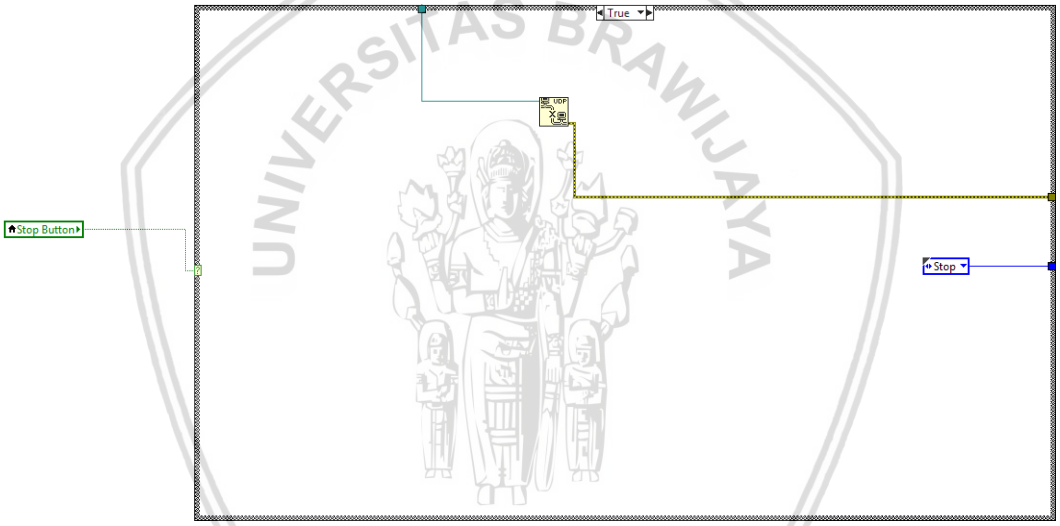


Gambar 5.28 Kode program *state Broadcast client*



Gambar 5.29 Kode program *broadcast error*

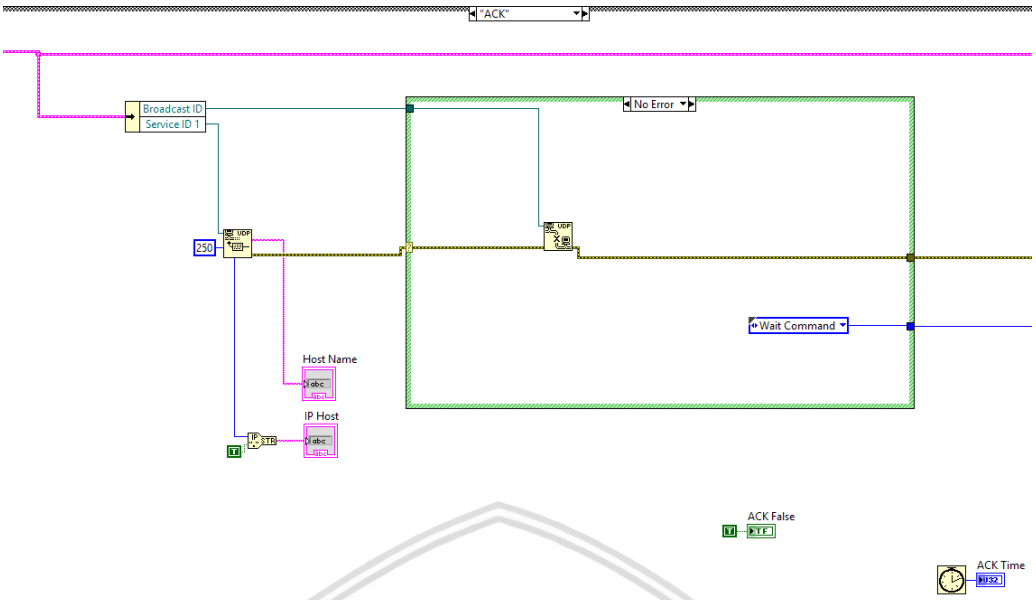
Fungsi selanjutnya pada *state Broadcast* yaitu pencatatan waktu disaat *client* memasuki *state Broadcast*. Selain itu penampilan pesan sistem *Inactive*, dan penampilan waktu terakhir host aktif. Gambar dapat dilihat pada Gambar 5.28. Fungsi lainnya yaitu disaat tombol *Stop Button* ditekan, maka port *Broadcast ID* akan ditutup dan *state client* akan berpindah pada *state Stop* yang dapat dilihat pada Gambar 5.30.



Gambar 5.30 Kode program *state Broadcast client stop*

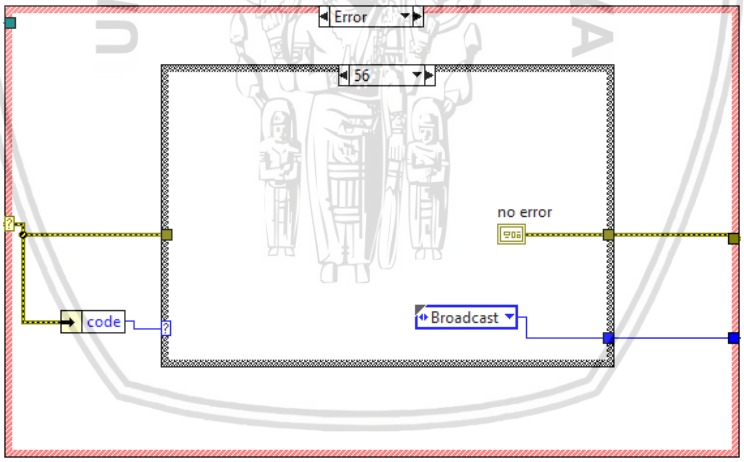
5.2.2.3 Implementasi state ACK

Pada Gambar 5.31 Kode program *state ACK client*. Pada state ini *client* menggunakan dua *port* UDP yaitu *Broadcast ID* dan *Service ID 1. Client* akan membaca dengan fungsi *UDP Read* apakah ada pesan balasan dari *host* pada port *Service ID 1.* Jika ada, maka akan menghasilkan tidak ada error dan akan menutup port *Broadcast ID* karena sudah tidak digunakan untuk fungsi kedepannya, dan akan menyimpan pesan yang diterima sebagai nama dari *host* dan *address* dari *host* sebagai *IP Host.* Nilai *ACK False* diisi dengan nilai *TRUE.* Pada Keadaan ini *client* akan berpindah state menuju *state Wait Command.*



Gambar 5.31 Kode program *state ACK client*

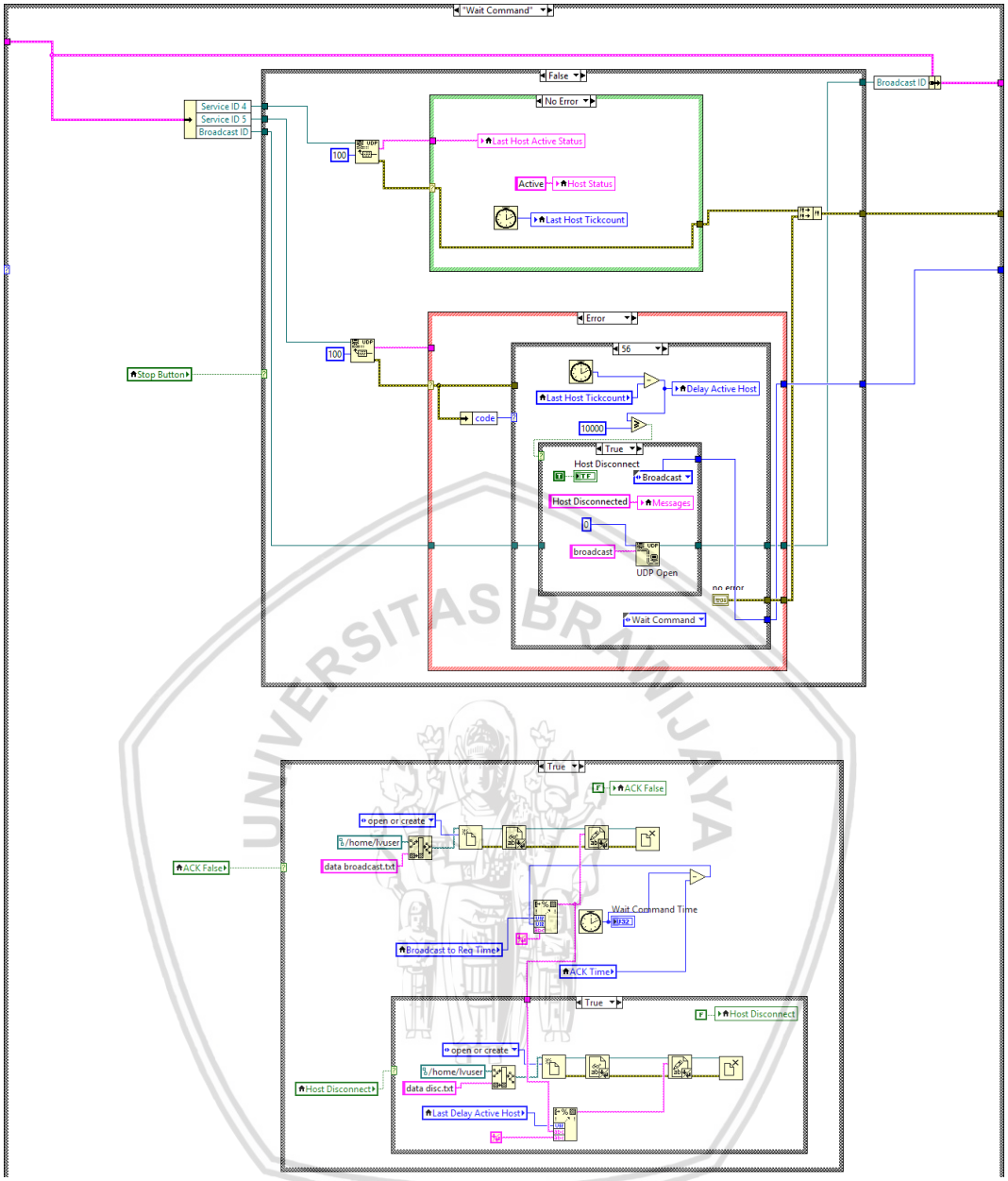
Jika *client* tidak menerima pesan balasan dari *host,* maka fungsi *UDP Read* menghasilkan *error* dengan kode 56. Pada keadaan ini, *client* akan kembali pada *state Broadcast*. Kode program dapat dilihat pada Gambar 5.32. Pada *state ACK, client* akan mencatat waktu pada saat *client* masuk pada *state ACK*.



Gambar 5.32 Kode program *state ACK client* gagal

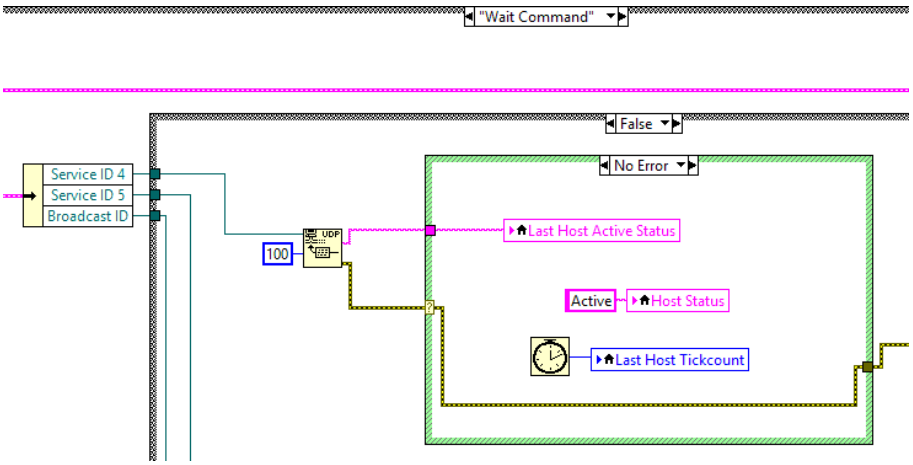
5.2.2.4 Implementasi state Wait Command

Pada keadaan ini, *client* akan menunggu perintah dari *host* untuk melakukan *request* data. Pada *State Wait Command* terdapat beberapa fungsi, diantaranya menerima status *host* aktif, menerima pesan jika *host* ingin melakukan *request,* dan pencatatan waktu yang dibutuhkan saat *client* berada pada *state Broadcast* hingga *state Wait Command.* Kode program *state Wait Command* dapat dilihat pada Gambar 5.33.



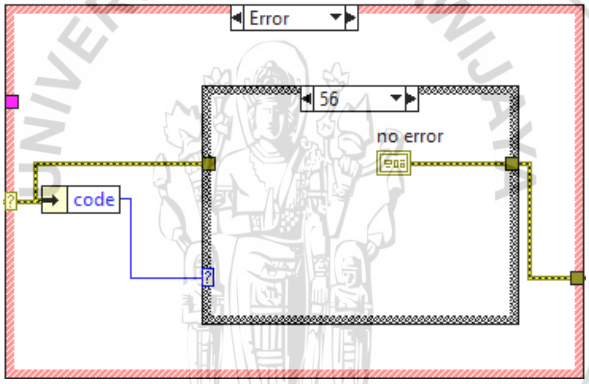
Gambar 5.33 Kode program *state Wait Command*

Fungsi yang pertama yaitu *client* menerima pesan status aktif *host*. Pada keadaan ini *client* akan menerima pesan dari *host* yang berisikan pesan *timestamp* dari *host* melalui *Service ID 4.* Jika *client* menerima pesan, maka akan menghasilkan *No Error* dan akan menyimpan pesan pada *Last Host Active Status*, selain itu *client* akan mencatat waktu sebagai waktu terakhir *host* aktif dengan fungsi *Tick Count (ms)*, dan menampilkan pesan sistem *Active* pada *Host Status*. Kode program dapat dilihat pada Gambar 5.34.



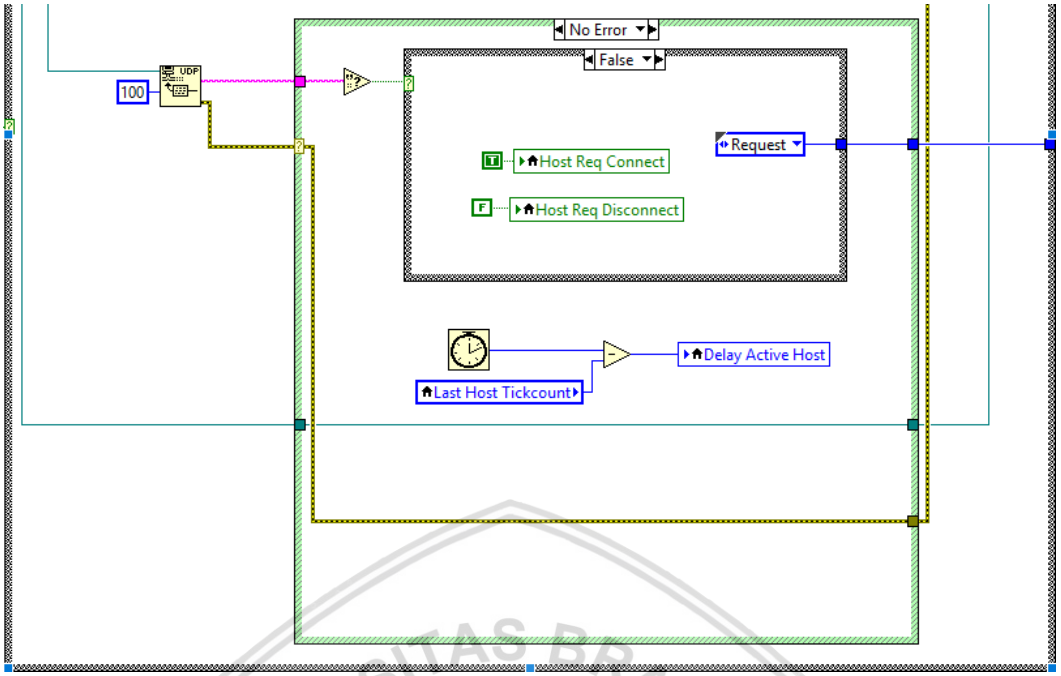
Gambar 5.34 Kode program menerima status aktif *host*

Jika *client* tidak menerima pesan status aktif *host,* maka *client* akan masuk pada kondisi *Error* dengan kode 56 dan kemudian menganggap pesan tersebut tidak ada *error*. *Client* tidak akan mencatat waktu sebagai waktu terakhir *host* aktif. Kode program dapat dilihat pada Gambar 5.35.

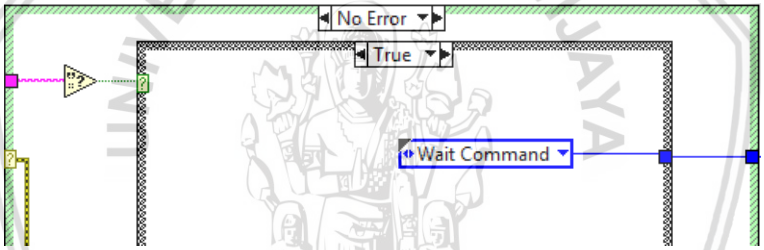


Gambar 5.35 Kode program menerima status aktif *host error*

Fungsi selanjutnya yaitu *client* menerima pesan *request* dari *host*. Pada keadaan ini *client* akan menunggu *host* untuk mengirimkan pesan *request.* Port yang digunakan yaitu melalui *Service ID 5*. Jika *client* menerima pesan dari *host,* maka *client* akan mengecek apakah pesan yang dikirimkan kosong, dan jika pesan tidak kosong atau *host* mengirimkan pesan berisikan *connect,* maka akan masuk pada keadaan *False* dimana *client* selanjutnya akan berpindah *state Request* dan menandai *Host Req Connect* sebagai *TRUE* yang artinya *host* telah terhubung dan menandai *Host Req Disconnect* dengan nilai *FALSE* yang kode programnya dapat dilihat pada Gambar 5.36. Jika *client* menerima pesan kosong, maka *client* akan tetap berada pada *state Wait Command* yang dapat dilihat pada Gambar 5.37. Selain itu, *client* ini akan mencatat waktu saat ini dan dikurangi dengan waktu *Last Host Tickcount* yang telah didapatkan pada fungsi sebelumnya sehingga menghasilkan waktu *delay* antara waktu saat ini dan waktu terakhir *host* aktif, yang disimpan pada *Delay Active Host.*

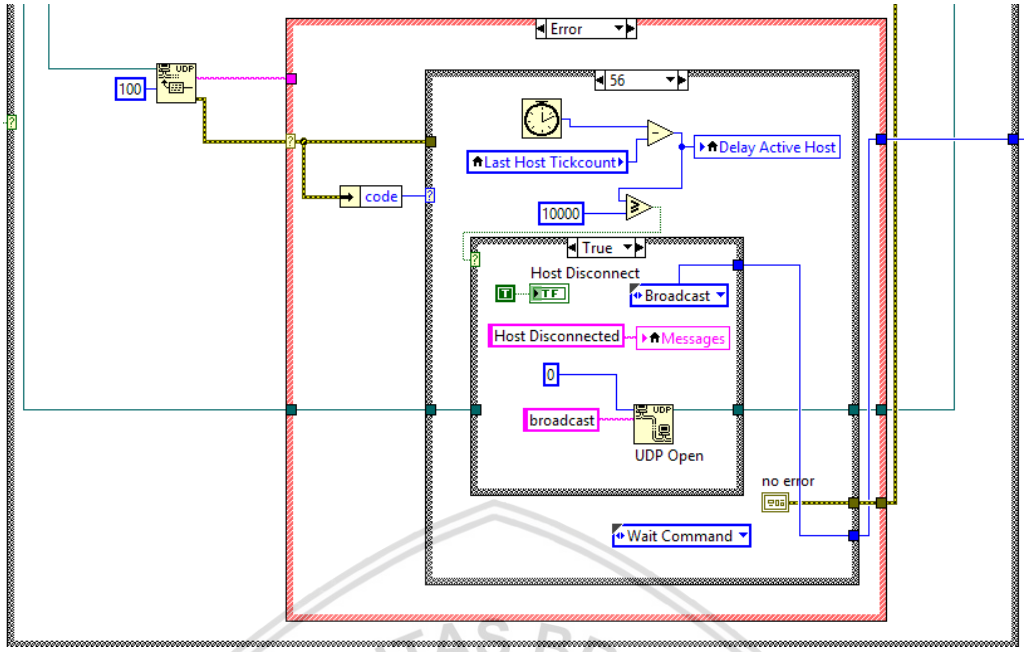


Gambar 5.36 Kode program menerima *request* dari *host*

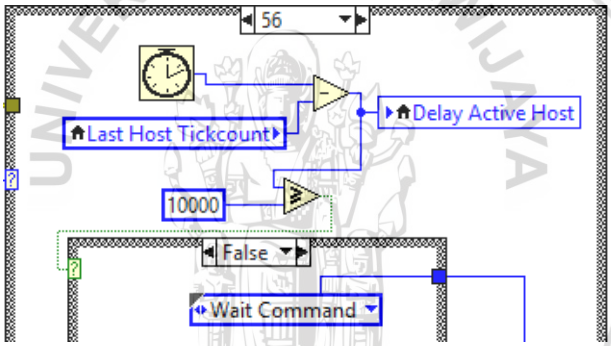


Gambar 5.37 Kode program menerima pesan *request* kosong

Jika *client* tidak menerima pesan *request* dari *host,* maka *client* akan masuk pada kondisi *error* dengan kode 56. *Client* akan mencatat waktu saat ini dan dikurangi dengan waktu *host* terakhir aktif yang dicatat pada fungsi sebelumnya. Jika *delay* antara *host* aktif dan waktu saat ini melebihi waktu yang ditentukan yaitu 10 detik atau 10000ms, maka *client* akan kembali pada *state Broadcast* dengan membuka kembali port *Broadcast ID* yang telah ditutup sebelumnya, selain itu *client* akan menandai jika *host* telah terputus dengan memberi nilai *Host Disconnect* dengan nilai *TRUE* dan menampilkan pesan sistem *Host Disconnected.* Kode program dapat dilihat pada Gambar 5.38. Jika waktu *delay* antara waktu saat ini dan waktu *host* terakhir aktif tidak lebih dari 10 detik, maka *client* akan tetap pada *state Wait Command* yang dapat dilihat pada Gambar 5.39.



