Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – IS184853

RANCANG BANGUN SISTEM WIRELESS DISPLAY MENGGUNAKAN RASPBERRY PI

DESIGN AND DEVELOPMENT OF WIRELESS DISPLAY SYSTEM USING RASPBERRY PI

SATRYA BAGUS NURAWAN
NRP. 05211640000079

Dosen Pembimbing
Dr.Eng. Febriliyan Samopa, S.Kom., M.Kom.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR – IS184853

RANCANG BANGUN SISTEM WIRELESS DISPLAY MENGGUNAKAN RASPBERRY PI

SATRYA BAGUS NURAWAN
NRP. 052116400000079

Dosen Pembimbing
Dr.Eng. Febriliyan Samopa, S.Kom., M.Kom.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

Halaman ini sengaja dikosongkan

UNDERGRADUATE THESIS – IS184853

DESIGN AND DEVELOPMENT OF WIRELESS DISPLAY SYSTEM USING RASPBERRY PI

SATRYA BAGUS NURAWAN
NRP. 05211640000079

Supervisor
Dr.Eng. Febriliyan Samopa, S.Kom., M.Kom.

DEPARTMENT OF INFORMATION SYSTEMS
Faculty of Electrical and Intelligent Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

RANCANG BANGUN SISTEM WIRELESS DISPLAY MENGGUNAKAN RASPBERRY PI

TUGAS AKHIR
Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh :

SATRYA BAGUS NURAWAN
NRP. 05211640000079

Surabaya, Januari 2020

KEPALA
DEPARTEMEN SISTEM INFORMASI

Dr. Mudjahidin, S.T., M.T.
NIP. 19701010 200312 1 001

Halaman ini sengaja dikosongkan

LEMBAR PERSETUJUAN

RANCANG BANGUN SISTEM WIRELESS DISPLAY MENGGUNAKAN RASPBERRY PI

TUGAS AKHIR

Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh :

SATRYA BAGUS NURAWAN
NRP. 05211640000079

Disetujui Tim Penguji : Tanggal Ujian : 14 Januari 2020
Periode Wisuda : Maret 2020

Dr.Eng. Febriliyan Samopa, S.Kom., M.Kom. **(Pembimbing I)**

Bekti Cahyo Hidayanto, S.Si., M.Kom. **(Penguji I)**

Nisfu Asrul Sani, S.Kom., M.Sc. **(Penguji II)**

Halaman ini sengaja dikosongkan

RANCANG BANGUN SISTEM WIRELESS DISPLAY MENGGUNAKAN RASPBERRY PI

Nama : Satrya Bagus Nurawan
NRP : 05211640000079
Departemen : Sistem Informasi ITS
Pembimbing I : Dr. Eng.Febriliyan Samopa, S.Kom., M.Kom.

ABSTRAK

Seiring dengan berkembangnya informasi dan teknologi, cakap dalam menyampaikan sesuatu sangat dibutuhkan pada zaman modern ini. Penyampaian materi dapat melalui proyektor ataupun televisi tergantung tempat dimana akan menyampaikan sebuah materi. Namun, dalam menyampaikan materi, diam di tempat adalah hal yang sering kita lihat saat pemateri ingin menyampaikan materi. Hal ini dikarenakan perangkat yang digunakan harus terhubung dengan kabel proyektor. Walaupun sekarang ini sudah terdapat alat wireless presenter. Penggunaan wireless presenter masih kurang interaktif. Dengan pemanfaatan teknologi peer to peer, penggunaan kabel dalam presentasi dapat digantikan.

Dalam menghubungkan perangkat presentator dengan proyektor konvensional atau televisi, digunakan Raspberry Pi sebagai alat perantara. Alat ini memungkinkan presentator untuk menggunakan smartphone sebagai media penyampaian materi. Karena alat ini tidak membutuhkan port, yang pada umumnya tidak dimiliki oleh smartphone, untuk dapat terhubung dengan proyektor.

Untuk dapat menggunakan sistem ini, presentator wajib memasang aplikasi pada perangkatnya. Aplikasi ini bertujuan untuk menangkap gambar pada layar perangkat, lalu mengatur skala resolusi yang nantinya akan dikirim ke Raspberry Pi

dalam bentuk JPG. Pengaturan skala resolusi dapat ditingkatkan maupun diturunkan tergantung pada resolusi perangkat presentator dengan resolusi pada proyektor atau televisi. Sistem pada Raspberry Pi akan menerima data tersebut setiap detiknya dan akan ditampilkan pada proyektor atau televisi. Jaringan antara perangkat presentator dan Raspberry Pi menggunakan jaringan peer to peer. Jaringan peer to peer yang terbuat jaringan berupa point to point. Jadi tidak memungkinkan untuk Raspberry Pi melakukan transfer data bergantian antara perangkat utama dengan perangkat yang lain saat ada lebih dari 1 pengguna terhubung dengan Raspberry Pi.

Hasil yang diharapkan dalam penelitian ini adalah sistem dapat memunculkan gambar dengan rate 10 fps hingga 25 fps. Serta dapat digunakan oleh semua perangkat media presentasi yang dapat terhubung dengan Wi-Fi, sehingga proses penyampaian materi tidak lagi terbatas pada jenis perangkat, port dan kabel.

Kata Kunci : Wireless Display, Raspberry Pi, Wi-Fi, Peer to Peer.

DESIGN AND DEVELOPMENT OF WIRELESS DISPLAY SYSTEM USING RASPBERRY PI

Name	: Satrya Bagus Nurawan
NRP	: 05211640000079
Department	: Information Systems ITS
Supervisor I	: Dr.Eng. Febriliyan Samopa, S.Kom., M.Kom.

ABSTRACT

Along with the development of information and technology, proficient in conveying something is needed in this modern age. Submission of material can be through a projector or television depending on the place where the material will be delivered. However, in delivering material, staying in one place is something that we often see when the speaker wants to deliver the material. This is because the device used must be connected to the projector cable. Although now there is a wireless presenter tool. The use of wireless presenters is still less interactive. With the use of peer to peer technology, the use of cables in presentations can be replaced.

In connecting the presentator device with a conventional projector or television, Raspberry Pi is used as an intermediary tool. This tool allows the presenter to use smartphones as a medium for delivering material. Because this tool does not need a port, which is generally not owned by a smartphone, to be able to connect with the projector.

To be able to use this system, the presentator must install the application on his device. This application aims to capture images on the device's screen, then adjust the resolution scale which will later be sent to Raspberry Pi in JPG format. The resolution scale setting can be increased or decreased depending on the resolution of the presentator device with the resolution on the projector or television. The system on the

Raspberry Pi will receive that data every second and will be displayed on a projector or television. The network between the presentation device and the Raspberry Pi uses a peer to peer network. Peer-to-peer networks are made of point-to-point networks. So it is not possible for Raspberry Pi to transfer data alternately between the main device and other devices when there are more than 1 user connected to the Raspberry Pi.

The expected results in this study are that the system can display images with a rate of 10 fps up to 25 fps. And can be used by all presentation media devices that can be connected with Wi-Fi, so the process of delivering material is no longer limited to the types of devices, ports and cables.

Keywords : Wireless Display, Raspberry Pi, Wi-Fi, Peer to Peer.

SURAT PERNYATAAN BEBAS PLAGIARISME

Saya yang bertandatangan di bawah ini:

Nama	: Satrya Bagus Nurawan
NRP	: 05211640000079
Tempat / Tanggal lahir	: Sidoarjo / 19 Januari 1998
Fakultas / Departemen	: FTEIC / Sistem Informasi
Nomor Telp/Hp/email	: 085853188974/ satryabagusn@gmail.com

Dengan ini menyatakan dengan sesungguhnya bahwa penelitian/makalah/tugas akhir saya yang berjudul

RANCANG BANGUN SISTEM WIRELESS DISPLAY MENGGUNAKAN RASPBERRY PI

Bebas Dari Plagiarisme Dan Bukan Hasil Karya Orang Lain.

Apabila dikemudian hari ditemukan seluruh atau sebagian penelitian/makalah/tugas akhir tersebut terdapat indikasi plagiarisme, maka saya bersedia menerima sanksi sesuai peraturan dan ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat dengan sesungguhnya dan untuk dipergunakan sebagaimana mestinya.

Surabaya, 20 Januari 2020

Satrya Bagus Nurawan
NRP.052116400000079

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT, Tuhan Semesta Alam yang telah memberikan kekuatan serta hidayah-Nya kepada penulis sehingga penulis dapat menyelesaikan tugas akhir ini yang merupakan salah satu syarat kelulusan di Departemen Sistem Informasi Fakultas Teknologi Elektro dan Informatika Cerdas Institut Teknologi Sepuluh Nopember Surabaya.

Terima kasih penulis sampaikan kepada pihak-pihak yang telah mendukung, memberikan saran, motivasi, semangat, dan bantuan baik berupa materiil maupun moril demi tercapainya tujuan pembuatan tugas akhir ini. Secara khusus penulis akan menyampaikan ucapan terima kasih yang sedalam-dalamnya kepada :

1. Segenap keluarga besar terutama kedua orang tua dan kakak penulis, Bapak Ir. Achmad Agung N. M.T., Ibu Ir. Jekti Trisila U., Shinta Nurmaraya F., dan Peny Nurmaraya M. yang senantiasa mendoakan, memberikan motivasi dan semangat, sehingga penulis mampu menyelesaikan pendidikan sarjana ini dengan baik.
2. Ibu Mahendrawathi Er, S.T., M.Sc., Ph.D. selaku Kepala Departemen Sistem Informasi ITS, Bapak Nisfu Asrul Sani, S. Kom., M. Sc. selaku Ketua Program Studi Sarjana Departemen Sistem Informasi ITS, serta seluruh dosen pengajar beserta staf dan karyawan Departemen Sistem Informasi ITS selama penulis menjalani perkuliahan.
3. Bapak Hatma Suryotrisongko, S. Kom., M.Eng. dan Ibu Nur Aini Rakhmawati, S. Kom., M.Sc.Eng., Ph.D. sebagai dosen wali penulis selama menempuh pendidikan di Departemen Sistem Informasi ITS.
4. Bapak Dr.Eng. Febriliyan Samopa, S. Kom., M. Kom. selaku dosen pembimbing yang telah banyak meluangkan waktu untuk membimbing, mengarahkan,

dan mendukung dengan memberikan ilmu, petunjuk, dan motivasi dalam penyelesaian tugas akhir ini.

5. Bapak Nisfu Asrul Sani, S. Kom., M.Sc. dan Bapak Bekti Cahyo Hidayanto, S.Si., M. Kom. selaku dosen penguji yang telah memberikan kritik dan saran dalam penyempurnaan tugas akhir ini.
6. Mbak Silfyana Dita dan Sisca Threecya Agatha yang telah membantu penulis untuk meminjamkan *laptop* dan *handphone*-nya untuk keperluan tugas akhir.
7. Teman-teman Sistem Informasi angkatan 2016 (Artemis) yang senantiasa menemani dan memberikan motivasi bagi penulis selama perkuliahan hingga dapat menyelesaikan tugas akhir ini.
8. Teman-teman laboratorium ADDI, RDIB, SE, dan MSI yang telah mempersilakan penulis bernaung dan mencari inspirasi dalam mengerjakan tugas akhir ini.
9. Seluruh pihak-pihak lainnya yang tidak dapat disebutkan satu per satu, yang telah membantu penulis selama perkuliahan hingga dapat menyelesaikan tugas akhir ini.

Penyusunan laporan tugas akhir ini masih jauh dari kata sempurna, sehingga penulis menerima adanya kritik maupun saran yang membangun untuk perbaikan di masa yang akan datang. Semoga buku tugas akhir ini dapat memberikan manfaat bagi pembaca.

Surabaya, 20 Januari 2020

Penulis

Satrya Bagus Nurawan

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

Pada bagian ini akan diuraikan proses identifikasi masalah penelitian yang meliputi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan relevansi terhadap pengerjaan tugas akhir. Berdasarkan uraian pada bagian ini, harapannya gambaran umum permasalahan dan pemecahan masalah pada tugas akhir ini dapat dipahami.

1.1. Latar Belakang

Pertumbuhan teknologi saat ini berkembang sangat pesat. Banyak inovasi teknologi yang terus bermunculan tiap tahunnya. Salah satunya adalah inovasi dalam menciptakan teknologi yang menggunakan jaringan *wireless*. Jaringan *wireless* merupakan teknologi yang dapat membuat jaringan komputer tanpa kabel dengan menggunakan gelombang frekuensi tinggi, frekuensi 2.4Ghz dan frekuensi 5Ghz [1]. Dengan adanya teknologi yang menggunakan jaringan *wireless*, seseorang dapat bergerak atau beraktifitas kemana dan dimanapun untuk melakukan komunikasi data. Komputer, laptop dan *smartphone* mendominasi penggunaan teknologi *wireless*. Jumlah pengguna *smartphone* di Indonesia dari tahun 2017 hingga 2018 telah meningkat secara signifikan. Peningkatan yang terjadi sebesar 8.6 juta unit, sebesar 74.9 juta unit pada tahun 2017 dan 83.5 juta unit pada tahun 2018 [2]. Berdasarkan data dari Google, pengguna *smartphone* di Indonesia pada tahun 2017 adalah 60%. Sedangkan pengguna komputer adalah sebesar 22% dan pengguna tablet hanya sebesar 8% [3].