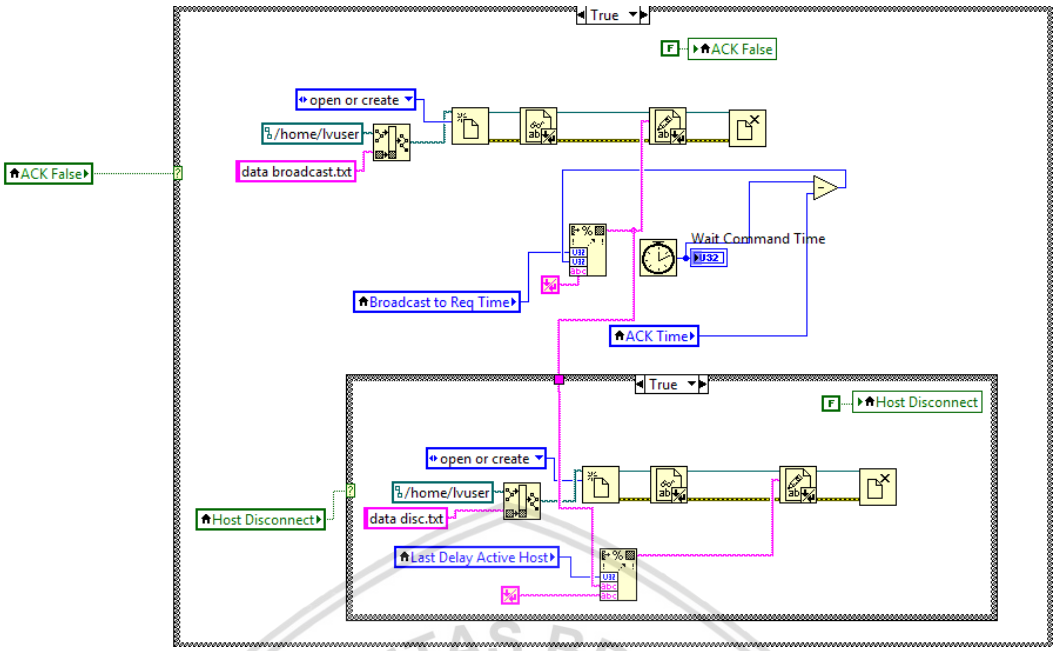
Gambar 5.38 Kode program menerima *request* dari *host error*



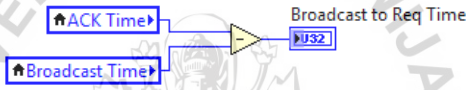
Gambar 5.39 Kode program *delay host* tidak terpenuhi

Fungsi selanjutnya yaitu pencatatan waktu *delay* pada *state Wait Command*. Jika *ACK False* bernilai *TRUE,* maka waktu akan dicatat yang diantaranya waktu yang dibutuhkan *client* dari *state Broadcast* menuju *state Wait Command.* Waktu tersebut didapatkan dari pengurangan pencatatan waktu pada *state ACK* dan *state Broadcast* sehingga menghasilkan nilai *Broadcast to Req Time* yang dapat dilihat pada Gambar 5.41, kode program ini terdapat diluar *state* dan berada di dalam perulangan *while*. Kemudian pengurangan pencatatan waktu pada *state Wait Command* dan *ACK* yang dapat dilihat pada gambar Gambar 5.40. Kedua waktu *delay* tersebut disimpan pada file dengan nama *data broadcast.txt.* Kode program dapat dilihat pada Gambar 5.40.

Fungsi lainnya yaitu jika *host* terputus dari *client,* maka *client* akan mencatat waktu seberapa lama *host* terputus dari *client.* Jika nilai *Host Disconnected* bernilai *TRUE,* maka *client* akan mencatat waktu *delay state client* yang dibutuhkan sebelumnya dengan waktu *host* terputus dengan *client* dan disimpan pada file dengan nama *data disc.txt* yang dapat dilihat pada Gambar 5.40.



Gambar 5.40 Kode program mencatat waktu *delay*



Gambar 5.41 Kode program *delay state Broadcast* menuju *state ACK*

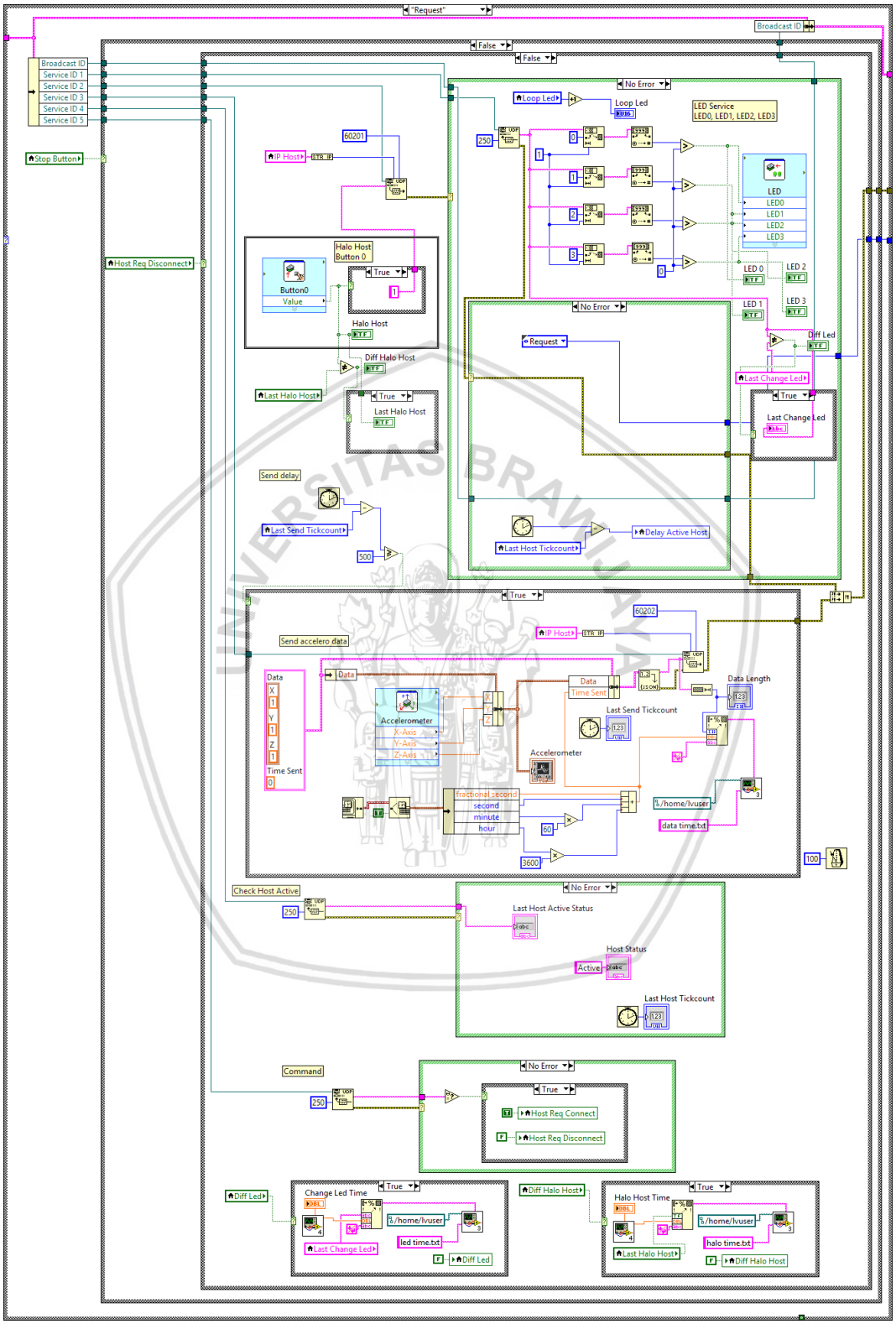
Fungsi terakhir yaitu jika tombol *Stop Button* ditekan pada saat *state Wait Command*. Jika tombol ditekan, *client* akan berpindah menuju *state Stop.* Kode program dapat dilihat pada Gambar 5.42.

Gambar 5.42 Kode program saat tombol *Stop Button* ditekan

5.2.2.5 Implementasi state Request

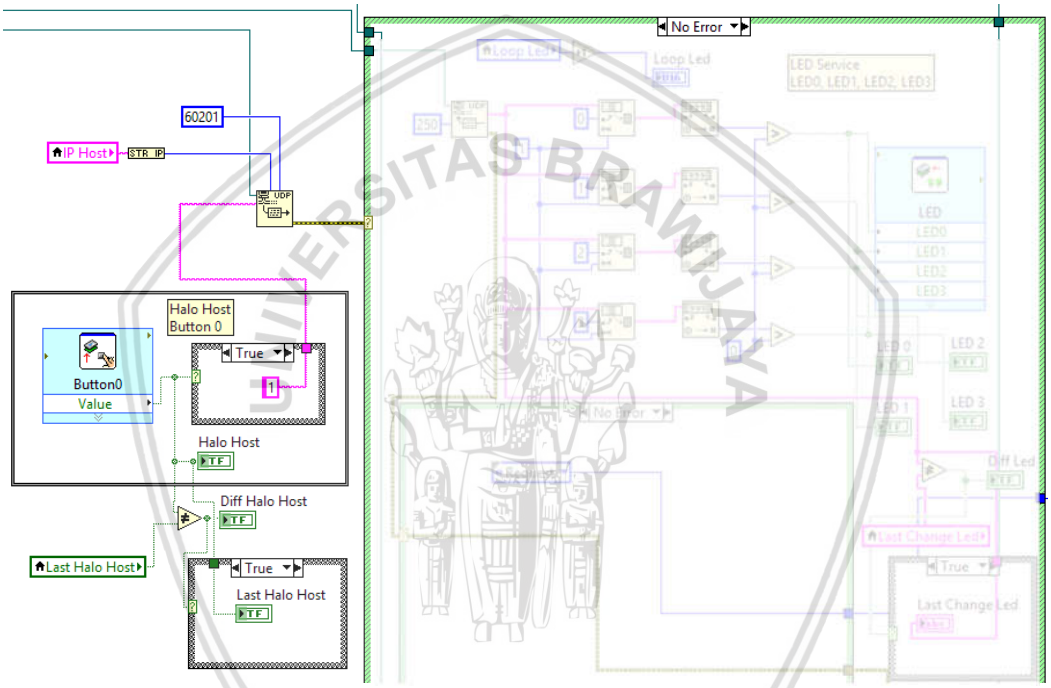
Pada *state* ini *client* akan mengirimkan data sensor *accelerometer, state push button,* dan menerima perintah LED untuk menghidupkan LED pada *client.* Pada Gambar 5.43 Kode program *state Request client*. *State Request* terdapat beberapa fungsi diantaranya yaitu penerimaan perintah LED, pengiriman *state push button,*

pengiriman data sensor *accelerometer*, menerima pesan aktif *host,* menerima pesan *disconnect,* dan pencatatan waktu.

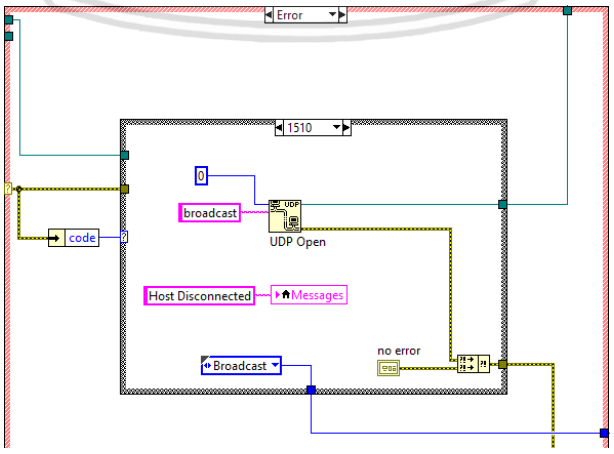


Gambar 5.43 Kode program *state Request client*

Fungsi yang pertama yaitu pengiriman *state push button* pada *host. Halo Host* merupakan fungsi *push button* pada *client. Client* akan mengirimkan *state push button* dengan mengambil fungsi *Button0* yang ada pada *client* melalui *Service ID 2,* jika pengiriman data sukses maka akan menjalankan fungsi selanjutnya yaitu penerimaan *state LED* dari *host*. Sedangkan jika pengiriman gagal, maka akan terdapat error dengan kode 1510 dimana *host* tidak aktif, dan *client* akan kembali pada *state Broadcast* dengan membuka kembali port *Broadcast ID* dan menampilkan pesan sistem *Host Disconnected* yang dapat dilihat pada Gambar 5.45. Kode program pengiriman *state push button* dapat dilihat pada Gambar 5.44. *Client* juga akan menyimpan *state* terakhir *push button,* jika terjadi perbedaan *state* saat ini dengan *state* terakhir maka *client* akan menandakan sebagai perbedaan *state* yang nantinya digunakan untuk pencatatan waktu.

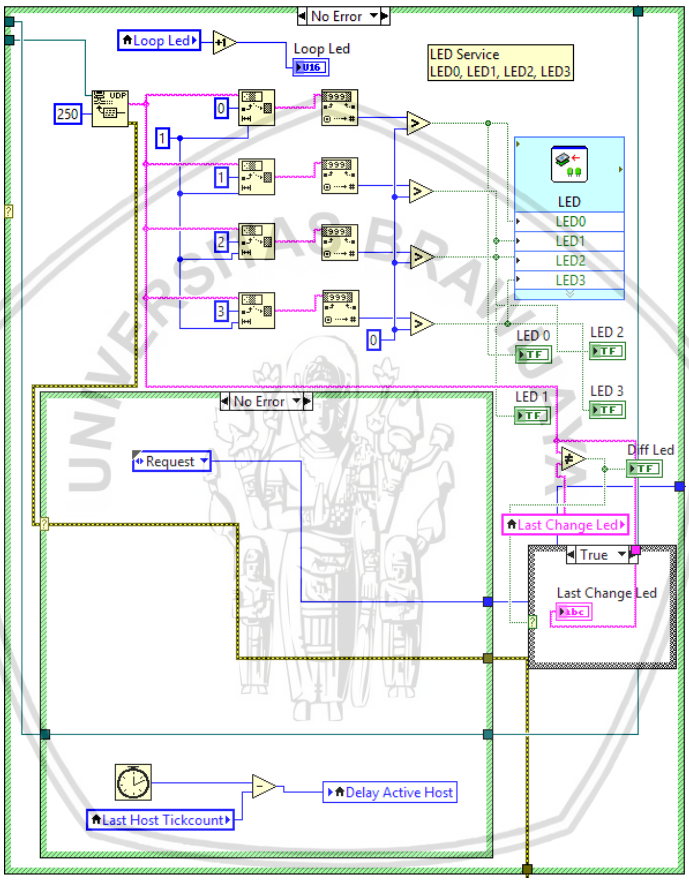


Gambar 5.44 Kode program pengiriman *state push button*



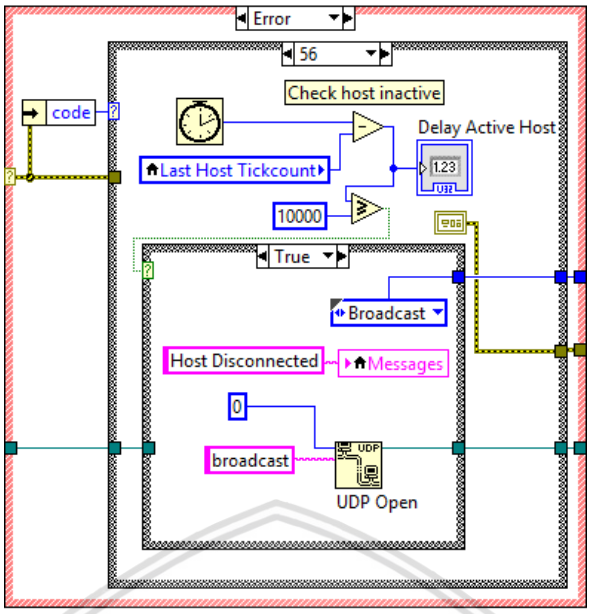
Gambar 5.45 Kode program *host* terputus

Fungsi selanjutnya yaitu menerima *state LED* dari *host* yang akan diteruskan untuk menghidupkan LED pada *client*. *Client* akan menerima pesan berisikan *state LED* melalui *Service ID 1*, kemudian pesan tersebut akan dibagi menjadi empat bagian untuk menghidupkan masing-masing LED pada *client.* Fungsi untuk menghidupkan LED pada *client* yaitu fungsi *LED* dan akan memberi nilai pada LED 0, LED 1, LED 2, LED 3 yang berada pada antarmuka LabVIEW *client. Client* juga akan mencatat *state* terakhir dari LED, dimana jika terjadi perbedaan dengan *state* saat ini maka akan tercatat sebagai perbedaan *state LED* yang disimpan pada *Diff Led.* Perbedaan *state LED* ini akan digunakan untuk pencatatan waktu. Kode program dapat dilihat pada Gambar 5.46.

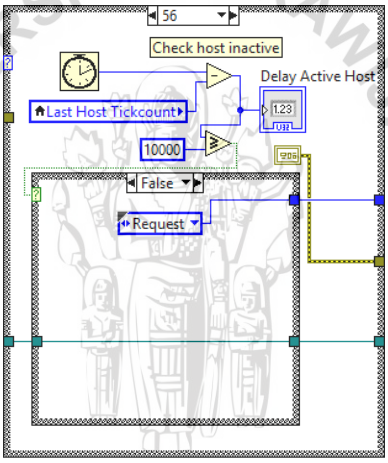


Gambar 5.46 Kode program menerima *state LED*

Jika pada keadaan *client* menerima *state LED* tidak mengalami error dimana *client* sukses menerima pesan *state LED* dari *host,* maka *client* akan tetap berada pada *state Request,* dan akan menyimpan waktu saat ini yang dikurangi dengan waktu terakhir *host* aktif. Jika mengalami error dengan kode 56, maka *client* akan mengecek waktu *delay* antara *host* tidak aktif dengan waktu saat ini. Jika waktu lebih dari 10 detik, maka *client* akan kembali menuju *state Broadcast* dengan membuka kembali port *Broadcast ID* dan menampilkan pesan sistem *Host Disconnected.* Kode program dapat dilihat pada Gambar 5.47. Jika *delay* wakti tidak melebihi 10 detik, maka *client* akan tetap pada *state Request* yang dapat dilihat pada Gambar 5.48.



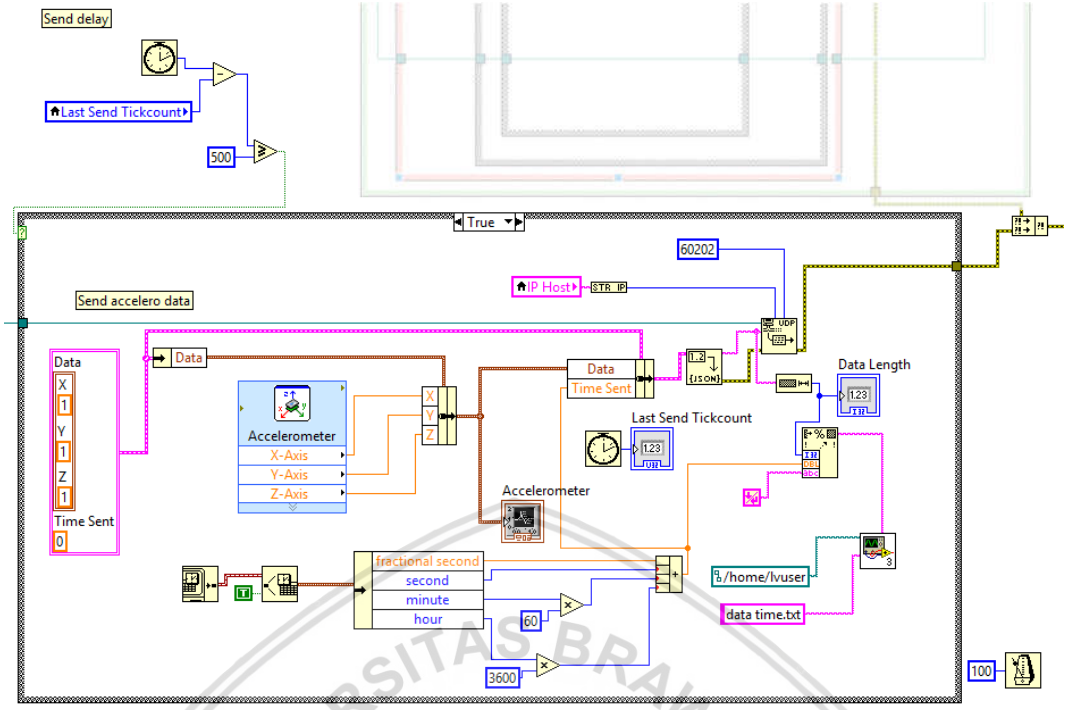
Gambar 5.47 Kode program menerima *state LED error*



Gambar 5.48 Kode program *delay host* tidak melebihi 10 detik

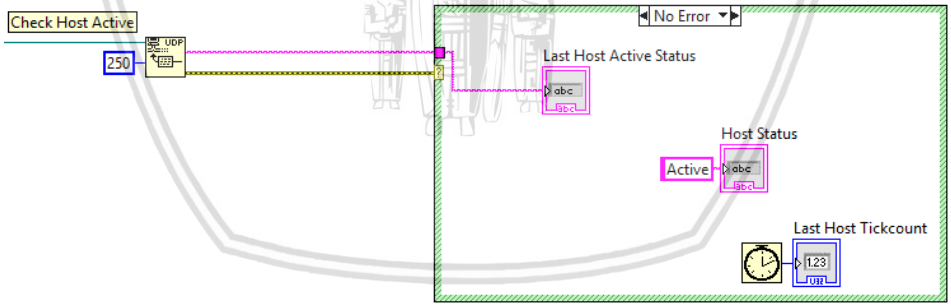
Fungsi selanjutnya yaitu pengiriman data sensor *accelerometer* pada *host*. *Client* akan mengambil data melalui fungsi *Accelerometer* yang dimiliki oleh *client* dengan hasil X, Y, dan Z. Data tersebut kemudian di-*bundle* menjadi satu sehingga dapat ditampilkan pada grafik *Accelerometer.* Setelah data di-*bundle,* data di-*bundle* kembali dengan waktu pengambilan data saat ini. Setelah data *accelerometer* dan *waktu* pengambilan data di-bundle, data ini akan dirubah menjadi format json dengan fungsi *flatten to JSON.* Setelah data berbentuk json, data akan dikirim melalui *Service ID 3* kepada *host* dengan port 60202. *Client* akan mengirimkan data sensor *accelerometer* setiap 500ms.

Selain fungi pengiriman data sensor *accelerometer, client* akan mencatat waktu pengiriman data beserta panjang data yang dikirimkan. Data tersebut akan disimpan pada file dengan nama *data time.txt.* Kode program pengiriman data sensor *accelerometer* dapat dilihat pada Gambar 5.49.



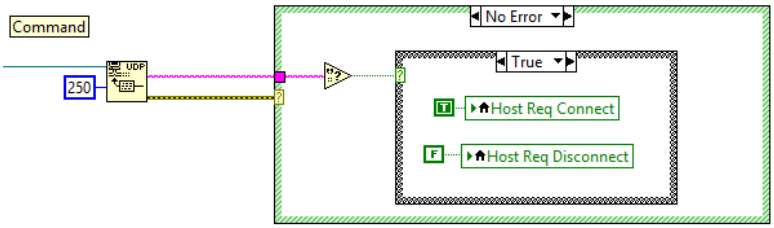
Gambar 5.49 Kode program pengiriman data sensor *accelerometer*

Fungsi selanjutnya yaitu menerima status aktif *host. Client* akan menerima pesan berisikan *timestamp* waktu aktif *host* saat ini, melalui port *Service ID 4.* Fungsi ini akan menampilkan pesan sistem waktu *host* aktif dan pesan *Active,* selain itu *client* akan mencatat waktu saat ini sebagai waktu *host* terakhir aktif. Kode program dapat dilihat pada Gambar 5.50.

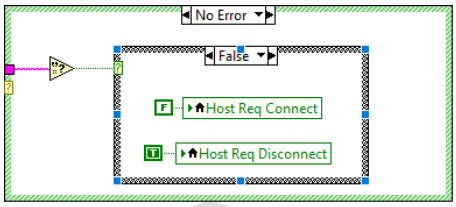


Gambar 5.50 Kode program penerimaan status aktif *host*

Fungsi selanjutnya yaitu *client* menerima pesan *disconnect* dari *host*. *Client* akan menunggu apakah *host* ingin mengakhiri untuk mengendalikan dan pemantauan *client.* Jika pesan tersebut kosong, maka nilai dari *Host Req Connect* dan *Host Req Disconnect* masih tetap sama yang dapat dilihat pada Gambar 5.51. Jika pesan tersebut berisikan pesan *disconnect,* maka nilai dari *Host Req Connect* menjadi *FALSE* dan *Host Req Disconnect* menjadi *TRUE* yang dapat dilihat pada gambar Gambar 5.52.

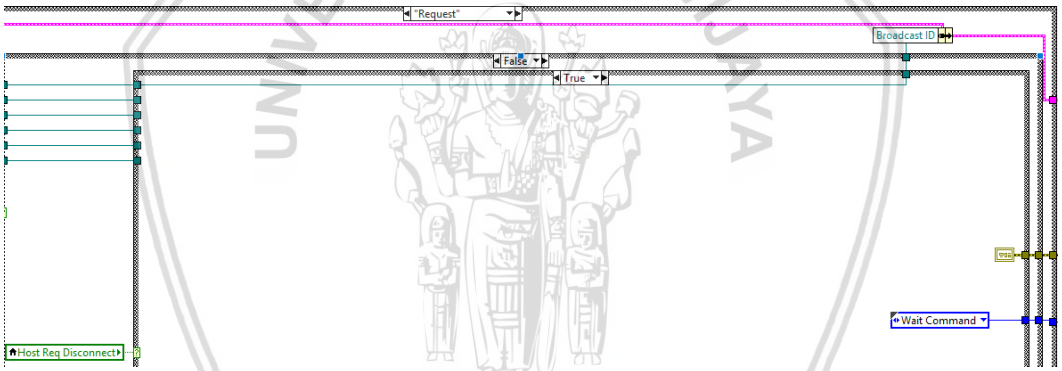


Gambar 5.51 Kode program tidak menerima pesan *disconnect*



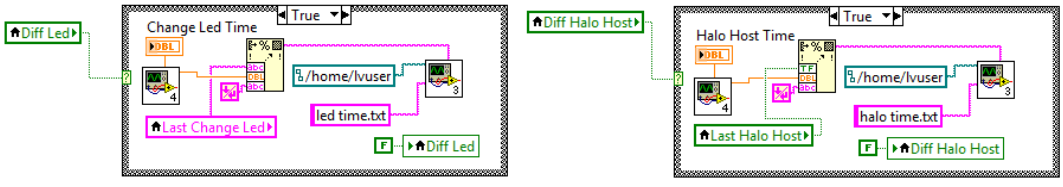
Gambar 5.52 Kode program penerimaan pesan *disconnect*

Fungsi selanjutnya yaitu ketika nilai dari *Host Req Disconnect* bernilai *TRUE.* Pada keadaan ini *client* akan berpindah pada *state Wait Command* yang bertanda bahwa *host* telah mengakhiri untuk mengendalikan dan memantau *client.* Kode program dapat dilihat pada Gambar 5.53.



Gambar 5.53 Kode program *Host Req Disconnect* bernilai *TRUE*

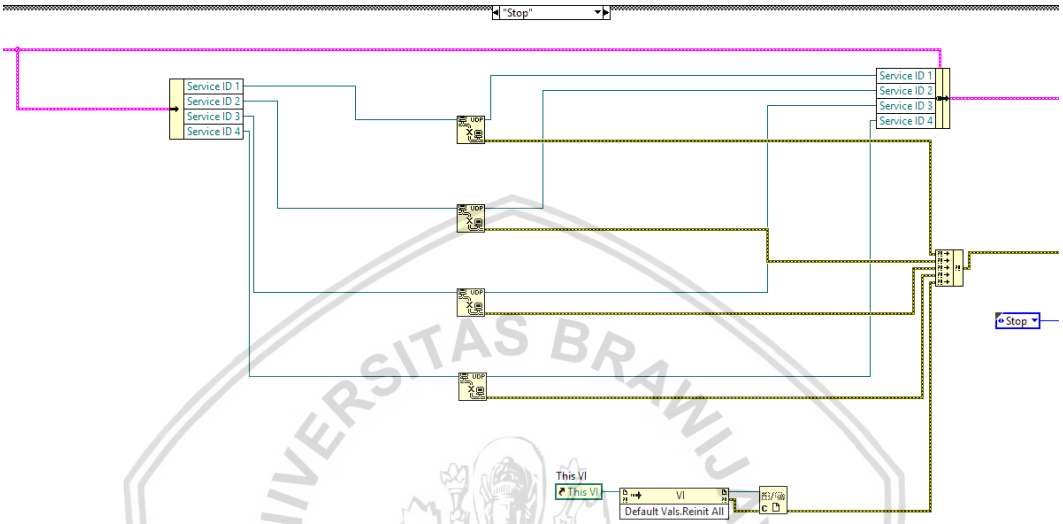
Fungsi selanjutnya yaitu pencatatan waktu saat ada perubahan *state LED* dan *state push button.* Fungsi untuk mengambil waktu yaitu *sub-vi* yang berisikan kode program untuk mengambil waktu saat ini. Jika ada perbedaan *state LED* yang diwakili nilai dari *Diff Led,* maka *client* akan menyimpan waktu saat ini dan *state LED* yang terakhir dan disimpan dengan nama *led time.txt.* Sedangkan jika ada perbedaan *state push button* yang diwakili nilai dari *Diff Halo Host,* maka *client* akan menyimpan waktu saat ini dan *state push button* yang terakhir dan disimpan dengan nama *halo time.txt.* Kode program dapat dilihat pada Gambar 5.54.



Gambar 5.54 Kode program pencatatan waktu *state LED* dan *state push button*

5.2.2.6 Implementasi state Stop

State machine client selanjutnya yaitu *state Stop.* Pada keadaan ini *client* akan menutup semua port yang telah dibuka pada awal inisialisasi diantaranya *Service ID 1, Service ID 2, Service ID 3,* dan *Service ID 4.* Selain itu akan me-*reset* tampilan antarmuka seperti awal. Setelah itu akan mengirimkan nilai *True* untuk menghentikan perulangan pada perulangan *while*. Kode program *state Stop* dapat dilihat pada Gambar 5.55.



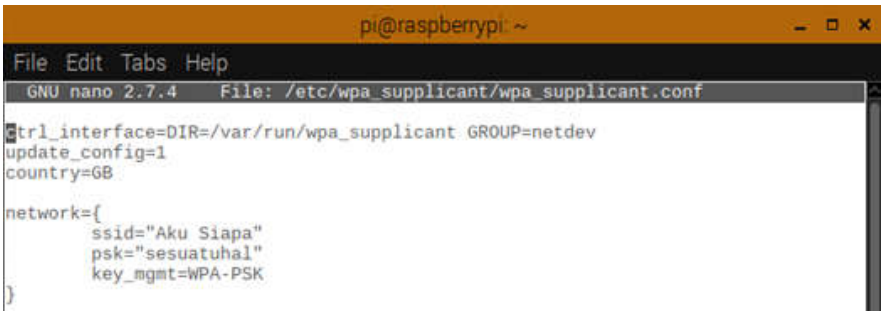
Gambar 5.55 Kode program *state Stop client*

5.2.3 Implementasi Koneksi Perangkat

Implementasi dilakukan pada perangkat, diantaranya yaitu *host* pada Raspberry Pi 3 dan *client* pada NI myRIO. Pada sub-sub-bab selanjutnya akan dijelaskan cara konfigurasi agar perangkat dapat terhubung pada jaringan. Jaringan *hotspot* pada *smartphone* telah diaktifkan sebelumnya.

5.2.3.1 Konfigurasi Raspberry Pi 3 sebagai host

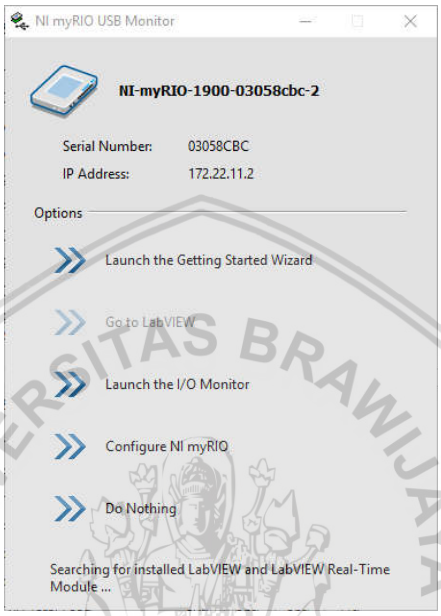
Langkah pertama yang dilakukan yaitu buka *terminal* pada Raspberry Pi 3. Lalu buka `/etc/wpa_supplicant/wpa_supplicant.conf` menggunakan aplikasi *nano.* Kemudian tambahkan informasi *hotspot* dengan awalan *network* seperti Gambar 5.56.



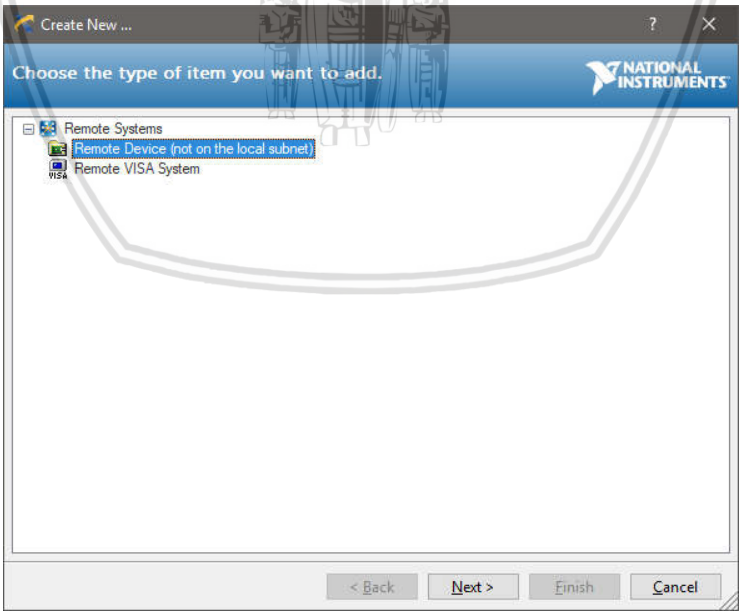
Gambar 5.56 Konfigurasi Raspberry Pi 3

5.2.3.2 Konfigurasi NI myRIO sebagai Client

Langkah pertama yaitu hubungkan perangkat NI myRIO dengan laptop dengan kabel USB, sehingga muncul jendela seperti pada Gambar 5.57. Kemudian buka aplikasi NI MAX dan klik kanan pada *Remote Systems* dan pilih *Create New...* sehingga muncul jendela seperti pada Gambar 5.58 dan pilih *Remote Device (not onthe local subnet).* Selanjutnya masukkan IP Address yang ada pada Gambar 5.57. Kemudian klik *Finish.*



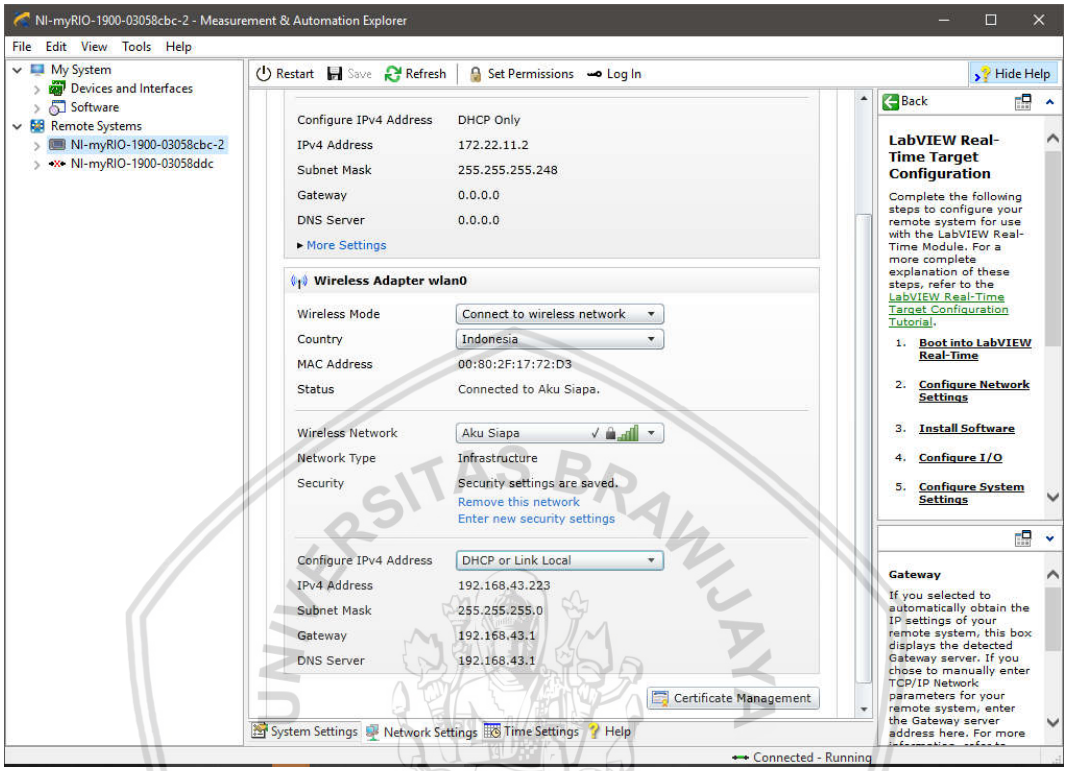
Gambar 5.57 Window NI myRIO USB Monitor



Gambar 5.58 Window Create New Remote Systems

Kemudian pilih perangkat yang baru ditambahkan pada sebelah kiri jendela di dalam *Remote Systems*. Setelah itu pilih tab *Network Settings* yang ada pada bawah jendela. Setelah itu pilih *Wireless Mode* dengan mode *Connect to wireless*

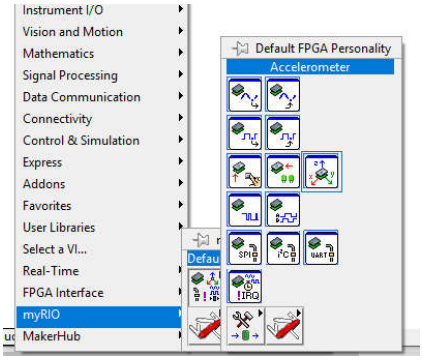
network, kemudian pilih jaringan yang ingin dihubungkan dan masukkan password jika dibutuhkan. Untuk *Configure IPv4 Address* bisa memilih mode *Static* untuk konfigurasi IP manual atau *DHCP or Link Local* untuk *IP dynamic.* Kemudian tekan *save* yang terletak di jendela atas. Jendela NI MAX dapat dilihat pada Gambar 5.59.



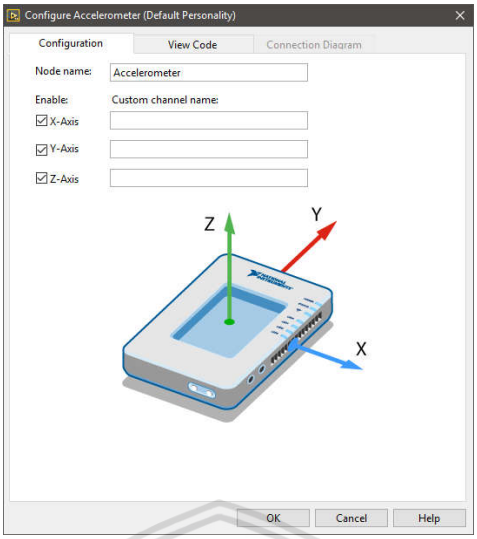
Gambar 5.59 Window NI MAX

5.2.4 Implementasi Sensor Accelerometer Client

Pada implementasi kali ini, akan menjelaskan bagaimana cara menggunakan sensor *Accelerometer* pada NI myRIO. Langkah pertama yaitu buat *vi* baru pada target NI myRIO. Kemudian di jendela *block diagram* klik kanan dan pilih myRIO, kemudian pilih Default FPGA Personality dan terakhir pilih Accelerometer seperti pada Gambar 5.60. Kemudian akan muncul jendela konfigurasi *accelerometer* seperti Gambar 5.61 dan klik OK.