Tidak semua perangkat keras dapat saling terhubung melalui jaringan *wireless*. Banyak faktor yang mempengaruhi mengapa perangkat keras tidak dapat menggunakan teknologi *wireless*. Salah satu faktor yang paling banyak saat ini adalah perangkat keras tersebut memang tidak didesain untuk menggunakan jaringan *wireless*, biasanya dijumpai pada perangkat keras

produksi lama. Seiring dengan berkembangnya zaman, perangkat keras yang mulanya tidak menggunakan teknologi *wireless* sekarang semua produsen saling berlomba untuk menambahkan teknologi *wireless* pada produknya. Namun, banyak individu atau organisasi yang sudah memiliki perangkat keras produksi lama yang tidak menggunakan teknologi *wireless* dan memutuskan untuk tidak membeli perangkat yang baru karena biaya yang cukup mahal. Proyektor LCD adalah contoh aset yang sudah lama dimiliki individu atau kelompok. Berdasarkan hasil pengamatan, banyak sekolah, instansi dan organisasi yang masih menggunakan proyektor LCD versi lama untuk melakukan aktifitas presentasi. Karena masih menggunakan proyektor LCD versi lama, untuk melakukan presentasi harus menggunakan komputer atau laptop. Komputer atau laptop ini kemudian dihubungkan melalui kabel agar dapat terhubung dengan proyektor LCD. Penggunaan kabel menyebabkan presentator tidak dapat berpindah tempat dan tidak bisa jauh dari perangkat yang digunakan untuk presentasi. Hal ini menyebabkan presentasi yang terjadi tidak interaktif. Walaupun sudah terdapat *wireless* presenter, perangkat ini masih tidak dapat memunculkan sifat fleksibilitas perangkat presentasi.

Penggunaan proyektor LCD semakin meningkat karena penyampaian materi melalui multimedia dapat membuat materi yang abstrak menjadi lebih riil [4]. Meningkatnya aktifitas presentasi dan pengguna *smartphone*. Sekarang banyak bermunculan aplikasi presentasi pada *smartphone*. Tetapi *smartphone* sendiri tidak dirancang untuk dapat berkomunikasi dengan proyektor LCD menggunakan kabel.

Dengan menggunakan perantara Raspberry Pi, perangkat yang tidak dirancang untuk saling berkomunikasi menjadi dapat saling berkomunikasi. Raspberry Pi dapat bertindak layaknya komputer biasa pada umumnya. Tetapi karena ukurannya yang kecil, spesifikasi yang dimiliki tidak sehebat komputer. Penggunaan Raspberry Pi sebagai penghubung antara perangkat presentasi dengan proyektor LCD dikarenakan

ukurannya yang kecil dengan kemampuan seperti komputer. Serta Raspberry Pi memiliki fitur Wi-Fi didalamnya [5].

Pada penelitian sebelumnya yaitu pada tahun 2016 oleh Yusman Fauzan dan Bobi Kurniawan dengan judul penelitian "Rancang Bangun Perangkat Wireless untuk Projector Konvensional". Bahwa penelitian tersebut dalam menghubungkan Raspberry Pi dengan perangkat presentasi harus menggunakan modul Wi-Fi tipe TP-LINK TL-WN832N yang akan berfungsi sebagai *Access Point* (AP) [6]. Sistem yang dibangun menggunakan aplikasi yang sudah dibuat sebelumnya, TightVNC. TightVNC adalah *software remote* gratis dan open source yang menghubungkan perangkat *client* dengan Raspberry Pi. Cara kerja dari TightVNC adalah menerima gambar tiap beberapa milidetiknya dan menampilkan gambar pada output Raspberry Pi.

Sehingga pada tugas akhir kali ini dikembangkan sebuah sistem aplikasi menggunakan Raspberry Pi sebagai perantara antara perangkat presentasi dengan proyektor LCD menggunakan jaringan *wireless*. Dimana jaringan *wireless* yang digunakan menggunakan teknologi *peer-to-peer* (P2P) yang tidak membutuhkan modul Wi-Fi tambahan pada Raspberry Pi dan tidak membutuhkan AP agar dapat terhubung.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah dikemukakan sebelumnya, maka rumusan masalah pada tugas akhir ini adalah sebagai berikut :

1. Bagaimana membuat aplikasi penangkap layar pada Android, iOS, Windows?
2. Bagaimana membuat resolusi yang didapat dari perangkat hingga pada tampilan akhir menjadi optimal?
3. Bagaimana membuat waktu transfer data yang terjadi menjadi cepat?
4. Bagaimana menghubungkan aplikasi perangkat dengan Raspberry Pi menggunakan jaringan *peer-to-peer*?

5. Bagaimana membuat antarmuka aplikasi yang dapat mengontrol sistem yang telah dibuat?

1.3. Batasan Masalah

Berdasarkan permasalahan yang telah diuraikan sebelumnya, adapun batasan masalah dalam tugas akhir ini adalah sebagai berikut :

1. Sistem tidak dirancang untuk gambar yang bergerak dan memiliki audio.
2. Sistem dirancang untuk perangkat *client* yang memiliki modul Wi-Fi.
3. Antarmuka aplikasi yang dibuat untuk memulai, memberhentikan dan memutuskan hubungan dengan sistem.

1.4. Tujuan

Berdasarkan latar belakang dan rumusan masalah yang telah disebutkan sebelumnya, adapun tujuan dari tugas akhir ini adalah sebagai berikut:

1. Terbentuknya aplikasi penangkap layar pada Android, iOS dan Windows.
2. Terbentuknya sistem yang dapat menampilkan keluaran gambar pada resolusi yang optimal.
3. Terbentuknya sistem yang dapat menampilkan gambar pada *scaling* resolusi yang tepat pada rentang 10 fps hingga 25 fps.
4. Terhubungnya aplikasi perangkat dengan Raspberry Pi menggunakan jaringan *peer-to-peer*.
5. Terbentuknya aplikasi pada perangkat *client* dengan perintah connect, play, pause, disconnect, logout, dan shutdown.

1.5. Manfaat

Manfaat yang diharapkan dapat diperoleh dari tugas akhir ini adalah memudahkan individu atau kelompok yang ingin melakukan aktifitas presentasi agar tidak terhalang oleh jenis port dan panjang kabel yang terhubung dengan proyektor.

Sehingga presentator dapat lebih interaktif dalam menyajikan presentasinya dan tidak terbatas hanya pada laptop atau komputer saja sebagai media presentasinya.

1.6. Relevansi

Tugas akhir ini disusun untuk memenuhi salah satu syarat kelulusan sebagai Sarjana Komputer di Departemen Sistem Informasi Fakultas Teknologi Informasi dan Komunikasi (FTIK) Institut Teknologi Sepuluh Nopember (ITS). Tugas akhir ini sesuai dengan penerapan mata kuliah dari laboratorium Infrastruktur dan Keamanan Teknologi Informasi (IKTI) Departemen Sistem Informasi FTIK ITS, yaitu Desain Manajemen Jaringan dan Komputer. Selain itu, juga berkaitan dengan mata kuliah wajib, yaitu Teknologi Bergerak serta Konstruksi dan Pengujian Perangkat Lunak.

Halaman ini sengaja dikosongkan

BAB II
TINJAUAN PUSTAKA

Pada bagian ini akan menjelaskan mengenai studi literatur terhadap penelitian sebelumnya dan dasar teori yang dijadikan acuan atau landasan dalam pengerjaan tugas akhir ini.

2.1. Studi Literatur

Dalam proses pengerjaan tugas akhir ini, dilakukan pencarian penelitian - penelitian yang sudah dilakukan untuk dijadikan sebagai referensi dalam pengerjaan tugas akhir ini.

Tabel 2. 1 Studi literatur

1) Rancang Bangun Perangkat Wireless untuk Projector Konvensional
Penulis; Tahun F. Yauzan; 2015 [6]
Pembahasan Penelitian ini menghubungkan antara laptop dan Raspberry Pi yang terhubung oleh proyektor dengan menjadikan modul Wi-Fi sebagai *Access Point* (AP). Pengguna wajib memasukkan *security key* yang terdapat pada modul Wi-Fi. Alat yang dihasilkan memungkinkan lebih dari 1 pengguna dalam satu waktu dengan jarak maksimal antara laptop dengan modul Wi-Fi adalah 20 hingga 25 meter. Sistem yang dibangun menggunakan aplikasi TightVNC. TightVNC bekerja dengan menangkap gambar lalu menampilkannya pada output Raspberry Pi tiap milidetiknya
Keterkaitan Penelitian tugas akhir yang dilakukan berkaitan dengan Raspberry Pi sebagai media penghubung antara end-device(laptop) dengan proyektor. Serta fungsi dari alat yang akan menampilkan tampilan pada *end device* ke layar LCD

proyektor. Cara kerja yang sama mengirimkan gambar tiap milidetiknya.
2) Rancang Bangun Sistem Printer Tanpa Kabel Berbasis Bluetooth dan WIFI
Penulis; Tahun Ahmad Irfan Yusuf; 2014 [7]
Pembahasan Komunikasi yang terjadi antara Raspberry Pi dan end device dengan menggunakan bluetooth dengan Wi-Fi berbeda. Jalur yang terbuat menggunakan bluetooth adalah point to point. Pada jaringan bluetooth tidak memungkinkan untuk banyak user terhubung dengan Raspberry Pi. Sedangkan jalur yang terbuat jika menggunakan Wi-Fi adalah *point to multipoint*. Sangat memungkinkan untuk banyak user dapat terhubung pada 1 Raspberry Pi. Saat banyak user terhubung dan ingin melakukan perintah. User pertamalah yang dapat memberikan perintah print. Setelah printer selesai dengan perintah user pertama, baru printer menjalankan perintah dari user kedua yang terhubung.
Keterkaitan Penelitian tugas akhir yang dilakukan berkaitan dengan alur komunikasi menggunakan Wi-fi antara user, Raspberry Pi dan output hardware. Yang memungkinkan banyak user dapat terhubung dengan 1 Raspberry Pi.
3) Penerapan Screen Mirroring Android Pada Projector Menggunakan Raspberry Pi
Penulis; Tahun Dwijaya Kongky Salim, Justinus Andjawirawan, Lily Puspa Dewi; 2019 [8]

Pembahasan Perekaman pada android menggunakan Media Projector dan dijadikan file berupa file stream. File ini akan dikirim kepada raspberry pi menggunakan framework Gstreamer. Pada perangkat raspberry pi menerima file stream dari Android menggunakan framework Gstreamer dan menampilkannya pada display yang terpasang.
Keterkaitan Penelitian tugas akhir yang dilakukan berkaitan dengan perekaman layar pada Android yang menggunakan Media Projection. File rekaman yang didapat dijadikan gambar dan di-compress menggunakan encoder jpg.

2.2. Dasar Teori

2.2.1. Peer-to-peer

Peer-to-peer (P2P) merupakan salah satu model jaringan yang menghubungkan dua *peer* atau lebih secara langsung atau tanpa perantara, dimana setiap *peer* dalam jaringan tersebut dapat saling berbagi. *Peer-to-peer* pertama kali dikenal sebagai konsep *host-to-host*. Pada konsep ini, peer yang saling terhubung bertindak sebagai *client* dan *server* secara bersamaan [9].

Menurut Choon, Sarana dan Rajkumar pada paper yang berjudul "Peer-to-Peer Network for Content Sharing" [10] menyebutkan bahwa model jaringan *peer-to-peer* memiliki dua karakteristik utama, yaitu:

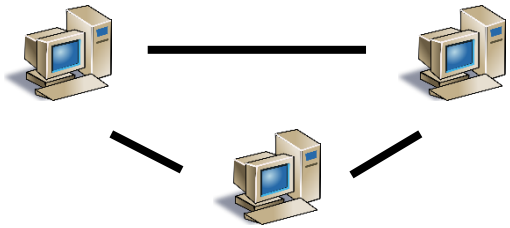
- Skalabilitas: tidak ada limitasi ukuran dari sistem
- Reliabilitas: peer yang mengalami malfungsi tidak akan memengaruhi sistem atau peer lainnya.

2.2.2. Arsitektur Peer-to-peer

Saat ini terdapat 3 model jaringan *peer-to-peer* yaitu terpusat, desentralisasi dan hibrida. Arsitektur jaringan *peer-to-peer* terpusat menggunakan *server* sebagai pusat indeks untuk wadah *database* antar *peer*. *Database* ini akan diperbarui setiap *peer* akan masuk kedalam jaringan.

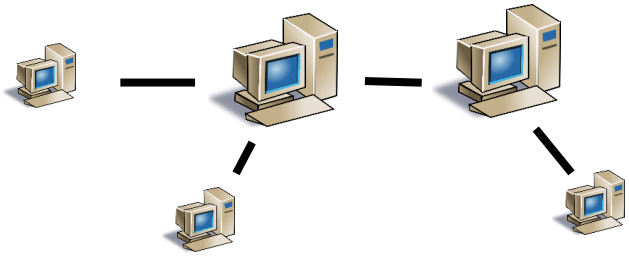
Gambar 2. 1 Arsitektur Jaringan P2P Terpusat

Arsitektur jaringan *peer-to-peer* desentralisasi tidak menggunakan *server* sebagai pusat indeks. Pada aristektur ini semua peer bertindak sebagai server ataupun *client*. Arsitektur ini dirancang untuk menghindari kelemahan pada arstektur terpusat.



Gambar 2. 2 Arsitektur Jaringan P2P Desentralisasi

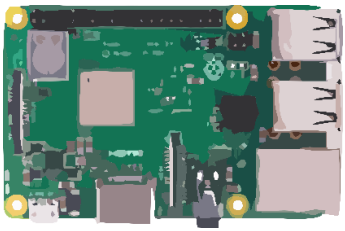
Arsitektur yang terakhir adalah hibrida. Arsitektur ini adalah gabungan dari arsitektur terpusat dengan arsitektur desentralisasi. Pada arsitektur ini, terdapat *peer* pusat dan *client*. *Peer client* adalah *peer* yang hanya terhubung pada peer pusat. Sedangkan peer pusat adalah peer yang terhubung oleh banyak peer. Saat *peer client* masuk kedalam jaringan, peer ini akan menyimpan informasi *peer* lainnya yang terhubung dengan *peer* pusat. Jika satu *peer* ingin berhubungan dengan *peer* lainnya, *peer* harus mengambil informasi dari *peer* pusat.



Gambar 2. 3 Arsitektur jaringan P2P hibrida

2.2.3. Raspberry Pi

Raspberry Pi merupakan *Single Board Computer* (SBC) yang memiliki system on a *chip* dari Broadcomm BCM2835 (SoC), yang mencakup ARM1176JZF-S 700 MHz *processor*, GPU Video Core IV dan RAM 256 MB untuk Rev. A dan RAM 512 MB untuk Rev. B [5]. Raspberry Pi adalah modul *micro* komputer yang juga mempunyai *input output digital port* seperti pada board microcontroller. Raspberry Pi dibuat di



Gambar 2. 4 Board Raspberry Pi

inggris oleh Raspberry Pi Foundation. Raspberry Pi Foundation yang awalnya membuat perangkat ini untuk menarik minat anak - anak agar dapat menjadi seorang pengembang atau *developer* dibidang perangkat keras maupun perangkat lunak kemudian telah dirilis dengan lisensi. *Open-Source Hardware* yang berarti rancangan perangkat kerasnya dapat dipelajari, dimodifikasi, didistribusi dan dijual pada public [11].

Raspberry Pi memiliki beberapa fitur seperti memori eksternal, port usb, port HMDI.

2.2.3.1 Port HDMI

Raspberry Pi mempunyai *output* port HDMI yang dapat dihubungkan dengan sebagian besar tv modern dan monitor komputer. Penggunaan kabel HDMI untuk menghubungkan antaran Raspberry Pi dengan proyektor.

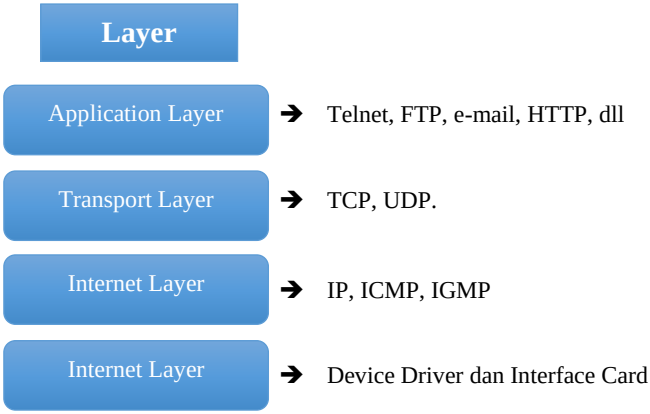
2.2.3.2 Built-in Wi-Fi

Modul Wi-Fi bawaan dari Raspberry Pi dapat beroperasi pada frekuensi 2.4 GHz dan 5 GHz (802.11n Wi-Fi). Penggunaan Wi-Fi sebagai penghubung antara perangkat presentasi dengan Raspberry Pi melalui jaringan *peer-to-peer*.

2.2.4. TCP/IP

TCP/IP (Transport Control Protocol/Internet Protocol) adalah sekumpulan protokol yang mampun mengatur komunikasi data komputer dan memungkinkan komputer untuk berkomunikasi dengan baik meskipun komputer tersebut berbeda jenis, vendor, maupun sistem operasi . TCP/IP dimodelkan dalam empat layer:

1. *Application layer*, merupakan *layer* program aplikasi.
2. *Transport layer*, berisi protokol yang bertanggung jawab untuk mengadakan komunikasi antar dua komputer.
3. *Internet layer*, berfungsi untuk menangani pergerkan paket data dalam jaringan dari komputer pengirim ke komputer tujuan.
4. *Network Layer*, merupakan *layer* paling bawah ynag bertanggungjawab mengirim dan menerima data.



Gambar 2. 5 Layer Pada TCP/IP

2.2.4.1 SSH

SSH (*Secure Shell*) adalah protokol pada *application layer* yang menggunakan port 22. Memfasilitasi jalur komunikasi yang aman antara *client* server dengan teknik enkripsi dan dekripsi di dalam koneksinya. Untuk menggunakan SSH memerlukan otentifikasi username dan password [12].

Penggunaan SSH banyak digunakan untuk remote server jarak jauh, *port forwarding*, kombinasi FTP akan dapat melakukan transfer file secara aman (SFTP) [13].

2.2.4.2 TCP

TCP (*Transmission Control Protocol*) adalah salah satu protokol pada lapisan transport TCP/IP yang bersifat *connection-oriented* dan dapat diandalkan (*reliable*) dalam mengirimkan data [14]. TCP memiliki tujuh fitur utama yaitu sebagai berikut:

1. *Connection oriented*, aplikasi meminta koneksi dan menggunakannya dalam transfer data.
2. *Point-to-point* communication, setiap koneksi TCP memiliki pasti dua titik.

3. *Reliability*, TCP menjamin bagi data yang dikirimkan dalam koneksi dapat terkirim dengan pasti tanpa ada yang hilang atau redudansi.
4. *Full-duplex* connection, koneksi TCP memperbolehkan data untuk berkoneksi dari satu titik koneksi setiap saat.
5. *Stream interface*, TCP memperbolehkan aplikasi untuk mengirimkan koneksi yang berkesinambungan.
6. *Reliable startup*, membutuhkan persetujuan dari kedua aplikasi untuk melakukan koneksi.
7. *Graceful shutdown*, aplikasi dapat membuka aplikasi, mengirim data dan menutup koneksi serta menjamin bahwa data sampai sebelum koneksi terputus.

2.2.4.3 UDP

UDP (*User Datagram Protocol*) adalah salah satu protokol lapisan transport TCP/IP yang mendukung komunikasi yang *unreliable*, *connectionless* antara *host-host* dalam jaringan. Kelebihan dari UDP adalah dapat mengirimkan data dengan cepat[14]. Serta, maksimal ukuran paket UDP adalah 65535 byte dan UDP tidak menyediakan mekanisme *flow-control*, seperti yang dimiliki oleh TCP.

2.2.5. SQLite

SQLite adalah library basis data open source yang menerapkan *self-contained, serverless, zero configuration, dan transactional*. Sifat *self-contained* adalah tidak membutuhkan dukungan dari library eksternal. *Serverless* adalah memungkinkan SQLite dapat dikedalikan dari jarak jauh. *Zero configuration* adalah SQLite tidak membutuhkan proses instalasi. Sifat terakhir adalah *transactional*, yaitu SQLite merapkan transaksional database [15].

2.2.6. Base64

Base64 adalah salah satu cara *encoding* dan *decoding* yang mengubah karakter nonASCII menjadi ASCII. Fungsi dari base64 adalah untuk menyembunyikan data penting atau mengubah suatu gambar menjadi bentuk string jika gambar

akan dikirim melalui RESTful API atau *web service* lainnya hanya dapat menerima input berupa JSON atau form data. Karakter yang didapatkan pada transformasi Base64 ini adalah dari 0...9, A..Z dan a...z, serta ditambah adanya2 karakter terakhir + dan / serta sebuah karakter sama dengan (=) yang berfungsi untuk menyesuaikan dan menggenapkan data binari [16].

Teknik *encoding* Base64 pada dasarnya sederhana, contohnya pada satu (string) byte yang akan disandikan ke Base64 caranya adalah sebagai berikut [17]:

i. Pecahan string diubah menjadi bentuk ASCII.
ii. Pecahan *string bytes* menjadi ke per-3 *bytes*.
iii. Satukan 3 *bytes* menjadi 24 bits. Dengan syarat 1 *bytes*=8 bit, sehingga 3x8=24 bits.
iv. Selanjutnya 24 bits yang tersimpan di-*buffer* dipecahkan menjadi 6 bits, maka akan mendapatkan 4 pecahan.
v. Pecahan masing-masing diubah ke dalam nilai decimal, dengan maksimal nilai 6 bit adalah 63.
vi. Terakhir, nilai-nilai decimal tersebut ubah menjadi indeks untuk memilih karakter penyusunan dari base64 dan maksimal adalah 63 atau indeks ke 64.

Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Gambar 2. 6 Index Base64

2.2.7. SHA512

SHA(Secure Hash Algorithm)adalah salah satu fungsi hash satu arah yang diciptakan oleh National Instute of Standards and

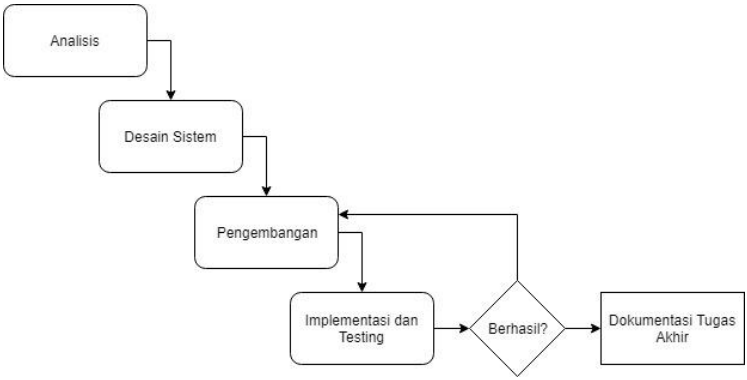
Technology (NIST) [18]. SHA-512 menghasilkan hash terpanjang daripada SHA lainnya. Panjang dari hash menggunakan metode ini adalah 512 bit.

2.2.8. Rpi-Play

Rpi-Play adalah aplikasi *opensource* yang mengimplementasikan raspberry pi sebagai server mirroring AirPlay [19]. Rpi-Play dikembangkan pertama kali oleh dsafa22 yang diambil dari sumber AirplayServer milik KqSMea8 [20]. Aplikasi ini membutuhkan package tambahan agar dapat diinstal pada Raspbian, seperti cmake, libavahi-compat-libdnssd-dev, libssl-dev, dan il*client*.

BAB III
METODOLOGI

Pada bagian metodologi akan menjelaskan bagaimana langkah pengerjaan tugas akhir dengan menggunakan metode SDLC (*System Development Life Cycle*) Iterative yang telah disesuaikan beserta deskripsi dan penjelasannya.



Gambar 3. 1 Alur Pengerjaan Tugas Akhir

3.1. Tahap Analisis

Pada tahap ini dilakukan analisa terhadap kebutuhan sistem yang akan dibuat berdasarkan tujuan dari sistem dan kebutuhan perangkat keras dan perangkat lunak. Sehingga keluaran pada tahap ini adalah terbentuknya spesifikasi kebutuhan dari perangkat lunak dan perangkat keras yang akan dibuat.

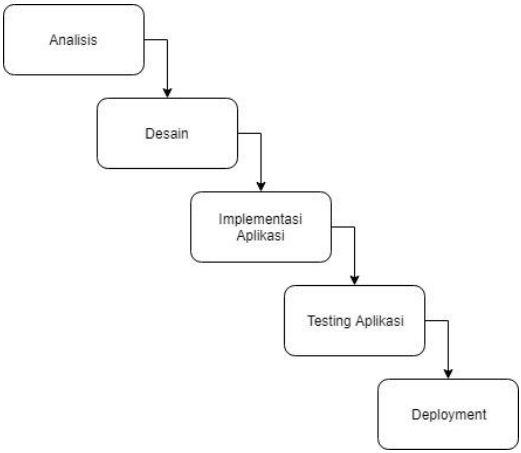
3.2. Tahap Desain Sistem

Pada tahap ini dilakukan desain dari sistem yang akan dibuat. Desain sistem yang akan dibuat menjelaskan bagaimana alur sebuah data diambil hingga menghasilkan keluaran yang diinginkan. Sehingga keluaran pada tahap ini adalah hasil analisa berupa alur sistem yang dibuat.