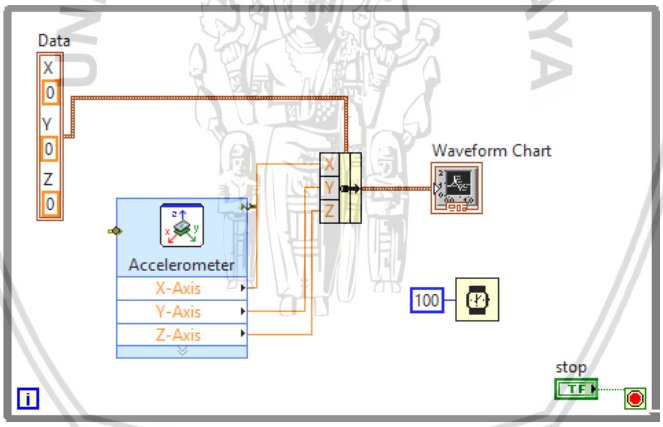


Gambar 5.60 Pemilihan fungsi *Accelerometer*

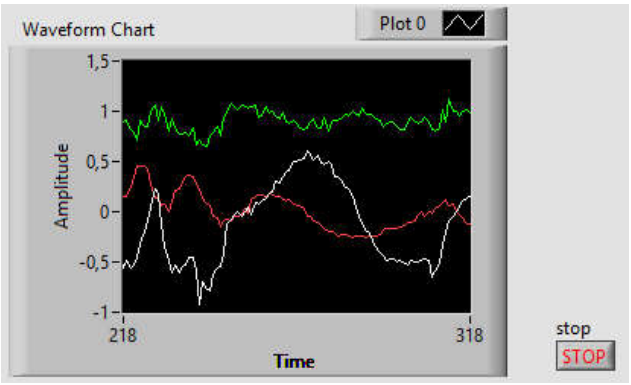


Gambar 5.61 Konfigurasi *Accelerometer*

Jika konfigurasi telah selesai, maka akan muncul fungsi *Accelerometer* dengan tiga keluaran yaitu X-Axis, Y-Axis, dan Z-Axis. Keluaran tersebut di-bundle menjadi satu agar tiga keluaran sebelumnya dapat ditampilkan pada grafik secara bersamaan, dengan input cluster konstanta tiga buah variabel *double*. Kode program dapat dilihat pada Gambar 5.62, dan hasil keluaran dapat dilihat pada



Gambar 5.62 Kode program pembacaan sensor *accelerometer*



Gambar 5.63 Keluaran grafik sensor *accelerometer*

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab kali ini akan dijelaskan mengenai pengujian dan analisa hasil dari perancangan dan implementasi yang sudah dilakukan sebelumnya. Pada pengujian penelitian kali ini, akan dijelaskan mengenai alur dan metode pengujian yang akan dilakukan. Sedangkan pada analisa akan dijelaskan hasil dari pengujian dan analisa hasil sistem tersebut.

6.1 Pengujian *Host* Mendeteksi *Client*

6.1.1 Tujuan pengujian

Tujuan pengujian kali ini untuk mengetahui keberhasilan Raspberry Pi 3 sebagai *host* untuk mendeteksi NI myRIO sebagai *client* baru. Tingkat keberhasilan akan dilihat dari berhasilnya *host* untuk menyimpan informasi yang diterima oleh *client*. Pengujian *Discovery* Perangkat.

6.1.2 Prosedur Pengujian

Prosedur pengujian pada penelitian kali ini adalah sebagai berikut:

1. *Deploy* program *host* pada Raspberry Pi 3 hingga pada kondisi *state Listen*.
2. *Deploy* program *client* pada NI MyRIO hingga pada kondisi *state Broadcast* menuju *state Wait Command*. Dilakukan pada semua *client*.
3. Perhatikan pada list *client* yang tersimpan pada antarmuka *host* pada LabVIEW.
4. Menghentikan program pada *client* dan melakukan pengujian ulang dimulai pada *step 2.* Dilakukan secara bergantian pada semua *client*.

6.1.3 Hasil Pengujian

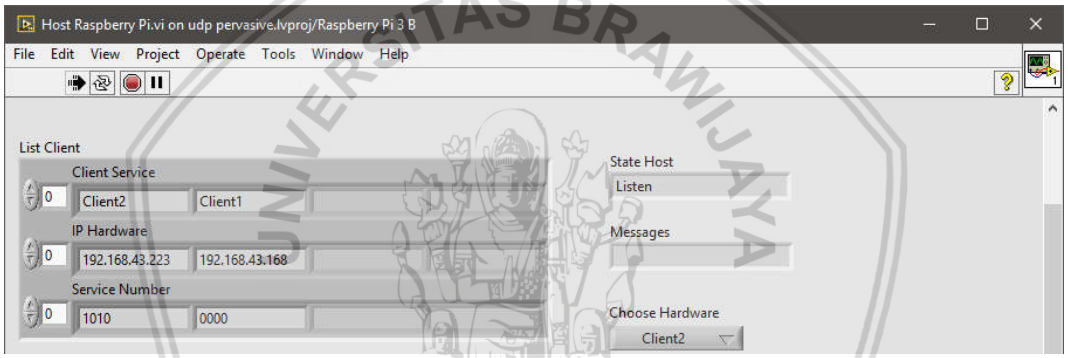
Pengujian dilakukan pada tiga perangkat, satu buah *host* dan dua buah *client*. *Client* akan terhubung dengan *host* dengan fitur yang berbeda-beda yang ditawarkan oleh *client.* Fitur yang ditawarkan diantaranya pada indeks 0 yaitu LED 0 dan LED 1, indeks 1 yaitu LED 2 dan 3, indeks 2 yaitu sensor accelerometer, dan indeks 3 tidak ada fitur yang ditawarkan. Hasil pengujian dapat dilihat pada Tabel 6.1.

Tabel 6.1 Pengujian *host* mendeteksi *client*

Pengujian ke-	Perangkat	IP *client*	Fitur yang ditawarkan	Berhasil
1	Client1	192.168.43.168	0000	Sukses
2	Client1	192.168.43.168	1000	Sukses
3	Client1	192.168.43.168	0100	Sukses
4	Client1	192.168.43.168	1100	Sukses
5	Client1	192.168.43.168	0010	Sukses
6	Client1	192.168.43.168	0110	Sukses
7	Client1	192.168.43.168	1010	Sukses

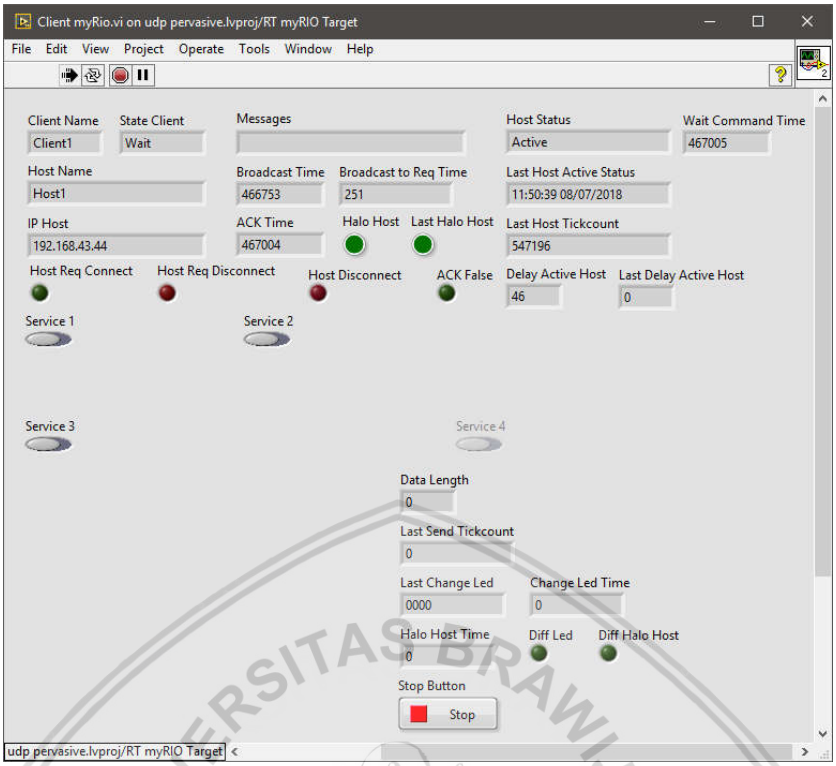
Pengujian ke-	Perangkat	IP *client*	Fitur yang ditawarkan	Berhasil
8	Client1	192.168.43.168	1110	Sukses
1	Client2	192.168.43.223	0000	Sukses
2	Client2	192.168.43.223	1000	Sukses
3	Client2	192.168.43.223	0100	Sukses
4	Client2	192.168.43.223	1100	Sukses
5	Client2	192.168.43.223	0010	Sukses
6	Client2	192.168.43.223	0110	Sukses
7	Client2	192.168.43.223	1010	Sukses
8	Client2	192.168.43.223	1110	Sukses

Host akan mendeteksi dan menyimpan informasi *client* pada antarmuka *List Client.* Pada Gambar 6.1 Antarmuka host *mendeteksi* client. Pada gambar tersebut, *host* sukses untu menyimpan informasi *client* dengan IP dan fitur *(Service Number)* yang berbeda-beda.

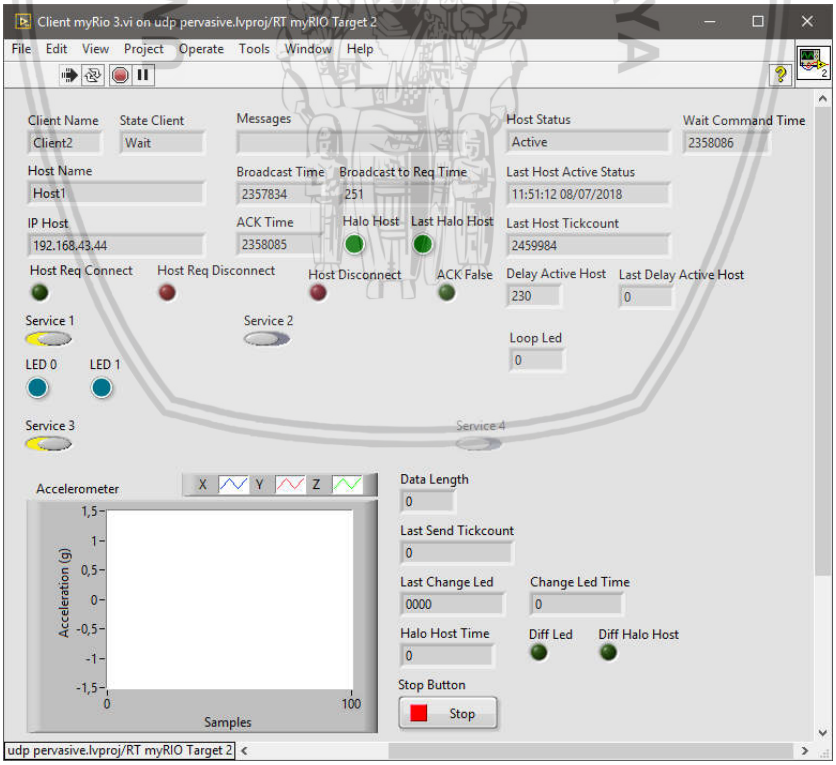


Gambar 6.1 Antarmuka *host* mendeteksi *client*

Pada Gambar 6.2 dan Gambar 6.3 merupakan antarmuka *client.* Pada gambar terdapat beberapa pelayanan diantaranya *Service 1, Service 2, Service 3* dan *Service 4.*Pada Gambar 6.2 merupakan antarmuka *client1* dengan tidak ada fitur yang dimiliki, sedangkan pada Gambar 6.3 merupakan antarmuka *client2* dengan memiliki fitur yaitu *Service 1* dengan LED 0 dan LED 1 dan *Service 2* dengan sensor *accelerometer.*



Gambar 6.2 Antarmuka *client1*



Gambar 6.3 Antarmuka *client2*

6.1.4 Analisis Pengujian

Berdasarkan dari hasil pengujian pada Tabel 6.1, pengujian fungsional *host* mendeteksi dan menyimpan informasi *client* dapat berjalan dengan baik. Dari 16 kali pengujian diantaranya 8 kali pengujian ditiap *client, host* dapat mendeteksi *client* dengan baik. Pengujian dilakukan dengan *service client* yang berbeda-beda dan tidak ada kendala dari *host* untuk mendeteksi *client.* Dari hasil pengujian ini, maka didapatkan pengujian fungsional *host* mendeteksi dan menyimpan informasi *client* dengan tingkat keberhasilan 100%.

6.2 Pengujian Penyimpanan Informasi *Host* pada *Client*

6.2.1 Tujuan Pengujian

Tujuan pengujian kali ini untuk mengetahui keberhasilan NI myRIO sebagai *client* untuk menyimpan informasi host. Informasi tersebut diantaranya yaitu nama *host,* dan *IP Host.*

6.2.2 Prosedur Pengujian

Prosedur pengujian pada penelitian kali ini adalah sebagai berikut:

1. *Deploy* program *host* pada Raspberry Pi 3 hingga pada kondisi *state Listen*.
2. *Deploy* program *client* pada NI MyRIO hingga pada kondisi *state Broadcast* menuju *state Wait Command*. Dilakukan pada semua *client*.
3. Perhatikan pada *Host Name* dan *IP Host* yang tersimpan pada antarmuka *client* pada LabVIEW.
4. Menghentikan program pada *client* dan melakukan pengujian ulang dimulai pada *step 2.* Dilakukan secara bergantian pada semua *client*.

6.2.3 Hasil Pengujian

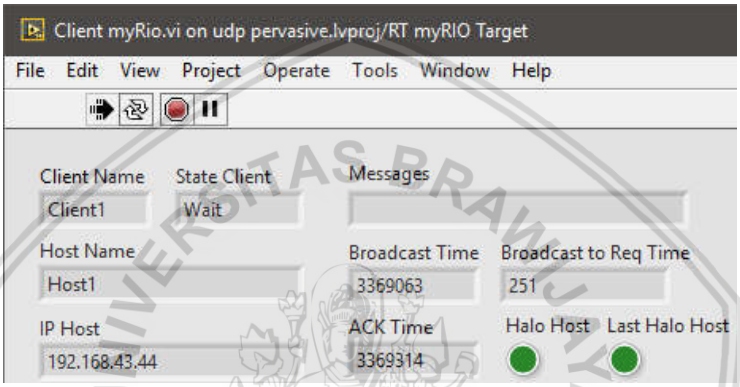
Pengujian dilakukan dengan memperhatikan antarmuka pada *client*. Pengujian dilakukan sebanyak 16 kali dimana terdapat 8 kali pengujian di tiap perangkatnya. Hasil pengujian dapat dilihat pada Tabel 6.2.

Tabel 6.2 Hasil pengujian *client* menyimpan informasi *host*

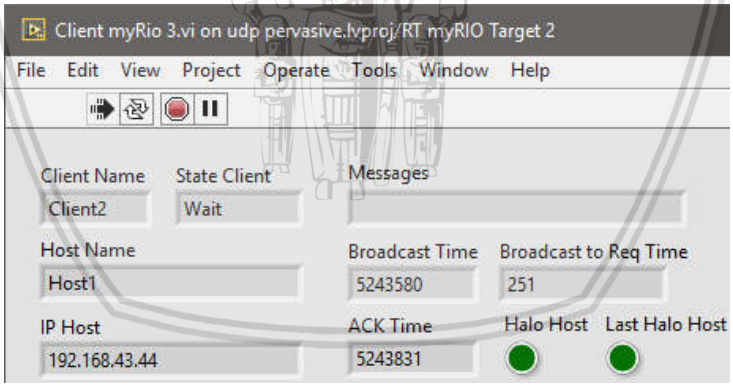
Pengujian ke-	Perangkat	Nama *host*	IP *host*	Berhasil
1	Client1	Host1	192.168.43.44	Sukses
2	Client1	Host1	192.168.43.44	Sukses
3	Client1	Host1	192.168.43.44	Sukses
4	Client1	Host1	192.168.43.44	Sukses
5	Client1	Host1	192.168.43.44	Sukses
6	Client1	Host1	192.168.43.44	Sukses
7	Client1	Host1	192.168.43.44	Sukses
8	Client1	Host1	192.168.43.44	Sukses
1	Client2	Host1	192.168.43.44	Sukses
2	Client2	Host1	192.168.43.44	Sukses

Pengujian ke-	Perangkat	Nama *host*	IP *host*	Berhasil
3	Client2	Host1	192.168.43.44	Sukses
4	Client2	Host1	192.168.43.44	Sukses
5	Client2	Host1	192.168.43.44	Sukses
6	Client2	Host1	192.168.43.44	Sukses
7	Client2	Host1	192.168.43.44	Sukses
8	Client2	Host1	192.168.43.44	Sukses

Pada Gambar 6.4 dan Gambar 6.5 merupakan antarmuka *client* pada LabVIEW. Pada gambar tersebut *client* dapat menyimpan informasi *host* yang berupa *Host Name* dan *IP Host*. Dimana *Host Name* yang terdeteksi yaitu Host1 dengan IP Host 192.168.43.44.



Gambar 6.4 Antarmuka *client1* mendeteksi *host*



Gambar 6.5 Antarmuka *client2* mendeteksi *host*

6.2.4 Analisis Pengujian

Berdasarkan dari hasil pengujian pada Tabel 6.2, pengujian fungsional *client* menyimpan informasi *host* yang terdeteksi dapat berjalan dengan baik. Dari 16 kali pengujian diantaranya 8 kali pengujian ditiap *client, client* dapat menyimpan informasi *host* dengan baik. Pengujian dilakukan dengan *Host Name* dan IP Host yang sama dan tidak ada kendala dari *client* untuk menyimpan informasi *host.* Dari hasil pengujian ini, maka didapatkan pengujian fungsional *client* yang menyimpan informasi *host* dengan tingkat keberhasilan 100%.

6.3 Pengujian Pengiriman Data Sensor

6.3.1 Tujuan Pengujian

Pengujian kali ini untuk mengetahui keberhasilan *client* untuk mengirimkan data sensor. Tingkat keberhasilan akan dilihat dari suksesnya *host* menerima data yang sama dengan *client*. Pengujian dilakukan 5 kali ditiap perangkatnya.

6.3.2 Prosedur Pengujian

Prosedur pengujian pada penelitian kali ini adalah sebagai berikut:

1. *Deploy* program *host* pada Raspberry Pi 3 hingga pada kondisi *state Listen*.
2. *Deploy* program *client* pada NI MyRIO hingga pada kondisi *state Broadcast* menuju *state Wait Command*. Dilakukan pada semua *client*.
3. Pada antarmuka *host,* pilih salah satu *client* untuk dipantau. Kemudian klik tombol Go Hardware. Dilakukan pada kedua perangkat.
4. Perhatikan data dan panjang data yang dikirimkan oleh *client* dan diterima oleh *host*.
5. Jika telah selesai, klik tombol Finish Hardware.

6.3.3 Hasil Pengujian

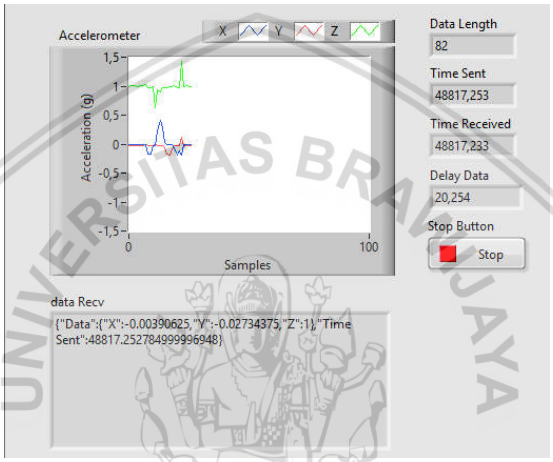
Pengujian dilakukan pada tiga perangkat, satu perangkat *host* dan dua perangkan *client.* Data yang diperhatikan yaitu data yang dikirimkan dan diterima kemudian panjang data. Hasil pengujian dapat dilihat pada Tabel 6.3.