3.3. Tahap Pengembangan

Pada tahapan ini akan dilakukan proses pembuatan aplikasi pada perangkat *client* dan pembuatan aplikasi pada Raspberry

Pi yang dapat menerima data dari beberapa operating system yang berbeda-beda. Metodologi yang digunakan untuk pembuatan aplikasi yang akan digunakan oleh perangkat *client* adalah metode SDLC (*Software Development Life Cycle*) waterfall. Serta untuk metodologi yang digunakan untuk pembuatan aplikasi yang akan digunakan untuk Raspberry Pi juga menggunakan metode SDLC (*Software Development Life Cycle*) waterfall. Penjelasan proses tahapan pengembangan perangkat lunak adalah sebagai berikut:



Gambar 3. 2 Alur Pengerjaan Perangkat Lunak

3.3.1 Analisis

3.3.1.1 Perangkat *Client*

Pada tahap ini dilakukan analisa terhadap kebutuhan perangkat lunak yang akan dibuat. Sehingga keluaran pada tahap ini adalah terbentuknya spesifikasi kebutuhan dari perangkat lunak dengan 3 jenis sistem operasi yang berbeda.

3.3.1.2 Raspberry Pi

Pada tahap ini dilakukan analisa terhadap kebutuhan perangkat lunak yang akan dibuat. Sehingga keluaran pada tahap ini adalah terbentuknya spesifikasi kebutuhan dari perangkat lunak yang akan digunakan untuk Raspberry Pi.

3.3.2 Desain

3.3.2.1 Perangkat *Client*

Pada tahap ini dilakukan desain dari perangkat lunak yang akan dibuat. Desain perangkat yang akan dibuat menjelaskan bagaimana alur sebuah data diambil hingga menghasilkan keluaran yang diinginkan. Sehingga keluaran pada tahap ini adalah hasil analisa berupa kebutuhan fungsional dan desain dari sistem.

3.3.2.2 Raspberry Pi

Pada tahap ini dilakukan desain dari perangkat lunak yang akan dibuat. Desain sistem yang akan dibuat menjelaskan bagaimana alur sebuah data diambil hingga menghasilkan keluaran yang diinginkan. Sehingga keluaran pada tahap ini adalah hasil analisa berupa kebutuhan fungsional dan desain dari sistem.

3.3.3 Implementasi Aplikasi

3.3.3.1 Perangkat *Client*

Pada tahapan ini akan dilakukan proses pembuatan aplikasi pada beberapa operating system yang akan digunakan sebagai aplikasi *client*. Masing-masing aplikasi perangkat *client* menggunakan protokol UDP untuk message control 6 perintah utama (connect, play, pause, disconnect, logout, dan shutdown) dan menggunakan protokol TCP untuk transportasi data gambar. Serta dapat melakukan enkripsi token menggunakan metode RSA yang akan digunakan untuk melakukan pengecekan terhadap token dari paket gambar.

3.3.3.2 Raspberry Pi

Pada tahapan ini akan dilakukan proses pembuatan aplikasi pada Raspberry Pi yang dapat menerima data melewati protokol UDP untuk message control (connect, play, pause, disconnect, logout, dan shutdown) dan TCP untuk menerima data gambar dari beberapa *operating system* yang berbeda-beda. Serta dapat melakukan enkripsi token yang akan dikirimkan dalam paket gambar.

Agar aplikasi iOS menggunakan sistem yang dibuat, diperlukan tambahan software opensource Rpi-Play untuk raspberry pi. Rpi-Play digunakan karena kebijakan dari Apple untuk developer agar tidak dapat melakukan screenshot diluar aplikasi yang dikembangkan.

3.3.4 Testing Aplikasi

3.3.4.1 Perangkat *Client*

Pada tahap ini akan dilakukan proses pengujian terhadap aplikasi yang telah dibuat. Berikut merupakan daftar pengujian yang akan dilakukan untuk menguji keandalan dari aplikasi yang dibuat:

- *Unit Testing*
- *Integration Test*ing

3.3.4.2 Raspberry Pi

Pada tahap ini akan dilakukan proses pengujian terhadap aplikasi yang telah dibuat. Berikut merupakan daftar pengujian yang akan dilakukan untuk menguji keandalan dari aplikasi yang dibuat:

- *Unit Testing*
- *Integration Testing*

3.3.5 Deployment Aplikasi

3.3.5.1 Perangkat *Client*

Pada tahap ini akan dilakukan proses perbaikan *bug* dan pengujian *bug* yang ditemukan pada tahapan *testing*. Sehingga keluaran dari tahap ini adalah aplikasi perangkat pada masing-masing *operating system* bebas dari *bug* dan siap untuk digunakan.

3.3.5.2 Raspberry Pi

Pada tahap ini akan dilakukan proses perbaikan *bug* dan pengujian *bug* yang ditemukan pada tahapan testing. Sehingga keluaran dari tahap ini adalah aplikasi Raspberry Pi bebas dari *bug* dan siap untuk digunakan.

3.4. Tahap Implementasi dan Testing Sistem

Pada tahap ini akan dilakukan pembuatan sistem yang terdiri dari aplikasi perangkat *client*, aplikasi pada raspbebrry pi dan dispay output. Dari sistem yang dibuat akan dilakukan proses pengujian terhadap sistem yang telah dibuat. Pengujian menggunakan metode performance testing *(black box testing)*. Pengujian diprioritaskan kepada pengujian fungsional. Namun pengujian nonfungsional juga dilakukan untuk mengetahui keandalan sistem. Berikut merupakan daftar pengujian yang akan dilakukan untuk menguji keandalan dari sistem yang dibuat:

3.4.1 Testing Fungsional

- Pengujian waktu delay antara tampilan *client* dengan output display.
- Pengujian *multiuser* pada sistem.
- Pengujian hasil fps yang didapat berdasarkan:
    - Jenis perangkat keras dan port yang digunakan pada output *display*.
    - Resolusi yang digunakan pada *output display* dan resolusi perangkat *client*.
    - Pengujian hasil gambar yang didapat terhadap adanya *jammer* sinyal Wi-Fi.

3.4.2 Testing non-Fungsional

- Pengujian jarak jaringan.

Jika masih ditemukan *bug* pada tahapan ini, proses kembali lagi pada tahapan pengembangan. Tahapan pengembangan adalah tahapan pembuatan dari masing-masing aplikasi. Sehingga hasil pada tahap ini adalah sistem telah bebas dari *bug* yang ditemukan pada tahapan testing dan sistem siap untuk diluncurkan.

3.5. Dokumentasi Tugas Akhir

Adapun dokumen tugas akhir merupakan *output* dari keseluruhan tahapan tugas akhir yang telah dilakukan sesuai

metodologi. Serta kesimpulan serta saran untuk penelitian kedepannya.

BAB IV
PERANCANGAN

Pada bagian ini akan menjelaskan mengenai proses perancangan terhadap sistem yang akan digunakan sebagai acuan dalam proses implementasi.

4.1 Tahap Analisis

Pada tahap awal dari pembuatan sistem dimulai dengan mengidentifikasi seluruh informasi dari sistem yang dibuat.

Sistem terbagi menjadi 2 bagian, yaitu aplikasi yang berfungsi sebagai mengirim data dan aplikasi untuk menerima gambar. Aplikasi pengirim data terdiri dari Windows, Android dan iOS. Sedangkan aplikasi penerima gambar adalah aplikasi yang ada di Raspberry Pi.

Aplikasi pengirim gambar/aplikasi *client* memerlukan ijin untuk mendapatkan gambar pada layar secara penuh. Gambar ini akan dilakukan proses scaling resolusi dan pengaturan ukuran kualitas gambar. Gambar akan disimpan pada memorystream atau inputstream. Sebelum gambar dikirim, gambar diubah menjadi data Base64 dan disisipkan alamat ip perangkat *client* dan token. Token yang didapatkan dari aplikasi Raspberry pi bertujuan untuk meningkatkan sekuritas dari sistem yang telah dibuat. Lalu, paket gambar akan dikirimkan melalui protokol UDP. Sehingga komunikasi aplikasi Raspberry Pi dan aplikasi *client* saat mengirim gambar menggunakan protokol UDP. Selain mengirim gambar, aplikasi *client* juga mengirimkan data berupa username, password, token yang telah *hash*, dan perintah berupa status menggunakan protocol TCP.

Aplikasi penerima gambar/aplikasi Raspberry Pi memerlukan fitur *hotspot* agar aplikasi *client* terhubung dengan aplikasi Raspberry Pi. Aplikasi ini memerlukan *database* untuk menyimpan username dan password. Selain itu, aplikasi

berperan sebagai server untuk protokol TCP dan UDP untuk komunikasi datanya.

Sistem berjalan hanya saat terdapat 1 aplikasi Raspberry Pi dan 1 aplikasi *client* yang berjalan. Jika terdapat lebih dari 1 aplikasi *client*, maka hanya aplikasi *client* yang terhubung pertama kali dengan aplikasi Raspberry Pi yang akan dijalankan. Aplikasi perangkat lain yang berusaha untuk membuat komunikasi dengan kondisi seperti ini akan tidak dipedulikan.

Berdasarkan analisa diatas didapatkan kebutuhan fungsional dan kebutuhan non fungsional sebagai berikut:

Tabel 4. 1 Kebutuhan Fungsional (Raspberry Pi)

No	Raspberry Pi
1	Melakukan query menggunakan terminal pada databse
2	Membuat wifi hotspot dan membuatkan alamat ip kepada tiap *client*
3	Menampilkan gambar pada output display yang terhubung dengan Raspberry Pi
4	Menerima dan mengirim data pada socket
5	Menggunakan SSH

Tabel 4. 2 Kebutuhan Non-Fungsional (Raspberry Pi)

No	Raspberry Pi
1	Dapat memastikan bahwa pertukaran informasi atau data yang terjadi harus terlindung dari akses yang tidak berwenang
2	Dapat memastikan hanya satu user saja yang dapat menggunakan fitur

Tabel 4. 3 Kebutuhan Fungsional (Client)

No	Aplikasi *Client*
1	Komunikasi data perintah menggunakan TCP
2	Mendeteksi resolusi layar

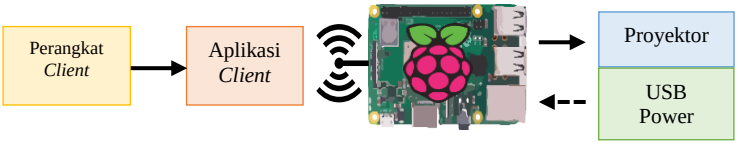
3	Mengambil gambar pada layar
4	Mengatur Resolusi dan kualitas gambar
5	Mengirim gambar

Tabel 4. 4 Kebutuhan Non-Fungsional (Client)

No	Aplikasi *Client*
1	Memiliki tampilan yang mudah dipahami
2	Aplikasi dapat memberikan informasi terhadap error yang terjadi

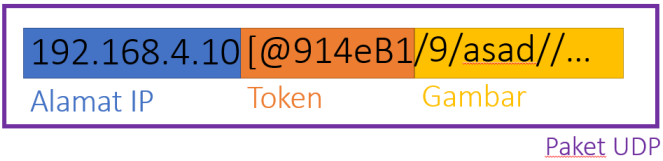
4.2 Tahap Desain Sistem

Desain sistem dirancang untuk menjadi acuan dari pengembangan sistem wireless display yang akan dibuat. Desain sistem yang akan dibuat menjelaskan bagaimana alur sebuah data diambil hingga menghasilkan keluaran yang diinginkan. Sehingga keluaran pada tahap ini adalah hasil analisa berupa alur dari sistem.



Gambar 4. 1 Perancangan Desain Sistem Wireless Display

Gambar 4. 1 menjelaskan alur data dari perangkat *client* hingga proyektor. Tahapan pertama yaitu pada perangkat *client*, data berupa gambar diambil tiap rentang *millisecond*. Kemudian pada aplikasi *client* yang terinstall pada perangkat *client* akan memproses gambar tersebut agar resolusi dan kualitas gambar yang dihasilkan optimal. Gambar yang sudah dilakukan pengaturan resolusi dan kualitas akan dijadikan dalam format base64 dan ditambahkan data mengenai alamat ip dan *hash* dari token.



Gambar 4. 2 Detail Paket UDP

Lalu, aplikasi *client* akan mengirimkan pesan dan gambar menuju Raspberry Pi menggunakan TCP dan UDP melalui koneksi Wi-Fi. Raspberry Pi akan menerima data gambar tersebut dan akan menampilkannya pada proyektor yang terhubung pada Raspberry Pi.