Tabel 6.3 Hasil pengujian pengiriman data sensor

Pengujian ke-	Perangkat	Data	Panjang Data	Berhasil
1	Client1	{"Data":{"X":-0.01171875,"Y":-0.00390625,"Z":1.01171875},"Time Sent":47522.748850999996648}	91	Sukses
2	Client1	{"Data":{"X":-0.0078125,"Y":-0.01171875,"Z":1.01953125},"Time Sent":47535.289390999998432}	90	Sukses
3	Client1	{"Data":{"X":-0.015625,"Y":-0.01171875,"Z":1.015625},"Time Sent":47581.933301999997639}	87	Sukses
4	Client1	{"Data":{"X":-0.015625,"Y":-0.0078125,"Z":1.01171875},"Time Sent":47623.568631000001915}	88	Sukses
5	Client1	{"Data":{"X":-0.0078125,"Y":-0.01171875,"Z":1.01953125},"Time Sent":47639.122704000001249}	90	Sukses
1	Client2	{"Data":{"X":-0.015625,"Y":-0.0078125,"Z":1.0234375},"Time Sent":47648.153831999996328}	87	Sukses
2	Client2	{"Data":{"X":-0.0078125,"Y":-0.00390625,"Z":1.01171875},"Time Sent":47706.337562000000617}	90	Sukses
3	Client2	{"Data":{"X":-0.01171875,"Y":-0.01171875,"Z":1.01171875},"Time Sent":47714.861886999999115}	91	Sukses

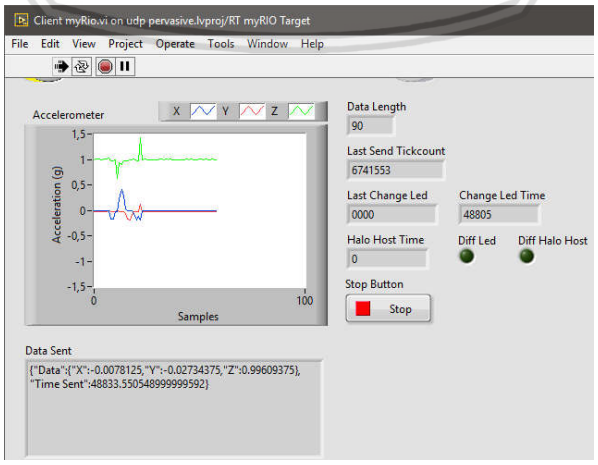
Pengujian ke-	Perangkat	Data	Panjang Data	Berhasil
4	Client2	{"Data":{"X":-0.0078125,"Y":-0.0078125,"Z":1.015625},"Time Sent":47723.893558999996458}	87	Sukses
5	Client2	{"Data":{"X":-0.01171875,"Y":-0.0078125,"Z":1.01953125},"Time Sent":47733.41998600000079}	89	Sukses

Pada Gambar 6.6 merupakan antarmuka *host* pada LabVIEW. Pada gambar tersebut *host* dapat menerima data sensor *accelerometer* dari *client.* Data direpresentasikan dengan bentuk grafik. Penjelasan grafik tersebut diantaranya X merupakan X-axis *accelerometer*, Y merupakan Y-axis *accelerometer*, dan Z merupakan Z-axis *accelerometer*. Selain itu terdapat Data Length yang merupakan panjang data, dan data Recv merupakan data dengan bentuk json.

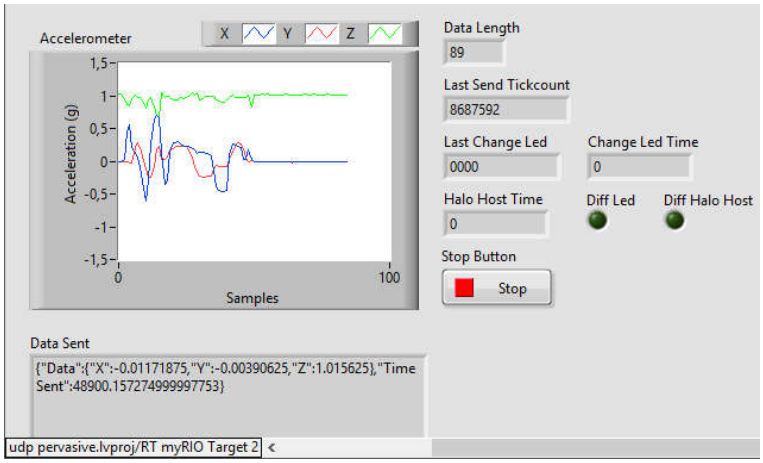


Gambar 6.6 Antarmuka *host* menerima data

Pada Gambar 6.7 dan Gambar 6.8 merupakan antarmuka *client*. Pada gambar tersebut *client* sedang mengirimkan data sensor *accelerometer.* Data tersebut direpresentasikan pada bentuk grafik. Selain itu terdapat panjang data yang dikirimkan yaitu Data Length dan data Sent yang merupakan data yang dikirimkan dalam bentuk json.



Gambar 6.7 Antarmuka *client1* mengirimkan data



Gambar 6.8 Antarmuka *client2* mengirimkan data

6.3.4 Analisis Pengujian

Berdasarkan dari hasil pengujian pada Tabel 6.3, pengujian fungsional *client* mengirimkan data sensor kepada *host* dapat berjalan dengan baik. Dari 10 kali pengujian diantaranya 5 kali pengujian ditiap *client, client* dapat mengirimkan data sensor kepada *host* dengan baik. Pengujian dilakukan dengan data sensor *accelerometer* yang bermacam-macam dan tidak ada kendala dari *client* untuk mengirimkan data kepada *host.* Data yang diterima oleh *host* juga sama dengan data yang dikirimkan oleh *client.* Dari hasil pengujian ini, maka didapatkan pengujian fungsional *client* mengirimkan data sensor kepada *host* dengan tingkat keberhasilan 100%.

6.4 Pengujian Pengendalian Fitur *Client*

6.4.1 Tujuan Pengujian

Tujuan pengujian kali ini untuk mengetahui keberhasilan *host* untuk memantau dan mengendalikan fitur *client.* Tingkat keberhasilan akan dilihat dari berhasilnya *host* mengendalikan fitur LED pada *client.* Pengujian ini dilakukan 5 kali pada tiap perangkat *client.*

6.4.2 Prosedur Pengujian

Prosedur pengujian pada penelitian kali ini adalah sebagai berikut:

1. *Deploy* program *host* pada Raspberry Pi 3 hingga pada kondisi *state Listen*.
2. *Deploy* program *client* pada NI MyRIO hingga pada kondisi *state Broadcast* menuju *state Wait Command*. Dilakukan pada semua *client*.
3. Pada antarmuka *host,* pilih salah satu *client* untuk dipantau. Kemudian klik tombol Go Hardware. Dilakukan pada kedua perangkat.
4. Lakukan pengendalian LED dengan menekan tombol LED pada antarmuka *host*.
5. Jika telah selesai, klik tombol Finish Hardware.

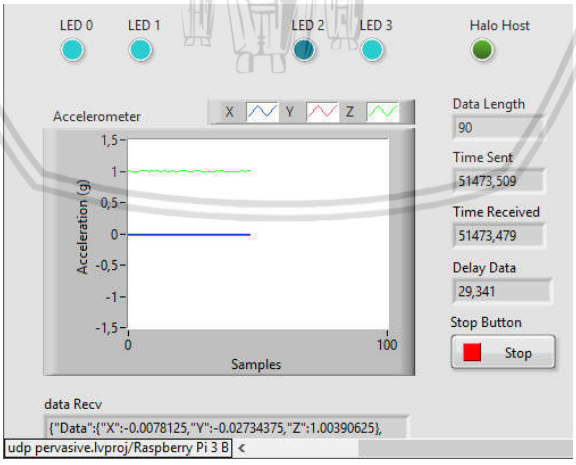
6.4.3 Hasil Pengujian

Pengujian ini memperhatikan antarmuka dan perangkat *client*. *Host* akan mengirimkan perintah untuk mengendalikan fitur yaitu LED pada *client*. Jika *host* melakukan pengendalian LED, maka pada *client* akan menjalankan perintah tersebut untuk menghidupkan LED. Hasil pengujian dapat dilihat pada Tabel 6.4.

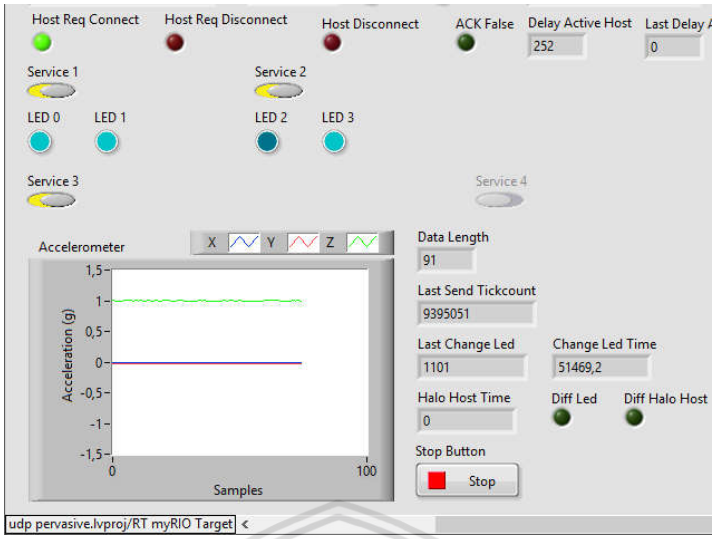
Tabel 6.4 Hasil pengujian pengendalian fitur

Pengujian ke-	Perangkat	Perintah LED	Berhasil	Keterangan
1	Client1	1000	Sukses	LED0 Hidup
2	Client1	1010	Sukses	LED0 dan LED2 Hidup
3	Client1	1101	Sukses	LED0, LED1, dan LED3 Hidup
4	Client1	1111	Sukses	LED0, LED1, LED2, dan LED3 Hidup
5	Client1	0010	Sukses	LED2 Hidup
1	Client2	0110	Sukses	LED1, dan LED2 Hidup
2	Client2	1001	Sukses	LED0, dan LED3 Hidup
3	Client2	0011	Sukses	LED2, dan LED3 Hidup
4	Client2	1100	Sukses	LED0, dan LED1 Hidup
5	Client2	0101	Sukses	LED1, dan LED3 Hidup

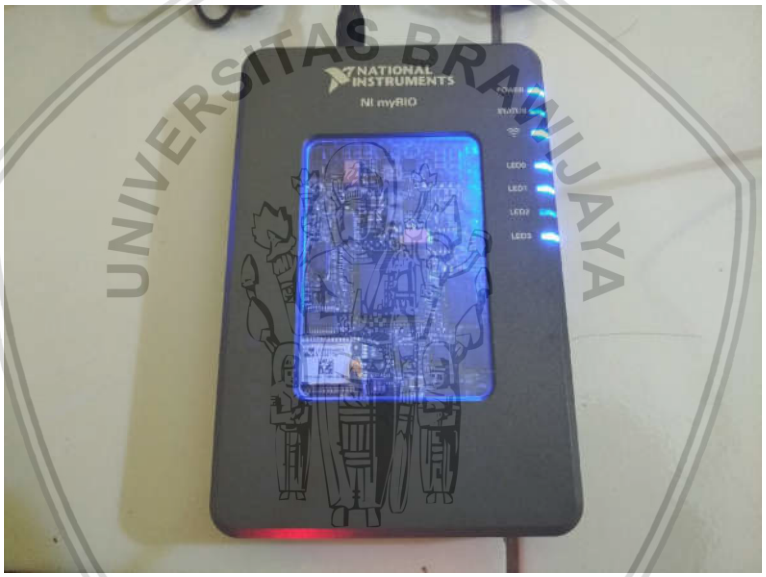
Pada Gambar 6.9 merupakan antarmuka *host* disaat mengendalikan LED pada *client1.* Pada gambar tersebut, *host* menghidupkan LED0, LED1, dan LED3. Kemudian Gambar 6.10 merupakan antarmuka *client* saat dikendalikan oleh *host.* Pada gambar tersebut LED yang dikendalikan oleh host yaitu LED0, LED1, dan LED3. Kemudian pada Gambar 6.11 merupakan keadaan perangkat *client1* saat dikendalikan oleh *host.* LED yang hidup pada perangkat tersebut yaitu LED0, LED1, dan LED3.



Gambar 6.9 Antarmuka *host* mengendalikan LED *client1*

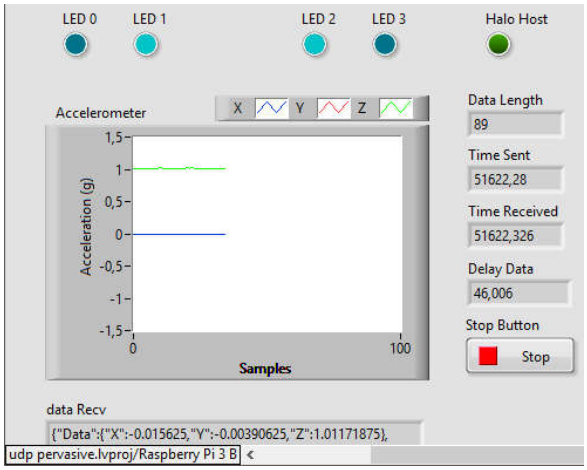


Gambar 6.10 Antarmuka *client1* saat dikendalikan

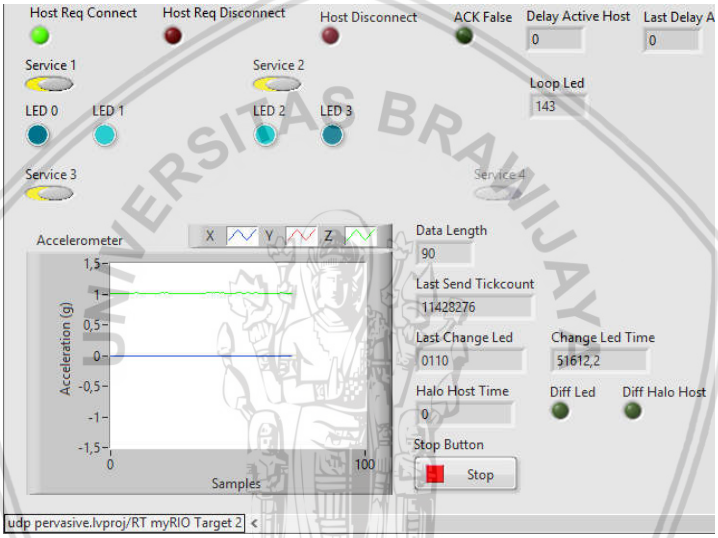


Gambar 6.11 Perangkat *client1* saat dikendalikan

Selanjutnya pada Gambar 6.12 merupakan antarmuka *host* disaat mengendalikan LED pada *client2.* Pada gambar tersebut, *host* menghidupkan LED1, dan LED2. Kemudian Gambar 6.13 merupakan antarmuka *client2* saat dikendalikan oleh *host.* Pada gambar tersebut LED yang dikendalikan oleh host yaitu LED1, dan LED2. Kemudian pada Gambar 6.14 merupakan keadaan perangkat *client2* saat dikendalikan oleh *host.* LED yang hidup pada perangkat tersebut yaitu LED1, dan LED2.



Gambar 6.12 Antarmuka *host* mengendalikan LED *client2*



Gambar 6.13 Antarmuka *client2* saat dikendalikan

Gambar 6.14 Perangkat *client2* saat dikendalikan

6.4.4 Analisis Pengujian

Berdasarkan dari hasil pengujian pada Tabel 6.4, pengujian fungsional pengendalian fitur *client* dapat berjalan dengan baik. Dari 10 kali pengujian diantaranya 5 kali pengujian ditiap *client, client* dapat menghidupkan LED dari perintah yang dikirimkan oleh *host*. Pengujian dilakukan dengan pengendalian LED untuk menghidupkan dan mematikan dan tidak ada kendala dari *client* untuk menjalankan perintah *host.* Perintah yang diterima oleh *client* juga sama dengan perintah yang dikirimkan oleh *host.* Dari hasil pengujian ini, maka didapatkan pengujian fungsional pengendalian fitur *client* dengan tingkat keberhasilan 100%.

6.5 Pengujian Notifikasi pada *host* jika *client* terputus koneksi

6.5.1 Tujuan Pengujian

Tujuan pengujian kali ini untuk mengetahui keberhasilan *host* menampilkan pesan notifikasi sistem jika *client* terputus dengan *host* dan menghapus informasi *client* pada *list client.* Tingkat keberhasilan akan dilihat dari berhasilnya *host* menampilkan pesan notifikasi sistem pada antarmuka LabVIEW *host.* Pengujian notifikasi sistem terdapat dua macam penyebabnya diantaranya disaat *host* ingin terkoneksi pada *client* yang tidak aktif dan disaat *host* sedang melakukan *request* data pada *client*. Pengujian ini dilakukan 5 kali pada tiap perangkat *client.*

6.5.2 Prosedur Pengujian

Prosedur pengujian pada penelitian kali ini adalah sebagai berikut:

1. *Deploy* program *host* pada Raspberry Pi 3 hingga pada kondisi *state Listen*.
2. *Deploy* program *client* pada NI MyRIO hingga pada kondisi *state Broadcast* menuju *state Wait Command*. Dilakukan pada semua *client*.
3. Jika pengujian disaat *host* ingin terkoneksi dengan *client,* maka hentikan program pada *client,* kemudian lakukan koneksi dengan menekan tombol Go Hardware.
4. Jika pengujian disaat *host* keadaan *request, maka* lakukan koneksi dengan *client* dengan menekan tombol Go Hardware, kemudian hentikan program pada *client*
5. Perhatikan pesan sistem yang ditampilkan oleh *host* dan list *client.*

6.5.3 Hasil Pengujian

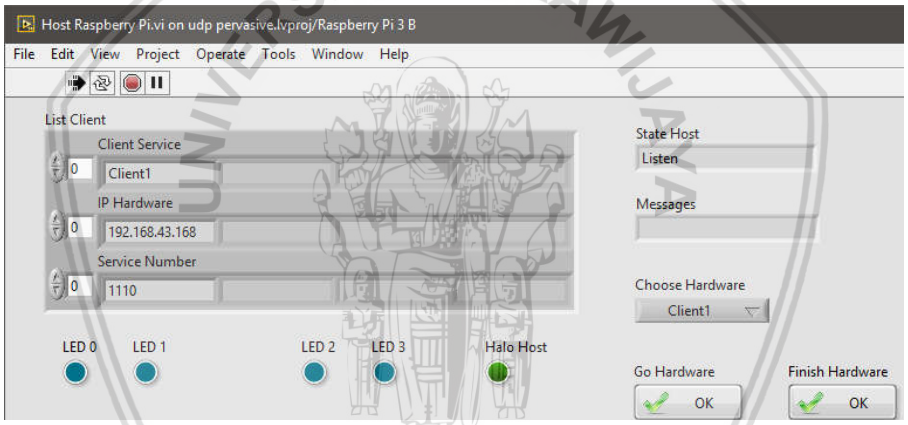
Pengujian dilakukan disaat *host* ingin melakukan koneksi dengan *client* ataupun disaat *host* pada keadaan *request. Client* dibuat seperti mengalami kegagalan dengan cara menghentikan program yang berjalan pada *client*. Kemudian memperhatikan pesan sistem yang disampaikan pada antarmuka *host*. Hasil pengujian dapat dilihat pada Tabel 6.5.

Tabel 6.5 Hasil pengujian notifikasi *host* saat *client* terputus

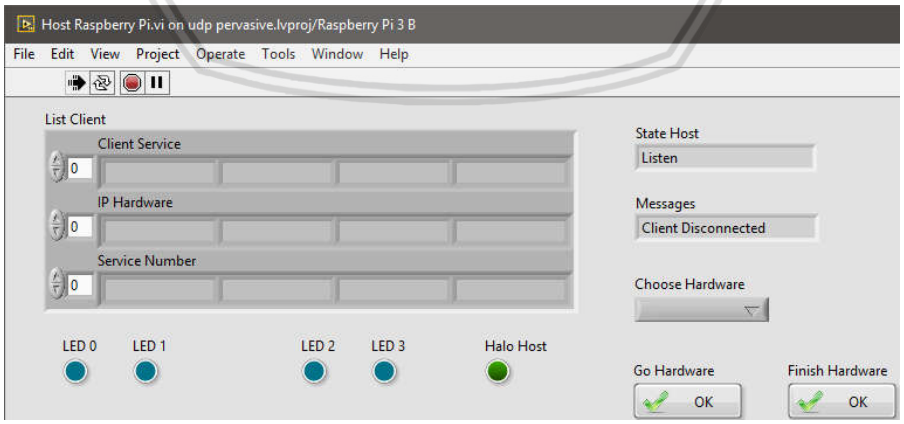
Pengujian ke-	Perangkat	Keadaan	Pesan yang disampaikan	Berhasil
1	Client1	Ingin koneksi dengan *client*	Client Disconnected	Sukses

Pengujian ke-	Perangkat	Keadaan	Pesan yang disampaikan	Berhasil
2	Client1	Terputus saat *request*	Client Disconnected	Sukses
3	Client1	Ingin koneksi dengan *client*	Client Disconnected	Sukses
4	Client1	Terputus saat *request*	Client Disconnected	Sukses
5	Client1	Ingin koneksi dengan *client*	Client Disconnected	Sukses
1	Client2	Terputus saat *request*	Client Disconnected	Sukses
2	Client2	Ingin koneksi dengan *client*	Client Disconnected	Sukses
3	Client2	Terputus saat *request*	Client Disconnected	Sukses
4	Client2	Ingin koneksi dengan *client*	Client Disconnected	Sukses
5	Client2	Terputus saat *request*	Client Disconnected	Sukses

Pada Gambar 6.15 Antarmuka *host* sebelum koneksi pada *client1*. Kemudian program pada *client1* dihentikan agar terlihat seperti mengalami kegagalan. Pada Gambar 6.16 Antarmuka *host* setelah koneksi pada *client1* saat tidak aktif. *Host* akan kembali pada *state Listen*, kemudian menampilkan pesan sistem pada *Messages* yaitu *Client Disconnected*, pada saat itu juga *host* menghapus informasi *client1* karena sudah tidak aktif.



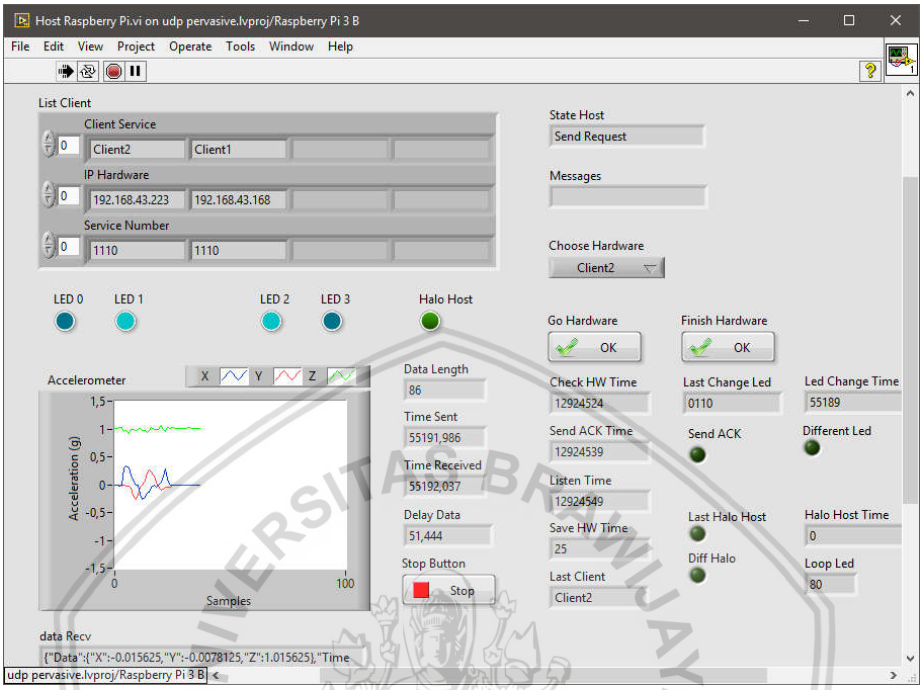
Gambar 6.15 Antarmuka *host* sebelum koneksi pada *client1*



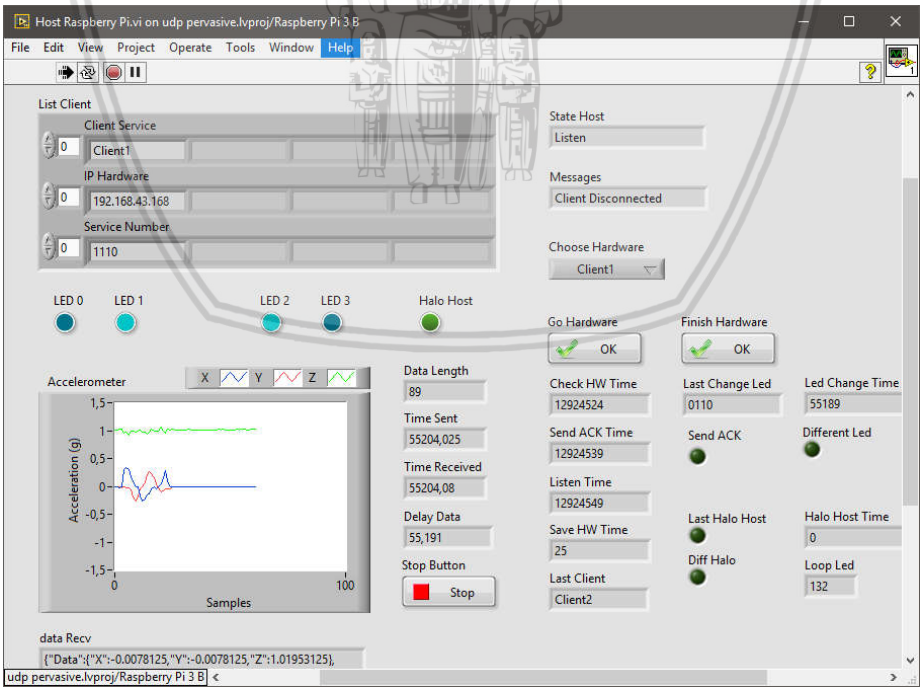
Gambar 6.16 Antarmuka *host* setelah koneksi pada *client1* saat tidak aktif

Kemudian pada Gambar 6.17 Antarmuka *host* saat *request* dengan *client2*. Pada keadaan ini, *host* berada pada *state Send Request. Client2* dibuat seolah-olah mengalami kegagalan dengan menghentikan program. Kemudian pada Gambar

6.18 Antarmuka *host* setelah *client2* terputus dari *host*. *Host* akan kembali pada *state Listen* dengan menampilkan pesan sistem pada *Messages* yaitu *Client Disconnected.* Selain itu *host* akan menghapus informasi *client2* dari *List Client* karena dianggap sudah tidak aktif.



Gambar 6.17 Antarmuka *host* saat *request* dengan *client2*



Gambar 6.18 Antarmuka *host* setelah *client2* terputus dari *host*

6.5.4 Analisis Pengujian

Berdasarkan dari hasil pengujian pada Tabel 6.5, pengujian fungsional notifikasi pada *host* saat *client* terputus dapat berjalan dengan baik. Dari 10 kali pengujian diantaranya 5 kali pengujian ditiap *client, host* dapat memberi informasi berupa pesan sistem jika *client* telah terputus dari koneksi. Pengujian dilakukan dengan dua keadaan yaitu disaat *host* ingin melakukan koneksi pada *client* tetapi *client* tidak aktif, dan disaat *host* dalam keadaan *Send Request* kemudian *client* mengalami dan tidak ada kendala dari *host* untuk menampilkan pesan sistem dan menghapus informasi *client.* Dari hasil pengujian ini, maka didapatkan pengujian fungsional notifikasi pada *host* saat *client* terputus dengan tingkat keberhasilan 100%.

6.6 Pengujian *Client* untuk mencari *host* jika terputus koneksi

6.6.1 Tujuan Pengujian

Tujuan pengujian kali ini untuk mengetahui keberhasilan *client* untuk kembali pada keadaan pencarian *host. Client* akan kembali pada *state Broadcast* untuk mencari *host* baru karena *host* sebelumnya mengalami kegagalan. Tingkat keberhasilan akan dilihat dari berhasilnya *client* kembali pada *state Broadcast* setelah terputus dari *host* selama 10 detik. Pengujian ini dilakukan 5 kali pada tiap perangkat *client.*

6.6.2 Prosedur Pengujian

Prosedur pengujian pada penelitian kali ini adalah sebagai berikut:

1. *Deploy* program *host* pada Raspberry Pi 3 hingga pada kondisi *state Listen*.
2. *Deploy* program *client* pada NI MyRIO hingga pada kondisi *state Broadcast* menuju *state Wait Command*. Dilakukan pada semua *client*.
3. Hentikan program pada *host.*
4. Perhatikan antarmuka *client* dan berpindah pada *state Broadcast.*
5. Jalankan kembali program *host* hingga *client* terhubung lagi
6. Ulangi pengujian pada step-3

6.6.3 Hasil Pengujian

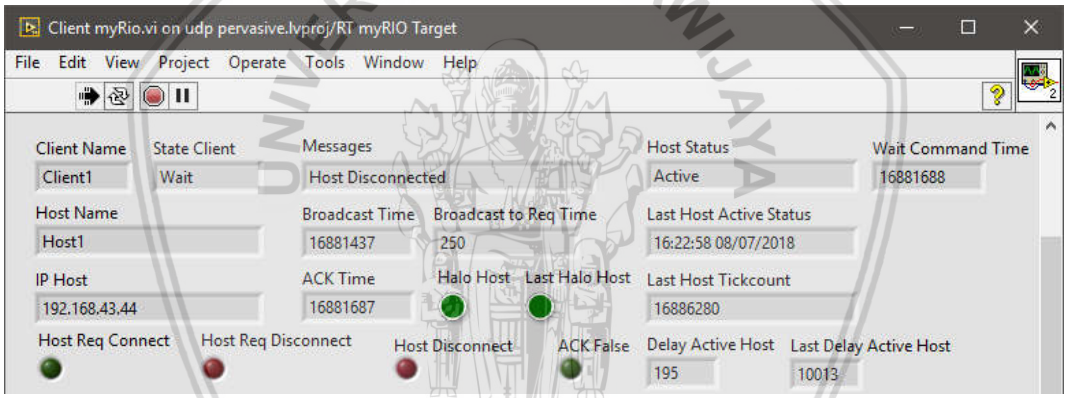
Pengujian dilakukan disaat *client* berada pada *state Wait Command. Host* dibuat seperti mengalami kegagalan dengan cara menghentikan program yang berjalan pada *Host*. Kemudian memperhatikan perilaku *client* pada antarmuka *client*. Hasil pengujian dapat dilihat pada Tabel 6.6.

Tabel 6.6 Hasil pengujian *host* terputus dari *client*

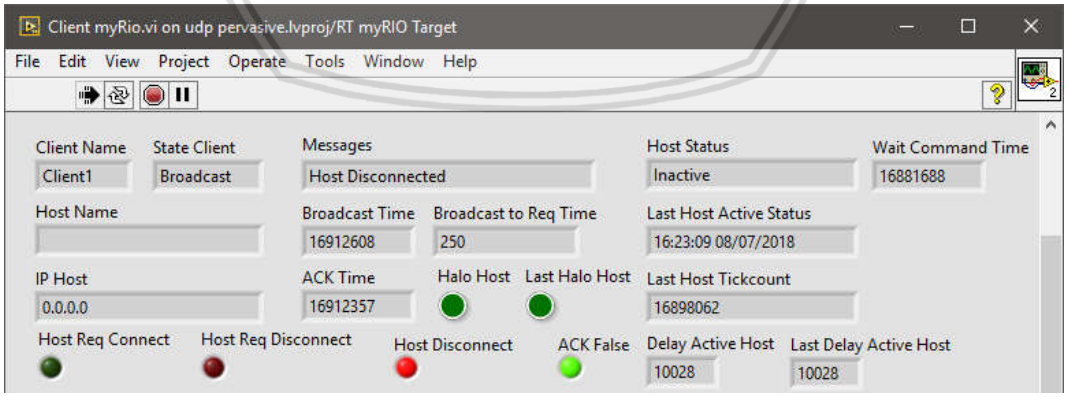
Pengujian ke-	Perangkat	Status LED Host Disconnect	Berhasil kembali pada state *Broadcast*
1	Client1	True	Sukses
2	Client1	True	Sukses
3	Client1	True	Sukses

Pengujian ke-	Perangkat	Status LED Host Disconnect	Berhasil kembali pada state *Broadcast*
4	Client1	True	Sukses
5	Client1	True	Sukses
1	Client2	True	Sukses
2	Client2	True	Sukses
3	Client2	True	Sukses
4	Client2	True	Sukses
5	Client2	True	Sukses

Pada Gambar 6.19 Antarmuka *client1* sebelum *host* terputus, *client* berada pada *state Wait Command*. Kemudian program pada *host* dihentikan agar terlihat seperti mengalami kegagalan. Pada Gambar 6.20 Antarmuka *client1* setelah *host* terputus*. Client* akan kembali pada *state Broadcast*. Pada LED *Host Disconnected* merupakan representasi bila *host* telah terputus dari *client.* Selain itu, *client* akan menampilkan pesan *Host Disconnected,* dan status dari *Host Status* menjadi *Inactive.* Waktu yang dibutuhkan *client* untuk kembali pada *state Broadcast* yaitu 10 detik, dan nilai yang didapatkan pada *Delay Active Host* yaitu 10028ms.

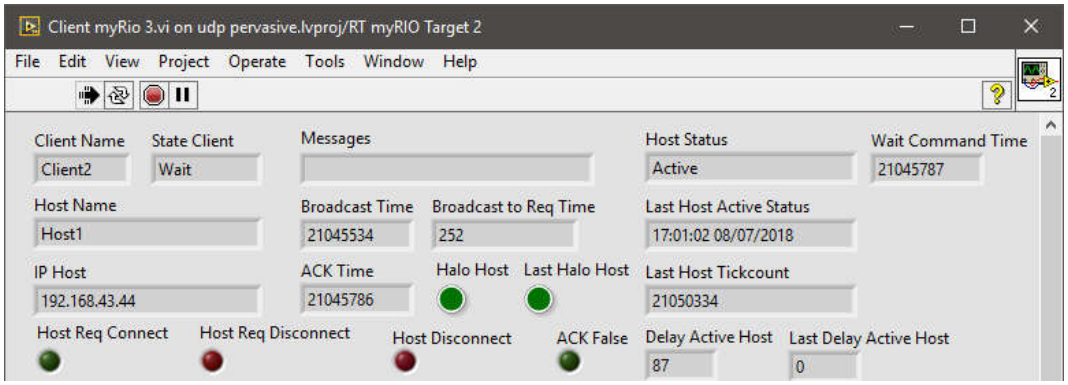


Gambar 6.19 Antarmuka *client1* sebelum *host* terputus

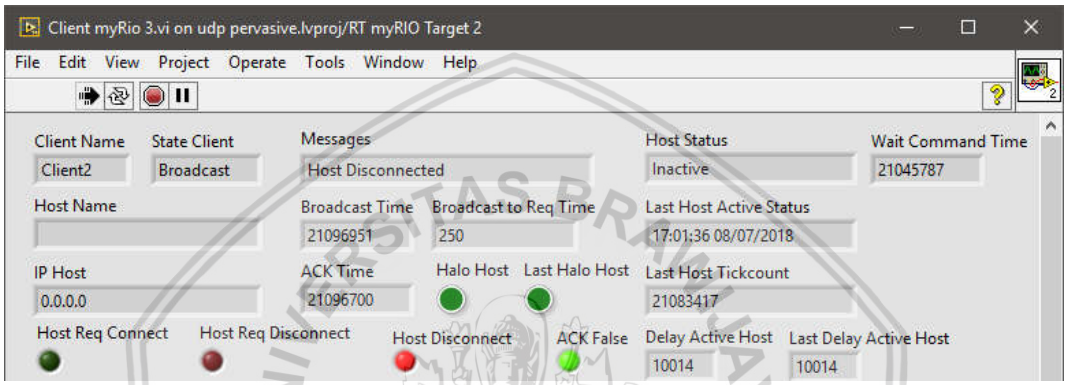


Gambar 6.20 Antarmuka *client1* setelah *host* terputus

Kemudian pengujian sama dilakukan pada *client2.* Gambar 6.21 Antarmuka *client2* sebelum *host* terputus. Setelah *host mengalami kegagalan,* maka Gambar 6.22 Antarmuka *client2* setelah *host* terputus*.*



Gambar 6.21 Antarmuka *client2* sebelum *host* terputus



Gambar 6.22 Antarmuka *client2* setelah *host* terputus

6.6.4 Analisis Pengujian

Berdasarkan dari hasil pengujian pada Tabel 6.6, pengujian fungsional *client* untuk kembali pada pencarian *host* setelah host mengalami kegagalan dapat berjalan dengan baik. Dari 10 kali pengujian diantaranya 5 kali pengujian ditiap *client, client* dapat kembali pada *state Broadcast* untuk mencari *host* yang baru. Dari hasil pengujian ini, maka didapatkan pengujian fungsional *client* untuk kembali pada pencarian *host* setelah host mengalami kegagalan dengan tingkat keberhasilan 100%.

6.7 Pengujian Discovery Perangkat

6.7.1 Tujuan Pengujian

Tujuan pengujian kali ini untuk mengetahui keberhasilan Raspberry Pi 3 sebagai *host* untuk mendeteksi NI myRIO sebagai *client* baru. Tingkat keberhasilan akan dilihat dari berhasilnya *host* untuk menyimpan informasi yang diterima oleh *client*. Tujuan lainnya untuk mengetahui kecepatan perangkat untuk saling mengenali.

6.7.2 Prosedur Pengujian

Prosedur pengujian pada penelitian kali ini adalah sebagai berikut:

1. *Deploy* program *host* pada Raspberry Pi 3 hingga pada kondisi *state Listen*.

2. *Deploy* program *client* pada NI MyRIO hingga pada kondisi *state Broadcast* menuju *state Wait Command*. Dilakukan pada semua *client*.
3. Perhatikan pada list *client* yang tersimpan pada antarmuka *host* pada LabVIEW.
4. Menghentikan program pada *client* dan melakukan pengujian ulang dimulai pada *step 2.* Dilakukan secara bergantian pada semua *client*.

6.7.3 Hasil Pengujian

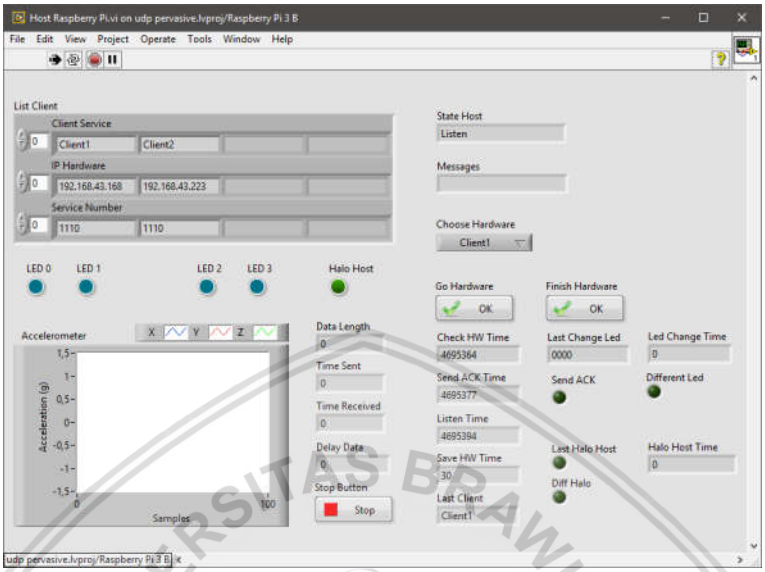
Pengujian dilakukan pada tiga perangkat, satu buah *host* dan dua buah *client*. Pada host akan menyimpan informasi dari *client* dan akan mencatat waktu transisi *state* dari menerima pesan *broadcast* hingga mengirimkan pesan *broadcast*. Sedangkan *client* akan menyimpan informasi dari *host* setelah menerima pesan ACK dan mencatat waktu transisi *state* dari mengirimkan pesan *broadcast* hingga waktu tunggu *client* untuk Request. Hasil pengujian dapat dilihat pada Tabel 6.7.