4.3 Tahap Desain Aplikasi

Desain aplikasi dirancang untuk menjadi acuan dari pengembangan aplikasi yang akan dibuat. Desain sistem aplikasi yang dibuat untuk aplikasi perangkat *client* dan aplikasi Raspberry Pi.

4.3.1 Desain Aplikasi Raspberry Pi

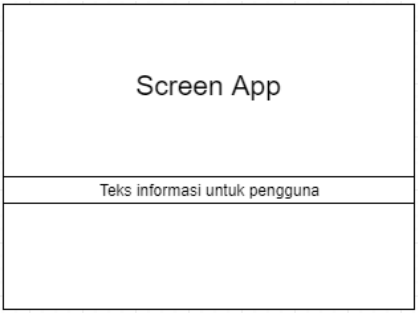
Desain aplikasi perangkat Raspberry Pi dirancang untuk menjadi acuan dalam pembuatan aplikasi pada Raspberry Pi. Desain ini dibuat untuk aplikasi dan juga aplikasi eksternal yang menunjang kebutuhan fungsional dari aplikasi Raspberry Pi.

Berdasarkan kebutuhan funsional dari tabel 4.1. Aplikasi Raspberry Pi memiliki 5 kebutuhan fungsional. Berikut penjelasan masing-masing kebutuhan fungsional:

Tabel 4. 5 Penjelasan Kebutuhan Fungsional (Raspberry Pi)

No	Kebutuhan Fungsional	Penjelasan
1	Melakukan query menggunakan terminal pada databse	Raspberry Pi membutuhkan database yang diinstall. Database hanya membutuhkan 1 tabel saja. Tabel yang

		dibutuhkan adalah tabel untuk menyimpan username dan password dari pengguna.
2	Membuat wifi hotspot dan membuatkan alamat ip kepada tiap *client*	Raspberry Pi membutuhkan aplikasi yang berfungsi sebagai hotspot yang diinstall. Serta Raspberry Pi perlu aplikasi untuk membuatkan alamat ip untuk setiap perangkat pengguna yang terhubung.
3	Menampilkan gambar pada output display yang terhubung dengan Raspberry Pi	Aplikasi raspberry pi memerlukan frame yang dapat menampilkan gambar sesuai data yang diterima.
4	Menerima dan mengirim data pada socket	Aplikasi Raspberry Pi memerlukan bahasa pemrograman yang memempunyai fitur socket dan dapat bertugas sebagai TCP dan UDP Server.
5	Menggunakan SSH	Pengaturan untuk dapat menggunakan port 22 telah diatur. Beserta username dan password yang telah diubah.



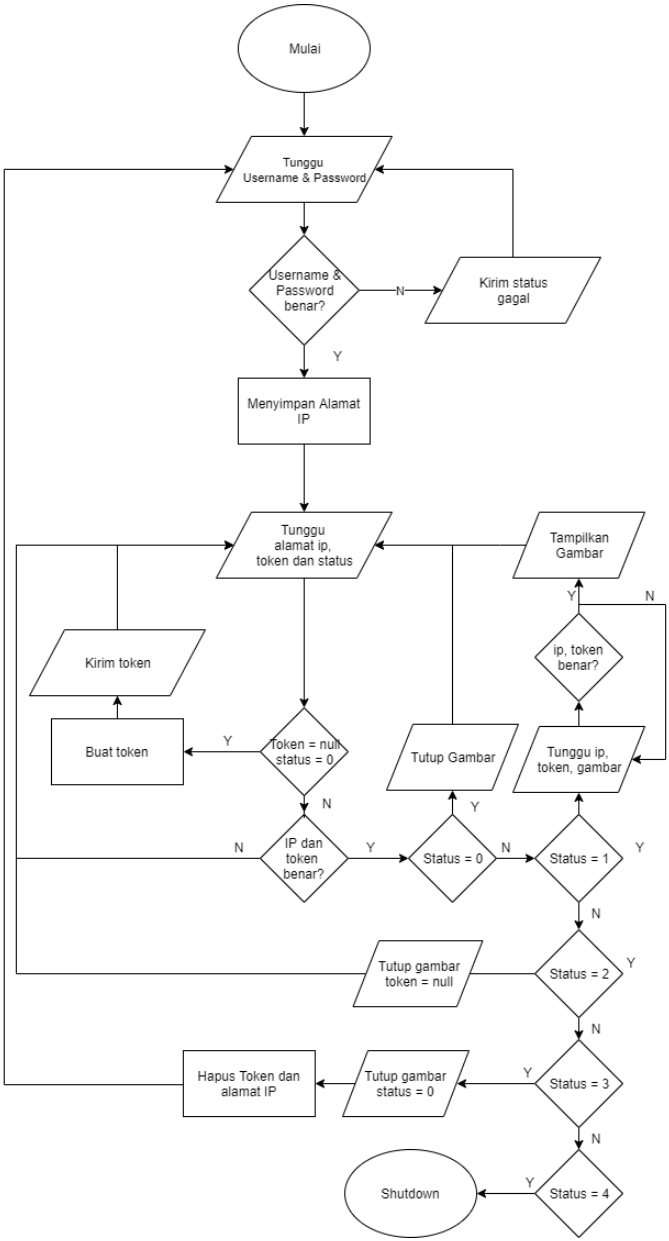
Gambar 4. 3 Rancangan Tampilan (Raspberry Pi)

Gambar 4. 3 merupakan tampilan dari aplikasi Raspberry Pi. Aplikasi Raspberry Pi tidak memiliki interaksi dengan pengguna. Aplikasi ini hanya menyediakan informasi status kepada pengguna. Status yang akan ditampilkan adalah saat aplikasi menunggu pengguna untuk masuk dan saat menunggu perintah dari pengguna. Tabel 4.6 adalah tabel status yang akan ditampilkan untuk pengguna.

Tabel 4. 6 Rancangan Tampilan Status (Raspberry Pi)

No	Kondisi	Teks informasi untuk pengguna
1	Menunggu Pengguna	"Waiting for user"
2	Menunggu Perintah	"Waiting for command …"

Gambar 4. 4 menjelaskan alur dari aplikasi Raspberry Pi secara keseluruhan. Dimulai dengan menunggu masukan username dan password dari pengguna. Saat username dan password benar, aplikasi akan menyimpan alamat ip dari pengirim username dan password. Lalu, aplikasi akan menunggu perintah berikutnya beserta mengganti tampilan dari teks untuk informasi pengguna. Aplikasi dalam menunggu perintah dari pengguna, aplikasi selalu mengecek apakah alamat ip dan token dari pengirim perintah sama seperti pengirim username dan password.



Gambar 4. 4 Diagram Alur Aplikasi Raspberry Pi

4.3.2.1 Perancangan Database

Perancangan database pada aplikasi Raspberry Pi dibuat menggunakan database sqlite. Dengan ketentuan tabel seperti pada tabel 4. 7.

Tabel 4. 7 Rancangan Detail Tabel Database

Nama Kolom	Tipe Data	Panjang
id	Primary key, AUTO_INCREMENT	-
username	Varchar	24
password	Varchar	24

Database yang digunakan oleh aplikasi Raspberry Pi cukup memerlukan 1 database saja. Serta database hanya memiliki 1 tabel saja. Dimana tabel berisikan 3 kolom, yaitu id, username dan password.

4.3.2.2 Perancangan Hotpspot

Hotspot pada Raspberry Pi memerlukan aplikasi tambahan yang perlu diinstall. Aplikasi untuk membuat hotspot dan aplikasi untuk menyediakan alamat IP dengan cara DHCP. Ketentuan jaringan seperti pada Tabel 4. 8.

Tabel 4. 8 Rancangan Detail Jaringan Raspberry Pi

Network Gateway	Subnet Mask	IP *Client* Range
192.168.4.1	255.255.255.0	192.168.4.2 – 192.168.4.255

Tabel 4.8 menjelaskan jaringan hotspot Raspberry Pi yang akan dibuat. Jaringan Raspberry Pi menggunakan DHCP untuk membagikan alamat IP dengan gateway 192.168.4.1 dengan subnet 255.255.255.0. Hal ini membuat perangkat *client* yang terhubung dengan jaringan Raspberry Pi akan mendapatkan alamat ip diantara 192.168.4.2 hingga 192.168.4.255.

4.3.2.3 Perancangan SSH

Pengaturan SSH perlu dilakukan pada Raspberry Pi. Hal ini dikarenakan untuk mengecek kondisi apakah SSH dari Raspberry Pi sudah dalan kondisi nyala atau mati. Serta, password default perlu dirubah karena masalah keamanan agar tidak semua orang dapat masuk ke dalam Raspberry Pi melalui SSH.

Tabel 4. 9 Rancangan User SSH Raspberry Pi

Username	Password
pi	Raspberry Pi

Tabel 4. 9 menjelaskan username dan password dari SSH Raspberry Pi. Username default dari Raspberry Pi adalah ‘pi’. Sedangkan password default dari Raspberry Pi adalah ‘Raspberry Pi’.

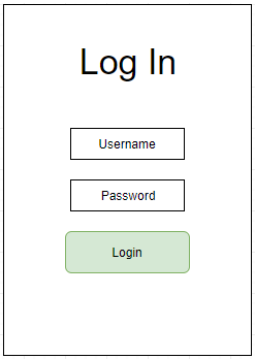
4.3.2.4 Perancangan Hash Token

Hash token menggunakan metode SHA512. Token dilakukan hash saat akan dikirim oleh aplikasi raspberry pi menuju aplikasi *client. Hash* dari token juga akan dilakukan oleh aplikasi client dan disimpan. Hash dari token akan dikirimkan bersama paket UDP (paket gambar) oleh aplikasi client saat pengguna menekan tolmbol play. Hash yang diterima oleh aplikasi raspberry pi akan dicocokkan dengan hash token yang pertama.

4.3.2 Desain Aplikasi Perangkat *Client*

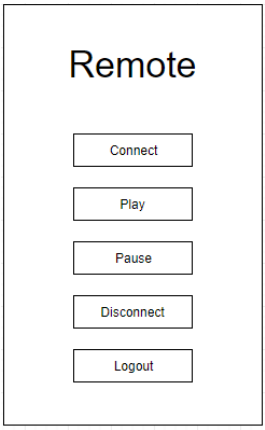
Desain aplikasi perangkat *client* dirancang untuk tiga sistem operasi, Android, Windows dan iOS. Semua aplikasi yang akan dibuat mempunyai alur yang sama. Perangkat *client* mampu mengirim dan menerima pesan menggunakan socket dan menangkap gambar pada layar lalu menjadikannya satu paket UDP dengan alamat ip dan token. Aplikasi perangkat hanya

memiliki 2 tampilan saja, yaitu halaman login dan halaman remote.



Gambar 4. 5 Rancangan Tampilan Login Client

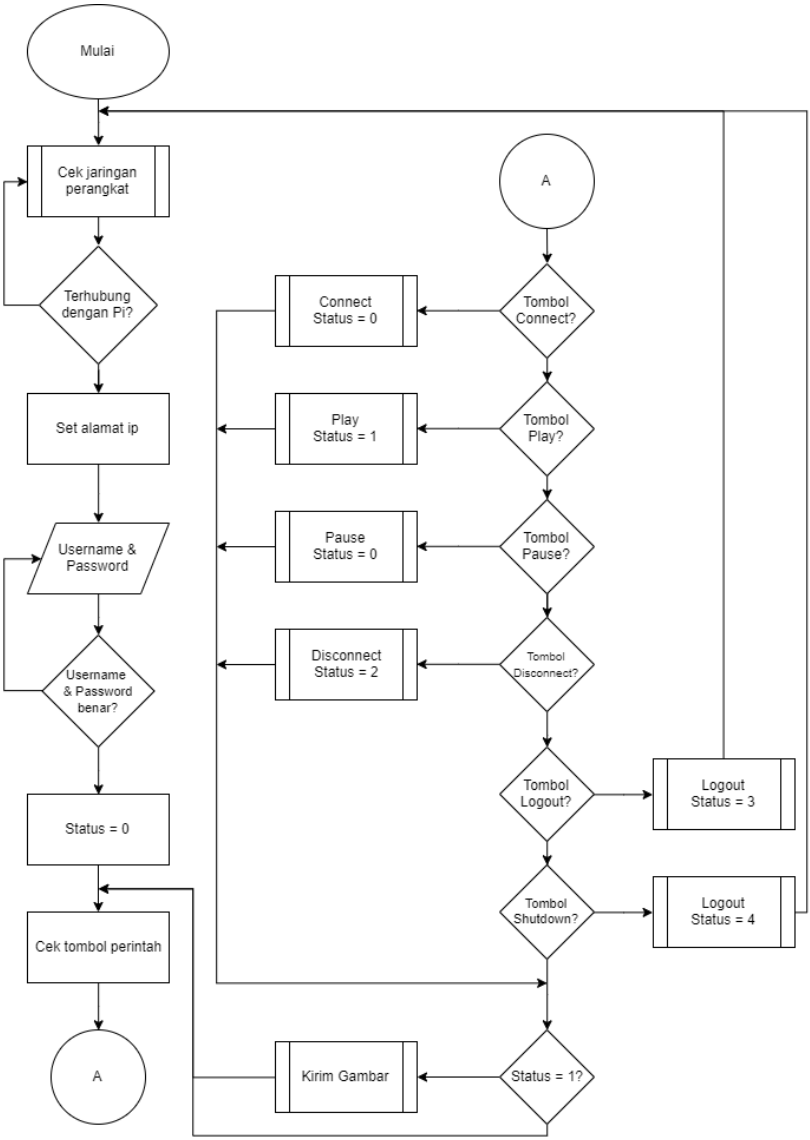
Gambar 4. 4 merupakan tampilan dari halaman login dari perangkat *client*. Tampilan ini merupakan tampilan yang pertama kali muncul ketika aplikasi *client* dibuka. Halaman login memiliki input username berupa teks dan input password berupa teks dengan tipe password (teks diganti dengan titik atau symbol bintang). Pada tampilan ini sistem akan mengecek jaringan perangkat sudah terhubung dengan wifi Raspberry Pi. Jika username dan password yang dimasukkan benar. Maka halaman aplikasi perangkat akan menuju halaman berikutnya.