Tabel 6.7 Data Pengujian *Discovery*

Pengujian ke-	Perangkat	Transisi Host (ms)	Transisi Client (ms) (Broadcast – ACK)	Transisi Client (ms) (ACK – Wait)	Total Waktu Discovery (ms)
1	Client1	107	252	1	360
2	Client1	44	251	1	296
3	Client1	696	250	90	1036
4	Client1	249	252	1	502
5	Client1	73	252	1	326
6	Client1	25	250	136	411
7	Client1	282	250	1	533
8	Client1	34	251	1	286
9	Client1	28	251	1	280
10	Client1	103	250	30	383
11	Client1	26	252	1	279
12	Client1	34	250	1	285
13	Client1	26	250	1	277
14	Client1	27	250	2	279
15	Client1	36	251	1	288
16	Client1	26	251	1	278
17	Client1	24	250	1	275
18	Client1	21	252	1	274
19	Client1	34	252	1	287
20	Client1	28	250	1	279
21	Client1	27	252	0	279
22	Client1	28	250	1	279
23	Client1	19	250	1	270
24	Client1	40	252	1	293

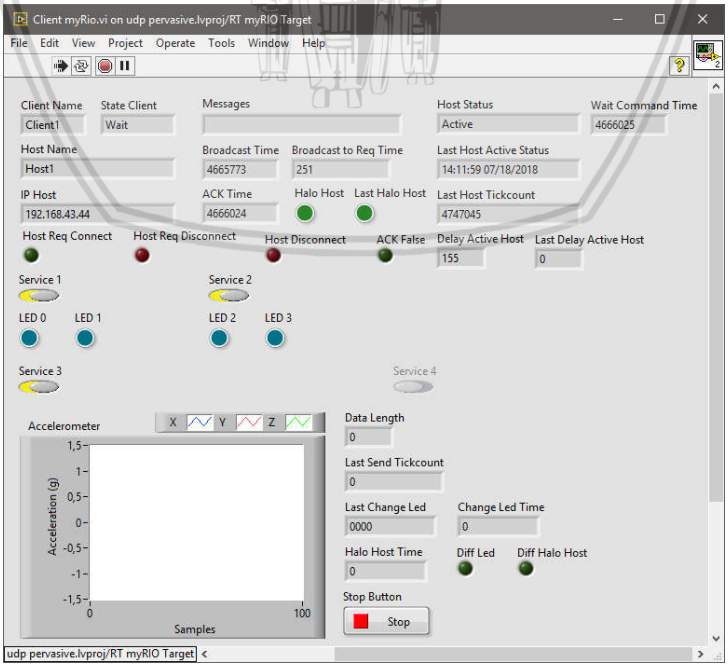
Pengujian ke-	Perangkat	Transisi Host (ms)	Transisi Client (ms) (Broadcast – ACK)	Transisi Client (ms) (ACK – Wait)	Total Waktu Discovery (ms)
25	Client1	26	252	1	279
26	Client1	24	251	1	276
27	Client1	23	252	1	276
28	Client1	29	250	1	280
29	Client1	27	252	1	280
30	Client1	29	253	1	283
1	Client2	35	251	1	287
2	Client2	24	252	1	277
3	Client2	27	251	1	279
4	Client2	165	250	26	441
5	Client2	28	252	1	281
6	Client2	26	251	1	278
7	Client2	29	251	1	281
8	Client2	25	251	1	277
9	Client2	25	252	1	278
10	Client2	34	251	1	286
11	Client2	42	251	1	294
12	Client2	28	250	1	279
13	Client2	82	251	1	334
14	Client2	26	250	1	277
15	Client2	30	251	1	282
16	Client2	34	251	1	286
17	Client2	27	250	1	278
18	Client2	98	252	18	368
19	Client2	109	251	1	361
20	Client2	22	252	1	275
21	Client2	30	252	1	283
22	Client2	21	252	1	274
23	Client2	31	252	1	284
24	Client2	27	252	1	280
25	Client2	24	252	1	277
26	Client2	31	251	1	283
27	Client2	27	250	2	279
28	Client2	24	251	1	276
29	Client2	32	250	1	283
30	Client2	27	250	1	278
	Rata-Rata	56,417	251,067	5,933	313,417
	Rata-rata Discovery Client1				333,633
	Rata-rata Discovery Client2				293,2

Pengujian *host* dapat dilihat pada Gambar 6.23. Gambar tersebut merupakan jendela antarmuka dari *host* pada aplikasi LabVIEW. Dapat dilihat bahwa proses *discovery* dari *host* dapat disimpan baik informasinya oleh *host* berupa nama *client*, IP *client*, dan fitur pelayanan yang dibawakan oleh *client.*

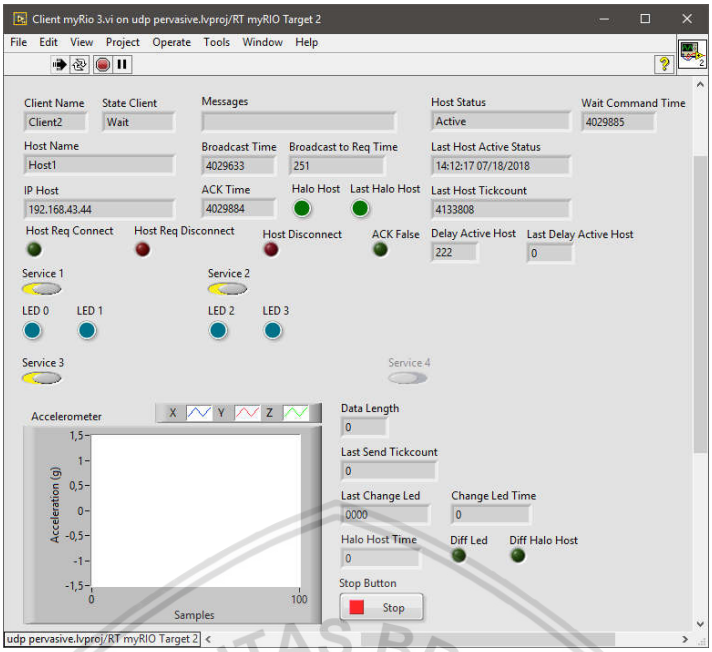


Gambar 6.23 Jendela Host LabVIEW

Pengujian *client1* dan *client2* dapat dilihat pada Gambar 6.24 dan Gambar 6.25 Gambar tersebut merupakan jendela antarmuka dari *client1* dan *client2* pada aplikasi LabVIEW. Dapat dilihat bahwa proses *discovery* dari *client* dapat berjalan dan informasi dari *host* dapat disimpan. Informasi tersebut diantaranya nama *host*, dan IP *host.*



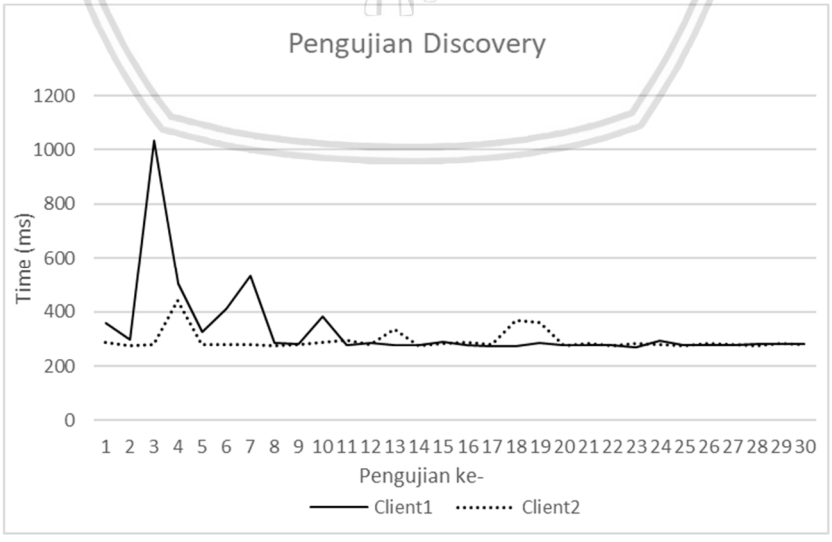
Gambar 6.24 Jendela *Client1* LabVIEW



Gambar 6.25 Jendela *Client2* LabVIEW

6.7.4 Analisis Pengujian

Dari 60 kali percobaan, diantaranya percobaan 30 kali pada masing-masing *client*, pengujian dapat berjalan dengan baik. Dari hasil data pada Tabel 6.7, dapat disimpulkan bahwa rata-rata dari pengujian *discovery* pada bagian *host* yaitu 56,417ms. Dimana pengujian ini disaat *host* menerima pesan *broadcast*, penyimpanan informasi, dan membalas pesan dengan pesan ACK. Sedangkan pengujian pada *client* yaitu 251,067ms untuk transisi pengiriman *broadcast* hingga menerima pesan ACK, dan untuk transisi dari menerima pesan ACK menuju *Wait Command* yaitu 5,933ms.



Gambar 6.26 Grafik pengujian *discovery*

Pada Gambar 6.26 merupakan grafik rata-rata dari pengujian *discovery*. Pada grafik tersebut terdapat data yang melonjak pada *client1* dengan nilai 1036ms, sedangkan nilai minimum yang dicapai yaitu 270ms. Sehingga rata-rata pengujian *discovery* pada *client1* yaitu 333,633ms. Pada *client2* grafik dari waktu yang dibutuhkan untuk *discovery* terlihat stabil. Dengan nilai maksimum 441ms dan nilai minimum yaitu 274ms. Nilai rata-rata yang didapatkan pada *client2* yaitu 293,2ms. Dan dapat disimpulkan, kecepatan dari proses *discovery* seluruhnya dari pengiriman pesan *broadcast* hingga *client* siap menerima *request* yaitu 313,417ms.

6.8 Pengujian Terhubung Kembali Pada Host Oleh Client

6.8.1 Tujuan Pengujian

Tujuan pengujian kali ini untuk mengetahui keberhasilan NI MyRIO sebagai *client* untuk terhubung kembali dengan *host* jika *host* mengalami kegagalan. Kegagalan yang dimaksudkan pada *host* yaitu *host* mengalami *restart,* terputus dari jaringan, daya, ataupun kegagalan lainnya yang dapat terjadi. Tingkat keberhasilan akan dilihat dari berhasilnya *client* untuk terhubung kembali pada *host*. Tujuan lainnya untuk mengetahui kecepatan *client* untuk terhubung kembali dengan *host*.

6.8.2 Prosedur Pengujian

Prosedur pengujian pada penelitian kali ini adalah sebagai berikut:

1. *Deploy* program *host* pada Raspberry Pi 3 hingga pada kondisi *state Listen*.
2. *Deploy* program *client* pada NI MyRIO hingga pada kondisi *state Broadcast* menuju *state Wait Command*. Dilakukan pada semua *client*.
3. Stop program dari *host,* dan deploy kembali program *host*.
4. Perhatikan pada program *client* hingga *client* terhubung kembali pada *host.*

6.8.3 Hasil Pengujian

Pengujian dilakukan pada tiga perangkat, satu buah *host* dan dua buah *client*. *Host* akan dibuat seakan-akan mengalami kegagalan, dengan cara menghentikan program dan men-*deploy* kembali program *host*. *Client* akan menunggu untuk waktu yang ditentukan hingga *client* melakukan *discovery* kembali hingga terhubung kembali pada *host.* Hasil pengujian dapat dilihat pada Tabel 6.8.

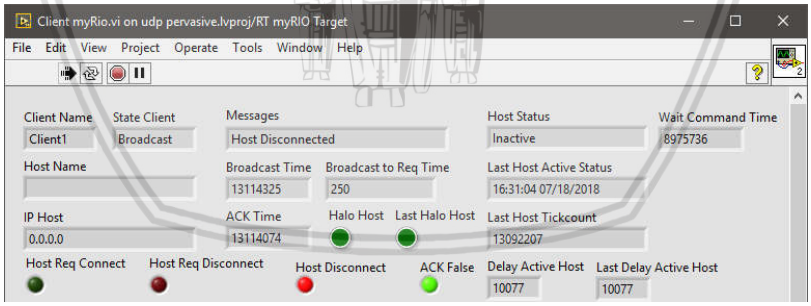
Tabel 6.8 Pengujian Terhubung Kembali

Pengujian ke-	Perangkat	Discovery Host (ms)	Client Timeout (ms)	Discovery Client (ms)	Total Terhubung kembali (ms)
1	Client1	26	10088	252	10366
2	Client1	42	10069	253	10364
3	Client1	32	10050	251	10333
4	Client1	27	10052	254	10333
5	Client1	27	10072	251	10350

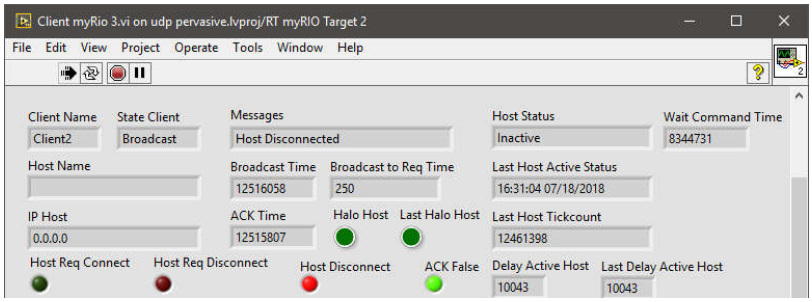
Pengujian ke-	Perangkat	Discovery Host (ms)	Client Timeout (ms)	Discovery Client (ms)	Total Terhubung kembali (ms)
6	Client1	27	10097	268	10392
7	Client1	151	10041	253	10445
8	Client1	26	10071	252	10349
9	Client1	36	10084	261	10381
10	Client1	28	10060	253	10341
11	Client1	34	10055	253	10342
12	Client1	87	10006	253	10346
13	Client1	22	10013	253	10288
14	Client1	30	10075	349	10454
15	Client1	26	10094	252	10372
16	Client1	35	10063	325	10423
17	Client1	29	10095	251	10375
18	Client1	33	10087	365	10485
19	Client1	28	10093	411	10532
20	Client1	33	10048	252	10333
21	Client1	31	10061	401	10493
22	Client1	26	10046	251	10323
23	Client1	70	10073	252	10395
24	Client1	36	10020	252	10308
25	Client1	31	10045	253	10329
26	Client1	30	10000	251	10281
27	Client1	37	10086	312	10435
28	Client1	34	10067	467	10568
29	Client1	29	10085	351	10465
30	Client1	36	10047	452	10535
1	Client2	29	10040	251	10320
2	Client2	21	10016	252	10289
3	Client2	34	10094	319	10447
4	Client2	27	10061	256	10344
5	Client2	51	10088	283	10422
6	Client2	32	10035	252	10319
7	Client2	33	10067	412	10512
8	Client2	25	10054	251	10330
9	Client2	25	10052	251	10328
10	Client2	29	10010	253	10292
11	Client2	28	10098	364	10490
12	Client2	36	10059	422	10517
13	Client2	27	10042	252	10321
14	Client2	31	10012	252	10295

Pengujian ke-	Perangkat	Discovery Host (ms)	Client Timeout (ms)	Discovery Client (ms)	Total Terhubung kembali (ms)
15	Client2	25	10035	252	10312
16	Client2	31	10009	253	10293
17	Client2	24	10077	251	10352
18	Client2	28	10067	252	10347
19	Client2	78	10062	252	10392
20	Client2	37	10065	313	10415
21	Client2	32	10039	252	10323
22	Client2	31	10070	260	10361
23	Client2	33	10099	447	10579
24	Client2	38	10043	441	10522
25	Client2	32	10055	359	10446
26	Client2	37	10015	295	10347
27	Client2	37	10045	251	10333
28	Client2	33	10057	253	10343
29	Client2	29	10066	252	10347
30	Client2	89	10038	253	10380
Rata-rata		36,35	10056,88	291	10384,23

Pada Gambar 6.27 dan Gambar 6.28 merupakan pengujian yang dilakukan pada *client1* dan *Client2*. Dapat dilihat pada gambar tersebut merupakan pengujian saat *host* terputus dengan *client*. Pada gambar juga dijelaskan tanda *host* telah terputus dengan LED warna merah pada *Host Disconnected* dan pencatatan waktu *timeout* dengan nama *Delay Active Host*.



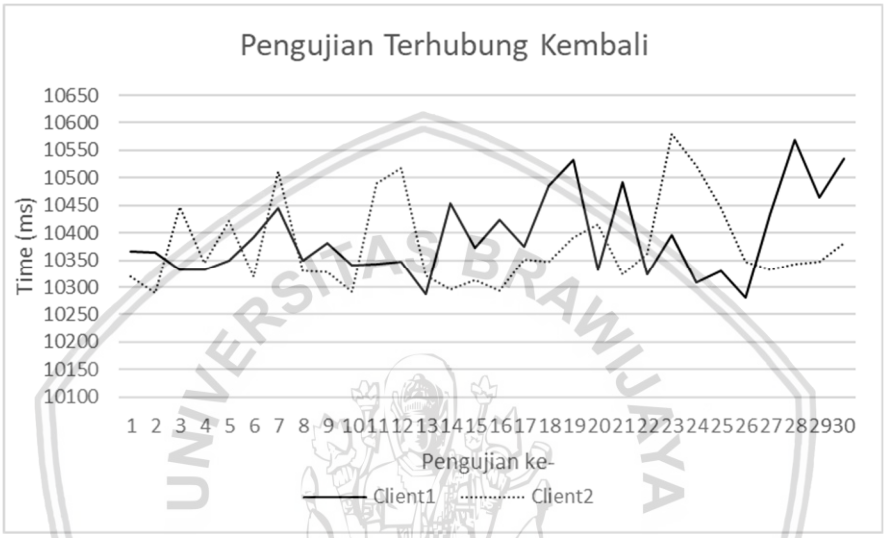
Gambar 6.27 Pengujian *Client1* terputus dengan *host*



Gambar 6.28 Pengujian *Client2* terputus dengan *host*

6.8.4 Analisis Pengujian

Pengujian kali ini dilakukan sebanyak 60 kali percobaan, dimana masing-masing *client* melakukan 30 kali percobaan. Pengujian kali ini yaitu saat *client* terputus dari *host*. *Client* mempunyai *timeout* untuk menuju kondisi *discovery* kembali, yang digunakan yaitu 10 detik. Berdasarkan Tabel 6.8, rata-rata *client* yang didapatkan untuk kembali pada kondisi *discovery* ulang yaitu 10056,88ms. Setelah kembali pada proses *discovery*, *client* dapat kembali pada keadaan *Wait Command* yaitu dengan rata-rata 291ms. Dan pada *host* proses *discovery* didapatkan rata-rata yaitu 36,35ms.



Gambar 6.29 Grafik pengujian terhubung kembali

Pada Gambar 6.29 merupakan grafik dari rata-rata pengujian terhubung kembali. Pada grafik terlihat bahwa grafik tidak stabil, waktu yang didapatkan selalu berubah-ubah di setiap percobaannya. Pada *client1* didapatkan nilai maksimum yaitu 10568ms dan nilai minimum 10281ms. Sehingga didapatkan nilai rata-rata dari *client1* yaitu 10391,2ms. Sedangkan pada *client2* didapatkan nilai maksimum yaitu 10579ms dan nilai minimum 10568ms. Sehingga didapatkan nilai rata-rata dari *client2* yaitu 10377,27ms. Jadi waktu yang dibutuhkan keseluruhan untuk terhubung kembali yaitu dengan rata-rata sebesar 10384,23ms.

6.9 Pengujian Penggunaan Fitur Layanan Client

6.9.1 Tujuan Pengujian

Tujuan pengujian kali ini untuk mengetahui keberhasilan *host* dan *client* untuk saling bertukar data. Data yang digunakan yaitu sensor *accelerometer,* LED, dan *push button* dari *client*. *Host* akan menerima data sensor dan kondisi *push button* dari *client,* dan dapat mengendalikan LED pada *client*. Tujuan lain dari pengujian kali ini yaitu untuk mengetahui waktu yang dibutuhkan agar data tersampai pada tujuan perangkat.

6.9.2 Prosedur Pengujian

Prosedur pengujian pada penelitian kali ini adalah sebagai berikut:

1. *Deploy* program *host* pada Raspberry Pi 3 hingga pada kondisi *state Listen*.
2. *Deploy* program *client* pada NI MyRIO hingga pada kondisi *state Broadcast* menuju *state Wait Command*. Dilakukan pada semua *client*.
3. Pilih salah satu *client* untuk mengendalikan fitur *client.* Dilakukan pada seluruh *client*.

6.9.3 Hasil Pengujian

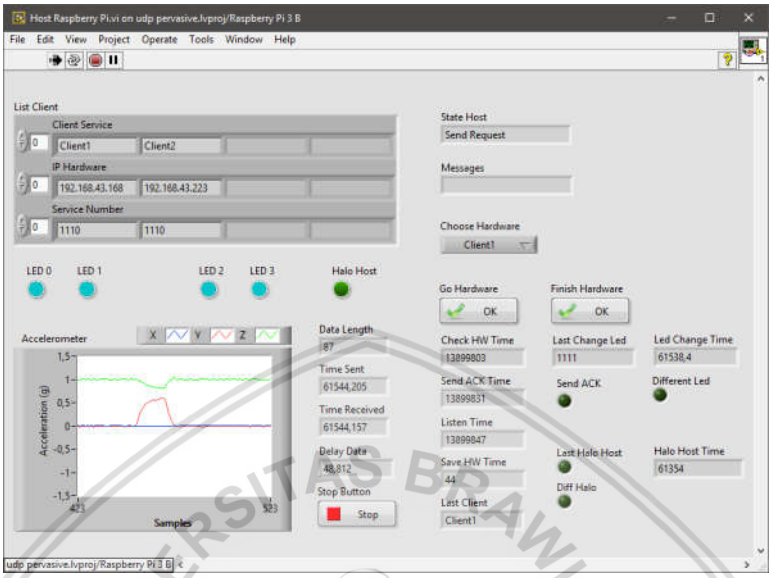
Pengujian dilakukan pada tiga perangkat, satu buah *host* dan dua buah *client*. *Host* akan mengirimkan pesan *request* untuk mengendalikan dan memantau fitur yang dimiliki oleh *client*. Terdapat tiga data yang digunakan, yaitu data sensor *accelerometer,* pengendalian LED, dan *push button client*. Pengujian data sensor dapat dilihat pada Tabel 6.9.