Gambar 4. 6 Rancangan Tampilan Remore Client

Gambar 4. 6 merupakan halaman remote dari aplikasi perangkat *client*. Halaman ini hanya dapat dibuka saat user berhasil melakukan login saja. Saat user baru masuk halaman ini, user harus melakukan connect terlebih dahulu. Tombol connect ini bertujuan untuk meminta token kepada Raspberry Pi. Tombol play bertujuan untuk mengirim ip, token dan status kepada aplikasi Raspberry Pi. Serta tombol play ini bertujuan untuk menangkap gambar. Lalu, tombol pause bertujuan untuk memerintahkan aplikasi Raspberry Pi untuk menutup penampil gambar. Tombol disconnect bertujuan untuk mengatur kembali nilai token dan mematikan screenshot jika sedang berjalan. Tombol terakhir adalah tombol logout yang bertujuan untuk mematikan screenshot jika sedang berjalan, mengatur kembali nilai token, mengatur agar tampilan aplikasi kembali menjadi dan memerintahkan aplikasi Raspberry Pi untuk menunggu input username dan password baru.

Gambar 4. 7 menjelaskan alur dari sistem pada aplikasi *client* secara keseluruhan. Dimulai dari pengecekan jaringan pada perangkat *client* agar pengguna dapat langsung memulai menjalankan sistem pada Raspberry Pi. Jika tidak benar, maka akan meminta pengguna untuk terhubung dengan jaringan Raspberry Pi. Terdapat 6 perintah utama, yaitu Connect, Play, Pause, Disconnect, Logout, dan Shutdown. Lalu diakhiri dengan perintah logout yang akan meminta pengguna untuk memasukkan username dan password lagi.

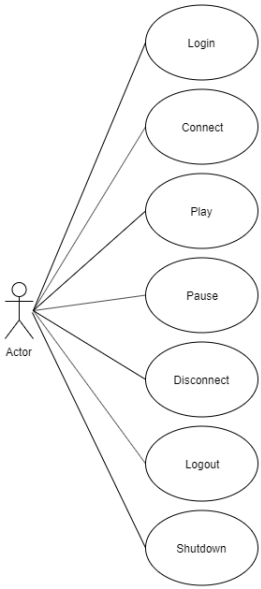


Gambar 4. 7 Gambar Diagram Alur Client

4.3.3 Perancangan UML Aplikasi Raspberry Pi

4.3.3.1 Perancangan Use Case Aplikasi Raspberry Pi

Perancangan *use case* bertujuan untuk menunjukkan interaksi antara aktor dan aplikasi Raspberry Pi, seperti pada gambar 4. 8.



Gambar 4. 8 Diagram Use Case Aplikasi Raspberry Pi

Use case yang dibuat adalah desain interaksi antara aplikasi perangkat pengguna dengan aplikasi Raspberry Pi. Aplikasi pada pre-condition telah menerima input dari pengguna.

Selain membuat diagram use case, juga dilakukan pembuatan scenario use case. Scenario use case yang dibuat sesuai dengan banyaknya use case pada gambar 4. 8 yaitu 7. Adapun use case scenario yang dibuat, seperti pada Tabel 4. 10 hingga Tabel 4. 16.

Tabel 4. 10 Use Case Scenario Login

Use Case	**Login**
Actor	Aplikasi Perangkat Pengguna.
Pre-Condition	Aktor telah mempunyai data berupa username dan password.
Basic Course	1. Aktor mengirimkan username dan password. 2. Sistem memeriksa username pada database. 3. Sistem memeriksa kecocokan username dengan password. 4. Sistem mengirimkan pesan bahwa username dan password ditemukan/tersedia.
Alternate Course	2a. Sistem memberikan pesan gagal. 3a. Sistem memberikan pesan gagal.
Post-Condition	Aktor telah masuk dan dapat menggunakan fitur aplikasi.

Tabel 4. 11 Use Case Scenario Connect (Raspberry Pi)

Use Case	**Connect**
Actor	Aplikasi Perangkat Pengguna.
Pre-Condition	Aktor berhasil login.
Basic Course	1. Aktor mengirimkan pesan untuk meminta token. 2. Sistem memeriksa pesan. 3. Sistem membuat token dan hash token menggunakan SHA512. 4. Sistem mengirim enkripsi token.
Alternate Course	-
Post-Condition	Aktor telah mendapatkan token dan sistem menunggu perintah berikutnya.

Tabel 4. 12 Use Case Scenario Play (Raspberry Pi)

Use Case	**Play**
Actor	Aplikasi Perangkat Pengguna.
Pre-Condition	Aktor telah mendapatkan token.
Basic Course	1. Aktor mengirimkan pesan play. 2. Sistem memeriksa kecocokan alamat ip. 3. Sistem memeriksa kecocokan token. 4. Sistem memeriksa pesan. 5. Sistem membuka tampilan gambar. 6. Sistem memeriksa ip dan token yang terdapat pada gambar 7. Sistem menampilkan gambar
Alternate Course	-
Post-Condition	Frame terbuka dan sistem menunggu perintah berikutnya.

Tabel 4. 13 Use Case Scenario Pause (Raspberry Pi)

Use Case	**Pause**
Actor	Aplikasi Perangkat Pengguna.
Pre-Condition	Aktor telah mendapatkan token.
Basic Course	1. Aktor mengirimkan pesan pause. 2. Sistem memeriksa pesan. 3. Sistem menutup tampilan gambar.
Alternate Course	-
Post-Condition	Frame tertutup dan sistem menunggu perintah berikutnya.

Tabel 4. 14 Use Case Scenario Disconnect (Raspberry Pi)

Use Case	**Disconnect**
Actor	Aplikasi Perangkat Pengguna.
Pre-Condition	Aktor telah mendapatkan token.
Basic Course	1. Aktor mengirimkan pesan disconnect. 2. Sistem memerika apakah terdapat token 3. Sistem memeriksa kecocokan alamat ip jika terdapat token. 4. Sistem memeriksa kecocokan token jika terdapat token. 5. Sistem memeriksa pesan jika terdapat token. 6. Sistem menutup tampilan gambar jika terbuka dan menghapus nilai token pada sistem.
Alternate Course	-
Post-Condition	Aktor tidak mempunyai nilai token dan sistem menunggu perintah berikutnya.

Tabel 4. 15 Use Case Scenario Logout (Raspberry Pi)

Use Case	**Logout**
Actor	Aplikasi Perangkat Pengguna.
Pre-Condition	Aktor telah berada pada tampilan remote.
Basic Course	1. Aktor mengirimkan pesan logout. 2. Sistem memerika apakah terdapat token 3. Sistem memeriksa kecocokan alamat ip jika terdapat token. 4. Sistem memeriksa kecocokan token jika terdapat token.

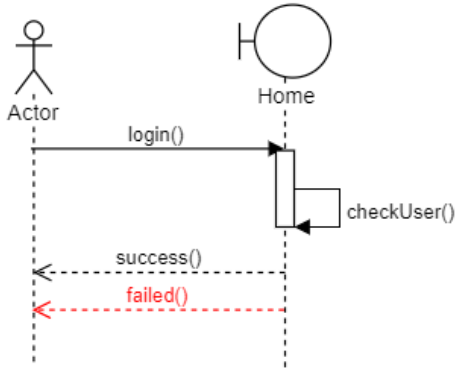
	5. Sistem memeriksa pesan jika terdapat token. 6. Sistem menutup tampilan gambar jika terbuka dan menghapus nilai token pada sistem.
Alternate Course	-
Post-Condition	Aktor telah keluar dari sistem dan sistem menunggu aktor baru yang login.

Tabel 4. 16 Use Case Scenario Shutdown (Raspberry Pi)

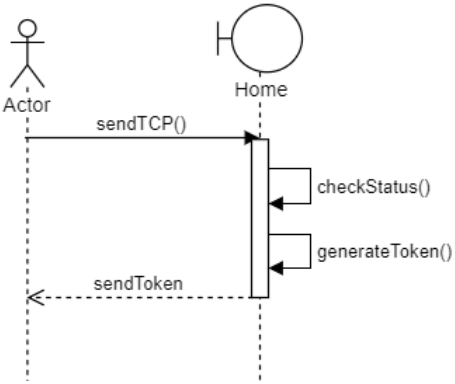
Use Case	**Shutdown**
Actor	Aplikasi Perangkat Pengguna.
Pre-Condition	Aktor telah berada pada tampilan remote.
Basic Course	1. Aktor mengirimkan pesan disconnect. 2. Sistem memerika apakah terdapat token 3. Sistem memeriksa kecocokan alamat ip jika terdapat token. 4. Sistem memeriksa kecocokan token jika terdapat token. 5. Sistem memeriksa pesan jika terdapat token. 6. Sistem menutup tampilan gambar jika terbuka dan menghapus nilai token pada sistem. 7. Sistem melakukan shutdown.
Alternate Course	-
Post-Condition	Aktor telah keluar dari sistem dan sistem menunggu aktor baru yang login.

4.3.3.2 Perancangan Sequence Diagram Aplikasi Raspberry

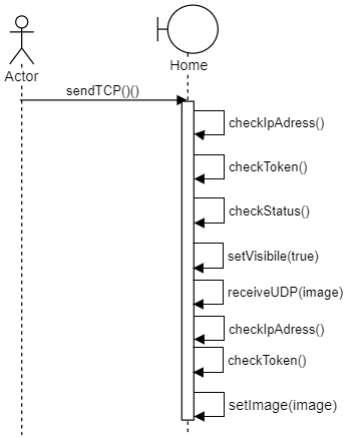
Sequence diagram dirancang berdasarkan usecase yang telah dibuat dengan berpedoman pada use case scenario pada poin 4.3.3.1.



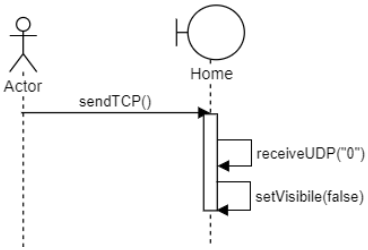
Gambar 4. 9 Sequence Diagram Login (Raspberry Pi)



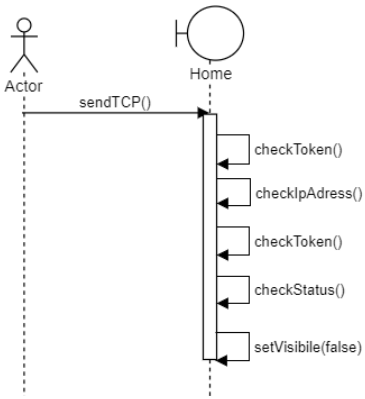
Gambar 4. 10 Sequence Diagram Connect (Raspberry Pi)



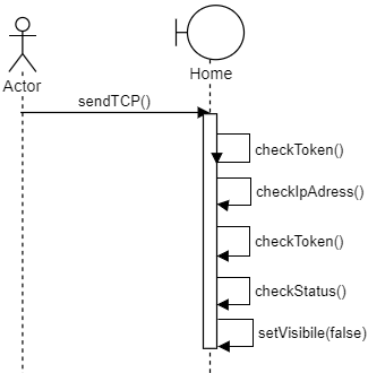
Gambar 4. 11 Sequence Diagram Play (Raspberry Pi)



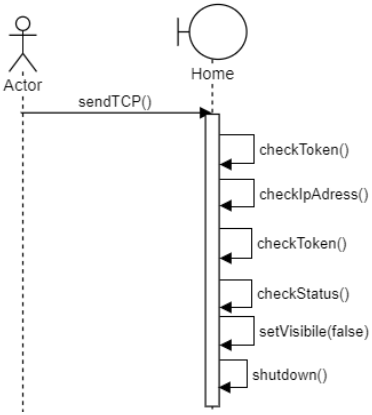
Gambar 4. 12 Sequence Diagram Pause (Raspberry Pi)



Gambar 4. 13 Sequence Diagram Disconnect (Raspberry Pi)



Gambar 4. 14 Sequence Diagram Logout (Raspberry Pi)



Gambar 4. 15 Sequence Diagram Shutdown (Raspberry Pi)

4.3.3.3 Perancangan Class Diagram Aplikasi Raspberry Pi

Aplikasi Raspberry Pi hanya memerlukan 1 class saja di dalam projectnya. 1 class ini mampu berjalan secara recursive untuk menunggu perintah dari pengguna. Berikut class diagram dari aplikasi yang mempunyai nama kelas Home.

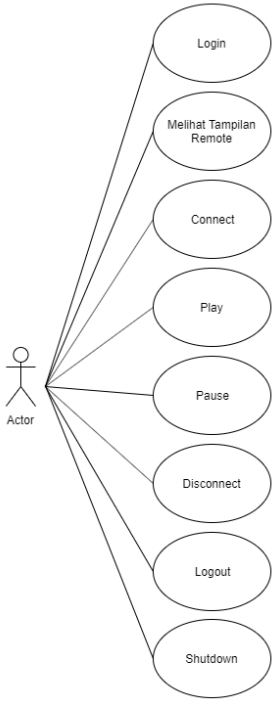
Class diagram dari aplikasi Raspberry Pi banyak bermain pada mengirim dan menerima data menggunakan TCP. Untuk membuka mematikan raspberry pi, aplikasi hanya memanggil menggunakan terminal saja.

Gambar 4. 16 Class Diagram Aplikasi Raspberry Pi

4.3.4 Perancangan UML Aplikasi *Client*

4.3.4.1 Perancangan Use Case Aplikasi *Client*

Perancangan *use case* dibuat dari hasil analisa kebutuhan fungsional pada tabel 4.1 dan tabel 4.3. *Use case* bertujuan untuk menunjukkan interaksi antara aktor dan sistem, seperti pada gambar 4.17.



Gambar 4. 17 Diagram Use Case Aplikasi Client

Use case yang dibuat hanya desain interaksi antara pengguna dengan aplikasi *client*. Pengguna harus melakukan login terlebih dahulu untuk dapat melihat tampilan remote dan mengirim perintah.

Selain membuat diagram use case, juga dilakukan pembuatan scenario use case. Scenario use case yang dibuat sesuai dengan banyaknya use case pada gambar 4.17 yaitu 7. Adapun use case

scenario yang dibuat, seperti pada Tabel 4.17 hingga Tabel 4.24.

Tabel 4. 17 Use Case Scenario Login (Pengguna)

Use Case	**Login**
Actor	Pengguna
Pre-Condition	Aktor membuka aplikasi
Basic Course	1. Aktor masuk ke halaman 'Login' 2. Sistem menampilkan halaman 'Login'. 3. Aktor menginput username dan password. 4. Aktor menekan tombol "Log In". 5. Sistem mengirimkan username dan password kepada Raspberry Pi. 6. Sistem menerima status sukses login 7. Sistem menuju ke halaman 'Remote'
Alternate Course	6a. Jika status yang diterima adalah gagal login, maka sistem akan mengarahkan pengguna kembali pada halaman login.
Post-Condition	Aktor telah masuk dan dapat menggunakan fitur aplikasi.

Tabel 4. 18 Use Case Scenario Melihat Tampilan Remote

Use Case	**Melihat Tampilan Remote**
Actor	Pengguna
Pre-Condition	Aktor berhasil login
Basic Course	1. Aktor masuk ke halaman 'Remote'. 2. Sistem menampilkan halaman 'Remote'. 3. Sistem memeriksa session pengguna. 4. Sistem menunggu paket TCP panjang resolusi raspberry pi 5. Sistem menunggu paket TCP lebar resolusi raspberry pi

	6. Sistem melakukan pengaturan skala resolusi berdasarkan panjang dan lebar raspberry pi yang telah didapat
Alternate Course	3a. Jika session pengguna tidak valid. Maka sistem akan menampilkan tampilan login.
Post-Condition	Aktor telah masuk dan dapat menggunakan fitur aplikasi

Tabel 4. 19 Use Case Scenario Connect (Pengguna)

Use Case	**Connect**
Actor	Pengguna
Pre-Condition	Aktor masuk pada halaman remote
Basic Course	1. Sistem menampilkan halaman 'Remote'. 2. Aktor menekan tombol Connect. 3. Sistem mengirimkan perintah kepada Raspberry Pi. 4. Sistem mendapatkan token dan melakukan hash dengan menggunakan metode SHA512. 5. Sistem memeriksa token.
Alternate Course	-
Post-Condition	Aktor telah mendapatkan token.

Tabel 4. 20 Use Case Scenario Play (Pengguna)

Use Case	**Play**
Actor	Pengguna
Pre-Condition	Aktor telah mendapatkan token
Basic Course	1. Aktor menekan tombol play. 2. Sistem menonaktifkan tombol play.

	3. Sistem mengirimkan paket TCP berupa alamat ip perangkat 4. Sistem mengirimkan paket TCP berupa token 5. Sistem mengirimkan paket TCP berupa status perintah 6. Sistem meminta izin untuk melakukan screenshot pada layar 7. Sistem melakukan screenshot 8. Sistem melukakan kompresi gambar 9. Sistem mengirim gambar kepada raspberry pi menggunakan protokol UDP 10. Sistem memeriksa token.
Alternate Course	-
Post-Condition	Aktor dapat melihat tampilan layar perangkat pada layar Raspberry Pi.

Tabel 4. 21 Use Case Scenario Pause (Pengguna)

Use Case	**Pause**
Actor	Pengguna
Pre-Condition	Aktor telah menekan tombol play.
Basic Course	1. Aktor menekan tombol pause. 2. Sistem menonaktifkan tombol pause 3. Sistem mengirimkan perintah kepada Raspberry Pi menggunakan protokol UDP. 4. Sistem berhenti melakukan screenshot 5. Sistem memeriksa token.
Alternate Course	-
Post-Condition	Aktor dapat melihat tampilan layar perangkat pada layar Raspberry Pi sudah berhenti.

Tabel 4. 22 Use Case Scenario Disconnect (Pengguna)

Use Case	**Disconnect**
Actor	Pengguna
Pre-Condition	Aktor telah menekan tombol connect.
Basic Course	1. Aktor menekan tombol disconnect. 2. Sistem memeriksa token 3. Sistem memeriksa apakah sedang melakukan screenshot 4. Sistem mengirimkan paket TCP berupa alamat ip perangkat jika terdapat token 5. Sistem mengirimkan paket TCP berupa token jika terdapat token 6. Sistem mengirimkan paket TCP berupa status perintah 7. Sistem memeriksa token 8. Sistem mengatur token menjadi null
Alternate Course	-
Post-Condition	Aktor tidak mempunyai token.

Tabel 4. 23 Use Case Scenario Logout (Pengguna)

Use Case	**Logout**
Actor	Pengguna
Pre-Condition	Aktor masuk pada halaman remote
Basic Course	1. Sistem menampilkan halaman ‘Remote’. 2. Aktor menekan tombol Logout. 3. Sistem memeriksa token 4. Sistem memeriksa apakah sedang melakukan screenshot

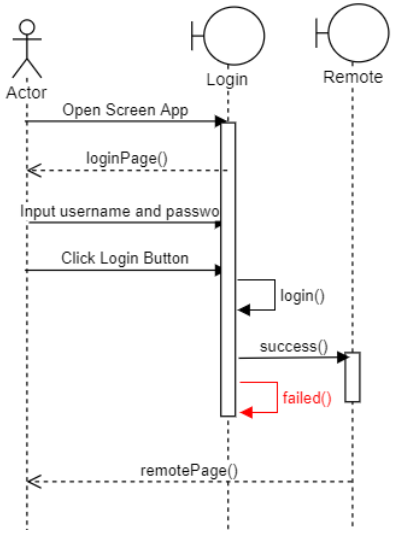
	5. Sistem mengirimkan paket TCP berupa alamat ip perangkat jika terdapat token 6. Sistem mengirimkan paket TCP berupa token jika terdapat token 7. Sistem mengirimkan paket TCP berupa status perintah 8. Sistem melakukan logut
Alternate Course	-
Post-Condition	Aktor telah keluar dari sistem.

Tabel 4. 24 Use Case Scenario Shutdown (Pengguna)

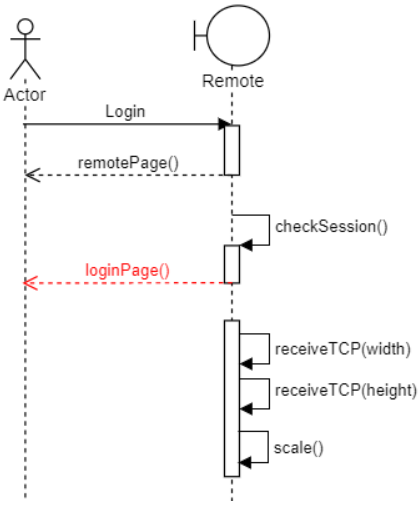
Use Case	**Shutdown**
Actor	Pengguna
Pre-Condition	Aktor masuk pada halaman remote
Basic Course	1. Sistem menampilkan halaman 'Remote'. 2. Aktor menekan tombol Shutdown. 3. Sistem memeriksa token 4. Sistem memeriksa apakah sedang melakukan screenshot 5. Sistem mengirimkan paket TCP berupa alamat ip perangkat jika terdapat token 6. Sistem mengirimkan paket TCP berupa token jika terdapat token 7. Sistem mengirimkan paket TCP berupa status perintah 8. Sistem melakukan shutdown
Alternate Course	-
Post-Condition	Aktor telah keluar dari sistem.

4.3.4.2 Perancangan Sequence Diagram Aplikasi *Client*

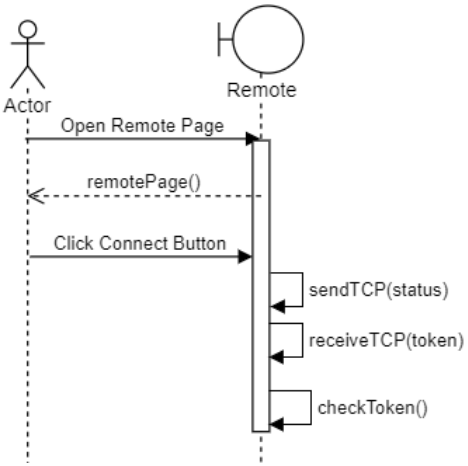
Sequence diagram dirancang berdasarkan usecase yang telah dibuat dengan berpedoman pada use case scenario pada poin 4.3.4.1.



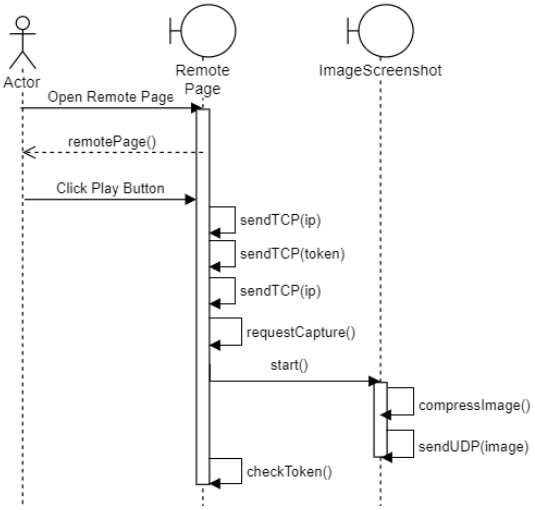
Gambar 4. 18 Sequence Diagram Login (Pengguna)



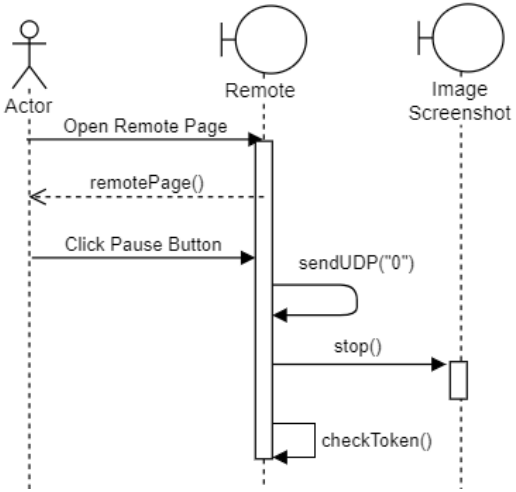
Gambar 4. 19 Sequence Diagram Melihat Tamplan Remote