Tabel 6.9 Pengujian Data Sensor

Pengujian ke-	Perangkat	Panjang Data	Waktu yang dibutuhkan (ms)	State LED	Waktu yang dibutuhkan (ms)	State Push button	Waktu yang dibutuhkan (ms)
1	Client1	90	679,814	0010	3810,153	TRUE	296,732
2	Client1	89	0,968	0011	3809,271	FALSE	295,456
3	Client1	87	46,87	0001	4062,224	TRUE	297,576
4	Client1	87	46,01	0101	4564,36	FALSE	296,667
5	Client1	89	35,407	0111	4552,169	TRUE	298,543
6	Client1	87	47,79	0011	4812,104	FALSE	297,321
7	Client1	85	47,186	0001	4810,164	TRUE	287,47
8	Client1	90	33,676	1001	4814,07	FALSE	297,259
9	Client1	90	46,298	1101	4815,262	TRUE	295,37
10	Client1	88	46,113	1111	4814,107	FALSE	292,839
11	Client1	82	8,088	1011	5020,78	TRUE	215,312
12	Client1	87	47,45	0011	5563,523	FALSE	256,911
13	Client1	90	47,958	0001	5565,238	TRUE	222,236
14	Client1	87	47,864	0101	6571,928	FALSE	44,614
15	Client1	84	44,702	1101	7251,994	TRUE	297,14
16	Client1	85	48,182	1111	8074,47	FALSE	47,991
17	Client1	87	47,901	1011	8327,576	TRUE	212,817
18	Client1	90	42,695	0011	9071,816	FALSE	297,705
19	Client1	88	48,223	0001	10089,225	TRUE	548,276
20	Client1	90	47,92	0000	10827,562	FALSE	204,729
21	Client1	90	47,575	0100	12081,961	TRUE	47,501
22	Client1	76	47,306	0110	12333,948	FALSE	300,376
23	Client1	88	46,957	1110	13329,447	TRUE	298,035

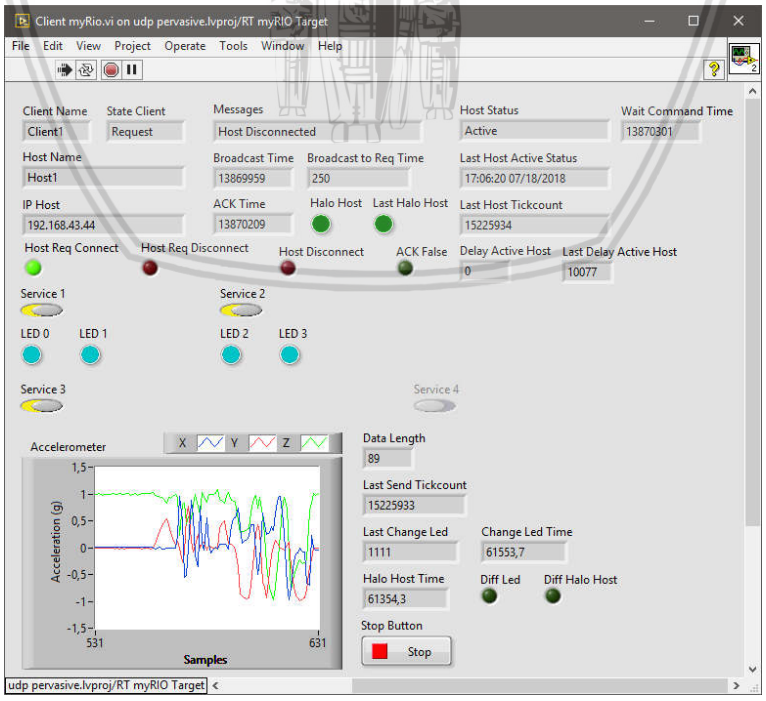
Pengujian ke-	Perangkat	Panjang Data	Waktu yang dibutuhkan (ms)	State LED	Waktu yang dibutuhkan (ms)	State Push button	Waktu yang dibutuhkan (ms)
24	Client1	88	46,28	1111	13333,315	FALSE	298,61
25	Client1	86	45,904	1011	13586,364	TRUE	300,546
26	Client1	78	47,557	1001	14083,281	FALSE	298,055
27	Client1	86	47,845	0001	14341,298	TRUE	297,344
28	Client1	78	508,626	0101	15093,435	FALSE	296,474
29	Client1	90	48,464	0111	15345,068	TRUE	297,122
30	Client1	90	51,089	1111	15346,769	FALSE	298,125
1	Client2	86	691,809	0010	487,603	TRUE	243,579
2	Client2	86	7,215	0011	480,632	FALSE	243,735
3	Client2	88	7,768	0111	738,062	TRUE	244,305
4	Client2	87	7,561	1111	744,308	FALSE	244,241
5	Client2	88	33,19	1011	745,545	TRUE	243,697
6	Client2	88	7,362	1001	996,265	FALSE	246,176
7	Client2	89	8,159	0001	1233,892	TRUE	244,201
8	Client2	89	9,781	0000	1495,135	FALSE	243,526
9	Client2	88	7,796	0010	1660,08	TRUE	244,446
10	Client2	87	27,348	0110	1744,266	FALSE	243,597
11	Client2	87	5,63	0111	1916,876	TRUE	239,78
12	Client2	86	8,03	1111	2003,125	FALSE	242,429
13	Client2	86	35,274	1011	2002,027	TRUE	242,964
14	Client2	86	7,129	1001	2000,727	FALSE	241,167
15	Client2	88	8,644	1000	1998,371	TRUE	242,528
16	Client2	88	36,191	1010	1995,791	FALSE	241,572
17	Client2	87	7,809	1110	1998,373	TRUE	243,547
18	Client2	86	7,706	1010	2749,351	FALSE	242,175
19	Client2	87	40,045	1000	2742,152	TRUE	244,031
20	Client2	86	5,532	1001	2751,674	FALSE	243,648
21	Client2	88	9,527	1101	2752,64	TRUE	243,588
22	Client2	86	8	0101	2752,028	FALSE	245,561
23	Client2	86	7,498	0001	2754,025	TRUE	245,84
24	Client2	87	55,797	0011	2747,855	FALSE	243,299
25	Client2	87	7,222	0010	2751,915	TRUE	244,878
26	Client2	86	7,749	0110	2753,083	FALSE	245,799
27	Client2	84	8,06	0111	2955,655	TRUE	243,619
28	Client2	88	8,381	0101	3003,918	FALSE	241,57
29	Client2	85	7,869	1101	3001,086	TRUE	244,452
30	Client2	87	6,963	1111	2996,231	FALSE	242,664
	Rata-rata		58,263		5350,926		255,696

Pada Gambar 6.30 merupakan pengujian pada *host*. Pada gambar tersebut merupakan keadaan saat terhubung dengan *client1*. *Host* dapat memantau sensor *accelerometer* dalam bentuk grafik yang mewakili dari X-axis, Y-axis, dan Z-axis dari *accelerometer*, keadaan *push button*, dan mengendalikan LED pada *client*.

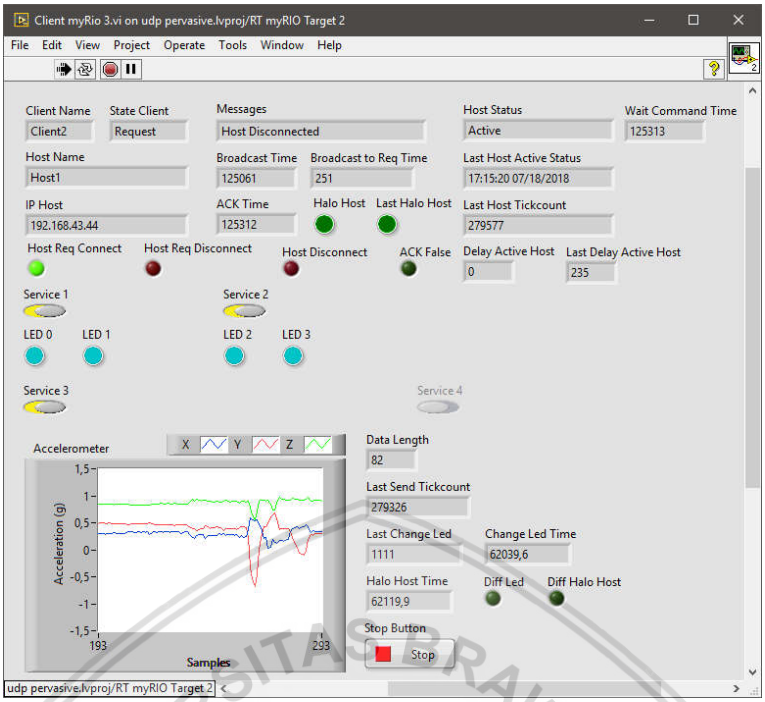


Gambar 6.30 Pengendalian fitur pada *Host*

Pada Gambar 6.31 dan Gambar 6.32 merupakan tampilan pada *client1* dan *client 2*. Pada gambar terlihat bahwa ada tanda *host* terhubung dengan *client1* dan *client2*. *Client* akan mengirimkan data sensor yang direpresentasikan pada grafik, *push button*, dan menerima data untuk mengendalikan LED pada *client*.



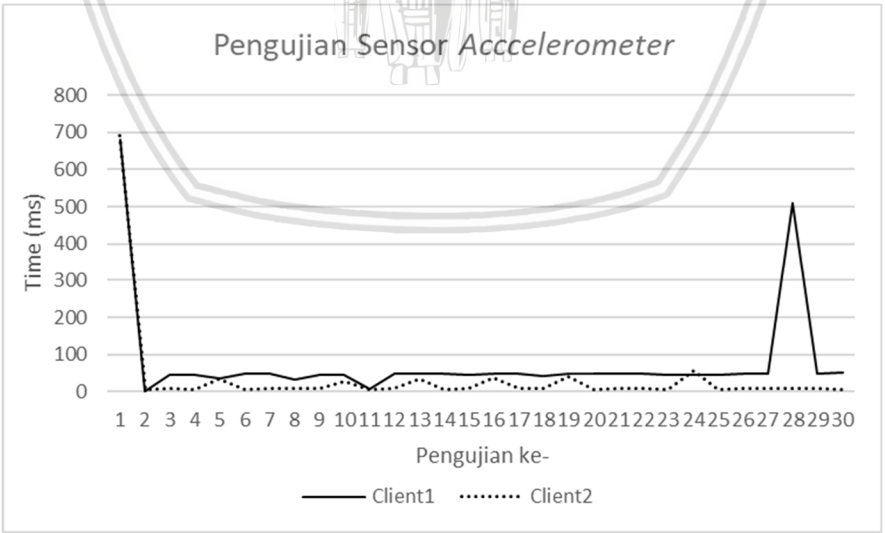
Gambar 6.31 Jendela LabVIEW pada *client1*



Gambar 6.32 Jendela LabVIEW pada *Client2*

6.9.4 Analisis Pengujian

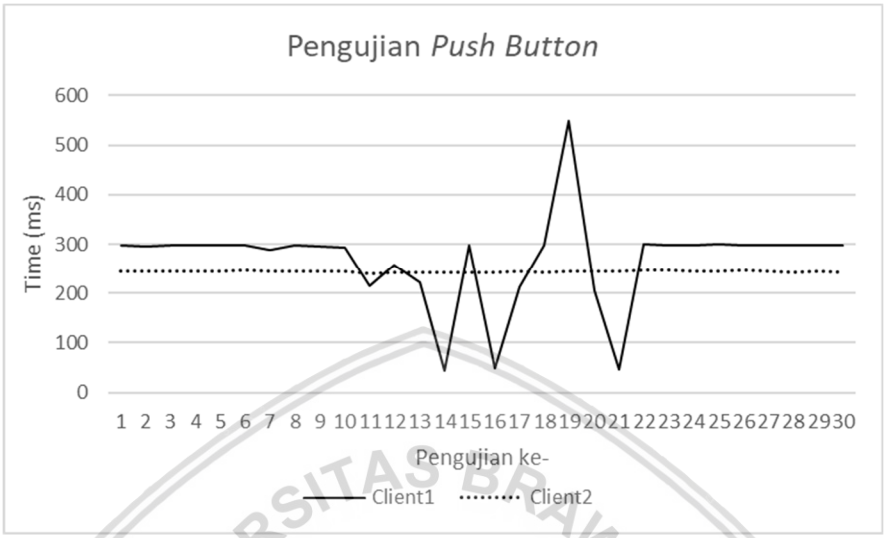
Berdasarkan pengujian yang dilakukan kali ini, dilakukan percobaan selama 60 kali pada masing-masing data yang dikirimkan, dimasing-masing *client* dilakukan 30 kali percobaan. Pengujian dapat berjalan dengan baik. Pada Gambar 6.33 merupakan grafik dari pengujian data sensor dan Gambar 6.34 merupakan pengujian *push button,* dan Gambar 6.35 merupakan pengujian LED.



Gambar 6.33 Grafik pengujian sensor *accelerometer*

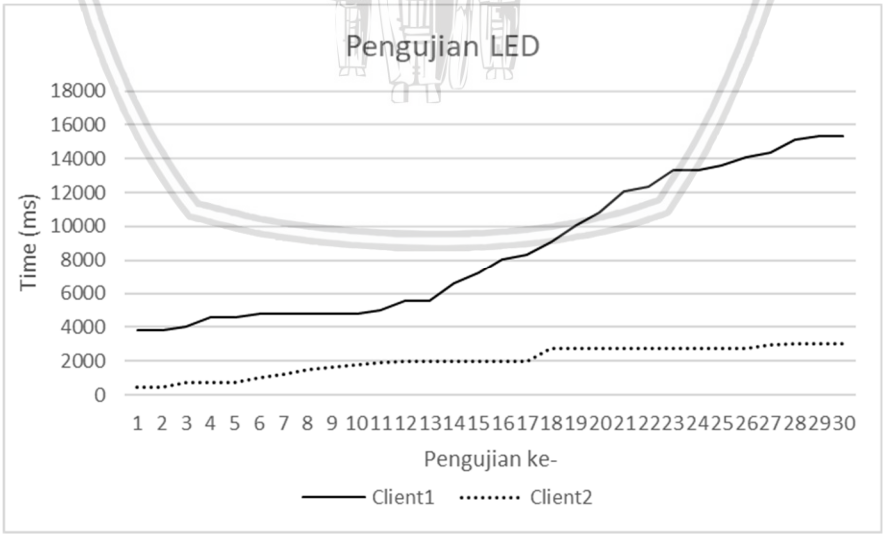
Berdasarkan Gambar 6.33, pada data sensor yang dikirimkan, grafik menggambarkan nilai yang stabil pada *client1* dan *client2*. Lonjakan nilai tersebut terjadi pada awal pengiriman data di kedua *client*. Nilai maksimum dari

pengiriman data sensor pada *client1* yaitu 679,814ms sedangkan nilai pada *client2* yaitu 691,809ms. Nilai minimum dari pengiriman data sensor pada *client1* yaitu 0,968ms dan *client2* yaitu 5,532ms. Jadi nilai rata-rata untuk pengiriman data sensor berdasarkan Tabel 6.9 yaitu 58,263ms.



Gambar 6.34 Grafik pengujian *push button*

Hasil selanjutnya yaitu pengujian pengiriman *state push button* kepada *host*. Berdasarkan Gambar 6.34, grafik yang digambarkan oleh *client2* berbentuk stabil, sedangkan *client1* tidak stabil. Berdasarkan hasil pengujian tersebut, nilai maksimum pada *client1* yaitu 548,26ms dan *client2* yaitu 246,176ms. Untuk nilai minimum pada *client1* yaitu 44,614ms dan *client2* yaitu 239,78ms. Waktu yang dibutuhkan untuk pengiriman data *push button* yaitu dengan rata-rata 255,696ms.



Gambar 6.35 Grafik pengujian LED

Pada Gambar 6.35 merupakan grafik pengujian data LED. Nilai maksimum waktu pengiriman dari *client1* yaitu 15346,77ms dan *client2* yaitu 3003,918. Sedangkan nilai minimum waktu pengiriman dari *client1* yaitu 3809,271ms dan *client2* yaitu 480,632ms. Dari hasil tersebut dapat dilihat bahwa waktu pengiriman

yang dibutuhkan, seiring berjalannya waktu akan membutuhkan waktu yang lebih lama. Hal ini terjadi karena terdapat perbedaan kecepatan antara pengiriman dan penerimaan, sehingga menyebabkan *buffer* pada bagian penerima. Pada kedua *client* terjadi perbedaan waktu yang sangat jauh, dikarenakan *client1* lebih lama terkoneksi dengan *host* sebelum melakukan pengujian ini sehingga menyebabkan *buffering* data yang menumpuk, dibandingkan dengan *client2* yang terhubung dengan *host* belum terlalu lama. Hasil pengujian LED pada Tabel 6.9, data yang tersampai membutuhkan waktu yaitu dengan rata-rata 5350,926ms.

BAB 7 PENUTUP

7.1 Kesimpulan

Adapun kesimpulan yang dihasilkan dari pengujian kali ini, yaitu:

1. Perangkat dapat mengenali perangkat sekitarnya tanpa konfigurasi dari pengguna yaitu dengan menggunakan metode *pervasive*. Metode ini dapat memudahkan manusia untuk tidak repot-repot men-konfigurasi perangkat. Perangkat akan diprogram untuk menyebarkan informasinya secara otomatis dan dapat menyimpan informasi dari perangkat lain jika ada balasan. Pada penelitian kali ini, protokol pengiriman yang digunakan adalah protokol UDP, sehingga pesan yang dikirimkan ringan dan tidak terlalu membebani memori perangkat.
2. Sistem pada penelitian kali ini diimplementasikan pada perangkat Raspberry Pi 3 dimana sebagai *host* dan NI myRIO sebagai *client*. Perangkat ini diprogram oleh LabVIEW dengan cara men-deploy program pada perangkat. Program dibuat sesuai alur sistem yaitu dengan *state machine* dimana pada masing-masing perangkat dapat melakukan *discovery* agar perangkat saling kenal dan mengetahui informasinya masing-masing tanpa konfigurasi manual.
3. Hasil pengujian untuk *discovery* perangkat pada bagian *host* yaitu 56,417ms, sedangkan proses *discovery* pada *client* didapatkan hasil yaitu 251,067ms. Sehingga untuk proses *discovery* seluruhnya yaitu 313,417ms. Sedangkan jika *host* mengalami kegagalan, waktu yang dibutuhkan *client* untuk terhubung kembali yaitu 10384,23ms. Sedangkan untuk pengiriman dan penerimaan data antar *host* dan client, untuk pengiriman data sensor membutuhkan waktu 58,263ms, untuk pengendalian LED membutuhkan waktu 5350,926ms, dan untuk *push button* membutuhkan waktu 255,696ms.

7.2 Saran

Adapun saran dari penelitian ini, agar dapat dilanjutkan untuk penelitian selanjutnya yang terkait dengan penelitian ini, yaitu:

1. Penambahan fitur *Internet of Things*, agar perangkat dapat dipantau dan dikendalikan secara jarak jauh. Sehingga tidak perlu repot-repot untuk terhubung pada jaringan lokal.
2. Penambahan fitur sensor dan atau aktuator yang berhubungan dengan *home environment*.

DAFTAR PUSTAKA

Elliott, C., Vijayakumar, V., Zink, W. & Hansen, R., 2017. National Instruments LabVIEW: A Programming Environment for Laboratory Automation and Measurement. *SLAS TECHNOLOGY: Translating Life Sciences Innovation,* 12(1), pp. 17-24.

Foundation, R. P., 2018. *Raspberry Pi 3 Model B - Raspberry Pi.* [Online] Available at: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
[Diakses 07 July 2018].

Hamed, B., 2012. Design & Implementation of Smart House Control Using LabVIEW. *International Journal of Soft Computing and Engineering (IJSCE),* 1(6), pp. 2231-2307.

Instruments, N., 2018. *myRIO Student Embedded Device.* [Online] Available at: https://www.ni.com/en-id/shop/select/myrio-student-embedded-device
[Diakses 07 July 2018].

Kurniawan, W., Ichsan, M. H. H., Akbar, S. R. & Arwani, I., 2017. Lightweight UDP Pervasive Protocol in Smart Home Environment Based on Labview. *IOP Conference Series: Materials Science and Engineering.*

Kurose, J. F. & Ross, K. W., 2013. *Computer Networking: A Top-Down Approach (6th Edition).* s.l.:Pearson.

Lee, B. & Lee, E. A., 1998. *Interaction of Finite State Machines and Concurrency Models.* California, University of California at Berkeley.

Ptolemaeus, C., 2014. *System Design, Modelling, and Simulation using Ptolemy II.* s.l.:Ptolemy.org.

Ryzkiansyah, N. E., Kurniawan, W. & Prasetio, B. H., 2017. Perancangan Smart Surveillance System Pada Smarthome Menggunakan NI MyRIO. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* pp. 2854-2860.

Silvis-Cividjian, N., 2017. *Pervasive Computing.* Switzerland: Springer Nature.

Zulfy, I., Syauqy, D. & Akbar, S. R., 2018. Implementasi Pervasive Computing Pada Sistem Monitoring Konsumsi Daya Listrik Stop Kontak Rumah. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* Volume 2, pp. 2555-2561.