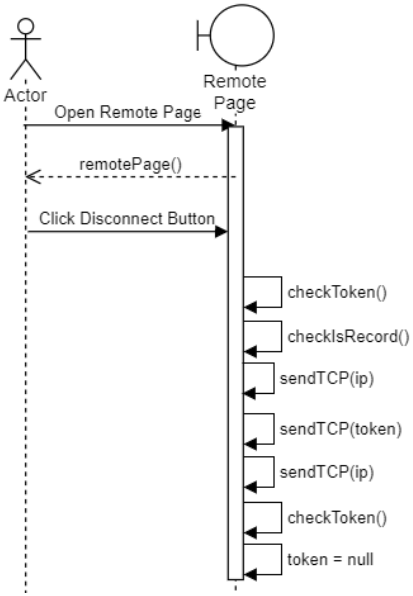
Gambar 4. 20 Sequence Diagram Connect (Pengguna)



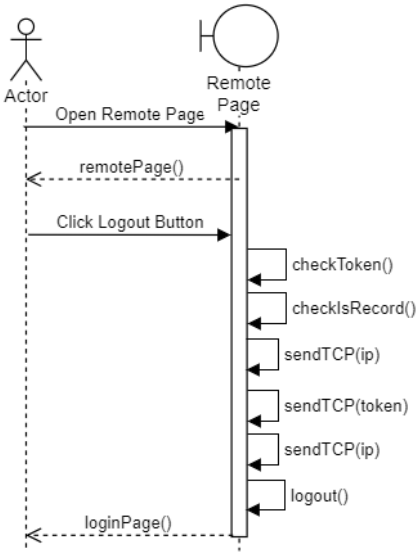
Gambar 4. 21 Sequence Diagram Play (Pengguna)



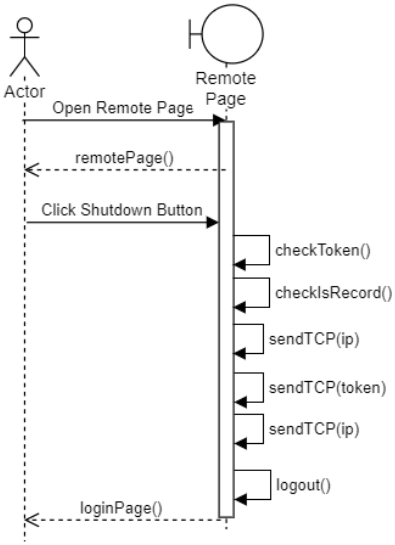
Gambar 4. 22 Sequence Diagram Pause (Pengguna)



Gambar 4. 23 Sequence Diagram Disconnect (Pengguna)



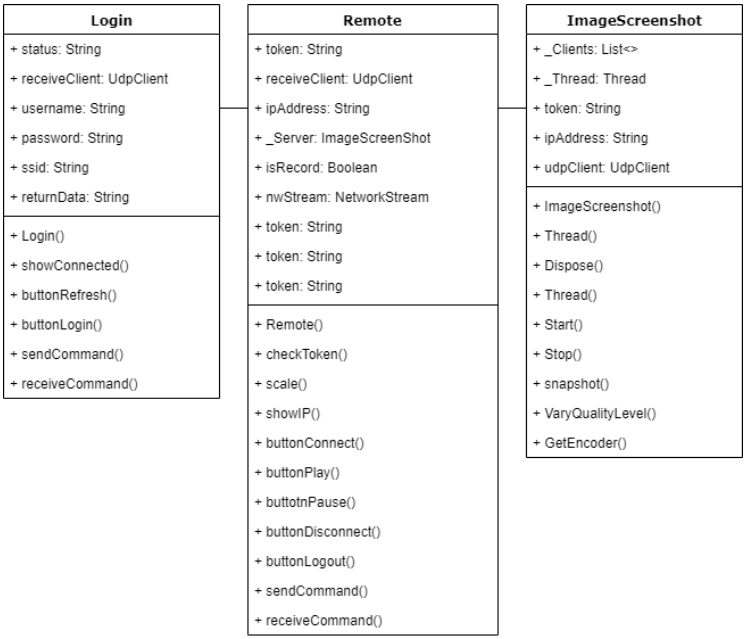
Gambar 4. 24 Sequence Diagram Logout (Pengguna)



Gambar 4. 25 Sequence Diagram Shutdown (Pengguna)

4.3.4.3 Perancangan Class Diagram Aplikasi *Client*

Class diagram perangkat aplikasi *client* terdiri dari 3. Hal ini dikarenakan tiap sistem operasi memiliki bahasa yang berbeda. Dalam tugas akhir ini penulis menggunakan bahasa pemrograman bawaan masing-masing sistem operasi.



Gambar 4. 26 Class Diagram Perangkat Client

Class diagram pada gambar 4.26 adalah class diagram untuk aplikasi windows. Pada halaman remote terdapat class yang menangani untuk melakukan screenshot dan menangani kualitas gambar.

BAB V
IMPLEMENTASI

Pada bagian ini akan menjelaskan mengenai proses implementasi terhadap sistem wireless display sesuai dengan *output* dari proses perancangan yang telah dibuat.

5.1 Lingkungan Implementasi

Pengembangan sistem wireless display menggunakan komputer dengan spesifikasi yang tertera pada Tabel 5.1 dan Tabel 5.3. Komputer 1 dibuat untuk membuat aplikasi pada Raspberry Pi. Komputer 2 dibuat untk membuat aplikasi pada Windows dan Android. Sedangkan komputer 3 dibuat untuk membuat aplikasi pada iOS.

Tabel 5. 1 Spesifikasi Komputer 1 Implementasi

Tipe	Rasberry Pi 3 Model B V1.2
Processor	Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
Memory (RAM)	1GB LPDDR2 (900 MHz)
Sistem Operasi	Raspbian
GPU	Broadcom VideoCore IV @ 250 MHz

Tabel 5. 2 Spesifikasi Komputer 2 Implementasi

Tipe	ASUS ROG GL553VD
Processor	Intel® Core™ i7-7700HQ CPU @2.8GHz
Memory (RAM)	16 GB
Sistem Operasi	Windows 10 Education 64Bit
Graphic Card	Nvidia GeForce GTX 1050

Tabel 5. 3 Spesifikasi Komputer 3 Implementasi

Tipe	MacBook Air Early 2015
Processor	1.6 GHz Dual-Core Intel Core i5
Memory (RAM)	4 GB 1600 MHz DDR3
Sistem Operasi	macOS Catalina 10.15.1
Graphic Card	Intel HD Graphic 6000 1536 MB

Sistem wireless display dikembangkan dengan menggunakan beberapa teknologi seperti IDE *(Integrated Development Environment)*, bahasa pemrograman dan database yang terdapat pada Tabel 5.4 hingga Tabel 5.7.

Tabel 5. 4 Teknologi Pengembangan Aplikasi Raspberry Pi

IDE	Netbeans
Bahasa Pemrograman	Java
Database	Sqlite3

Tabel 5. 5 Teknologi Pengembangan Aplikasi Windows

IDE	Microsoft Visual Studio Community 2017
Bahasa Pemrograman	C#
Database	-

Tabel 5. 6 Teknologi Pengembangan Aplikasi Android

IDE	Android Studio 3.5.2
Bahasa Pemrograman	Java
Database	-

Tabel 5. 7 Teknologi Pengembangan Aplikasi iOS

IDE	XCode 11.2.1
Bahasa Pemrograman	Swift 4
Database	-

5.2 Implementasi Sistem

Pengembangan *sistem wireless display* dimulai pada implementasi aplikasi terlebih dahulu kemudian implementasi sistem. Implementasi aplikasi dilakukan dengan menuliskan kode program yang digunakan. Sedangkan implementasi sistem dilakukan dengan menghubungkan aplikasi sesuai dengan rancangan sistem.

5.2.1 Implementasi Aplikasi Raspberry Pi

Aplikasi Raspberry Pi dibuat sebagai UDP Server dan menjadi TCP Server. Implementasi aplikasi memiliki beberapa bagian di dalamnya. Komponen – komponen yang akan diimplementasi antara lain adalah aplikasi Raspberry Pi itu sendiri, database, hotspot, dan SSH. Berikut adalah kode dan gambar yang akan menjelaskan implementasi dari aplikasi Raspberry Pi.

```
87  public Home() throws ClassNotFoundException, SQLException, InterruptedException,
88  AWTException {
89
90      initComponents();
91      Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
92      this.setLocation(dim.width / 2 - this.getSize().width / 2,
93      dim.height / 2 - this.getSize().height / 2);
94      try {
95          das = new DatagramSocket(9876);
96          das1 = new DatagramSocket(9875);
97          serverSocket = new ServerSocket(5000);
98      } catch (SocketException ex) {
99          Logger.getLogger(Home.class.getName()).log(Level.SEVERE, null, ex);
100     } catch (IOException ex) {
101         Logger.getLogger(Home.class.getName()).log(Level.SEVERE, null, ex);
102     }
103     textStatus.setHorizontalAlignment(JTextField.CENTER);
104     Class.forName("org.sqlite.JDBC");
105     conn = DriverManager.getConnection("jdbc:sqlite:/home/pi/user.db");
106     conn.setAutoCommit(false);
107     stmt = conn.createStatement();
108     System.out.println("sukses");
109     decoder = new BASE64Decoder();
110     imageSource = new Source();
111     f.setSize(1024,768);
112     f.setExtendedState(JFrame.MAXIMIZED_BOTH);
113     f.setUndecorated(true);
114     f.add(l);
115     initSelfListeners();
```

Kode 5. 1 Implementasi Class Home (Raspberry Pi)

Aplikasi pada Rasberry Pi ini menggunakan Java Swing. Penggunaa Java Swing pada tugas akhir ini karena penulis sudah pernah menggunakannya dan lebih mudah mengimplementasikannya saat menggunakan GUI (Graphical User Interface). Pada Kode 5.1 terdapat baris kode yang merupakan inisiasi untuk pembuatan Datagram Socket. Datagram socket pada Raspberry Pi berfungsi sebagai server yang menangani sistem. Server udp ini memiliki port 9876 dan 9875. Selain menginisiasi server udp, pada kode ini terdapat kode yang menginisiasi ServerSocket pada port 5000 yang akan digunakan oleh TCP Server. Serta, terdapat kode untuk membuat koneksi terhadap database sqlite yang bernama “user.db” dan menginisiasi tampilan gambar yang akan digunakan untuk menampilkan gambar.

```
SwingWorker<Integer, Integer> StartupLoader = new SwingWorker<Integer, Integer>() {
    @Override
    protected Integer doInBackground() throws Exception {
        System.out.println("check user");
        tcpSocket = serverSocket.accept();
        username = receiveCommand();
        password = receiveCommand();
        System.out.println(username + password);
        query = "SELECT count(*) FROM users WHERE username ='" + username + "'";
        ipClient = tcpSocket.getInetAddress();
        ipAddressClient = ipClient.toString().substring(1);
        System.out.println(ipAddressClient);
        port = tcpSocket.getPort();
        System.out.println(port);
        try (ResultSet rs = stmt.executeQuery(query)) {
            if (rs.next()) {
                boolean found = rs.getBoolean(1);
                if (found) {
                    query = "SELECT username, password from users where username='" + username
                    pst = conn.prepareStatement(query);
                    ResultSet rs1 = pst.executeQuery();
                    if (password.equals(rs1.getString("password"))) {
                        System.out.println("sini");
                        sendCommand("Sukses Login");
                        tcpSocket = serverSocket.accept();
                        Thread.sleep(2000);
                        //CHECK RESO
                        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
                        String widthScreen = Double.toString(screenSize.getWidth());
                        String heightScreen = Double.toString(screenSize.getHeight());
                        Thread.sleep(500);
                        sendCommand(widthScreen);
                        Thread.sleep(500);
                        sendCommand(heightScreen);
```

Kode 5. 2 Fungsi Raspberry Pi doInBackground 1

Kode 5.2 mengimplementasikan kode doInBackground dari SwingWorker. Penggunaan SwingWorker bertujuan untuk menghindari kerja program yang bersifat sequence / berurutan. Jika tidak menggunakan SwingWorker, aplikasi akan seolah-olah tidak berjalan dan tidak dapat memberikan informasi yang benar kepada pengguna.

Didalam penggunaan SwingWorker teradapat pengecekan akun pengguna dua kali. Pengecekan yang pertama adalah pengecekan apakah username yang dimasukkan oleh pengguna terdapat pada database. Lalu, pengecekan yang kedua adalah password masukkan pengguna apakah cocok dengan username yang dimasukkan. Jika pengecekan yang dilakukan tidak

mendapatkan hasil atau salah, aplikasi rasberry akan mengirimkan pesan “gagal login”.

Saat pengguna sukses melakukan login, raspberry pi akan mengirimkan panjang dan lebar resolusi raspbery pi.

```
155         while (remote) {
156             if (token == null) {
157                 status1 = receiveCommand();
158                 if (status1.equals("3") ||status1.equals("4")) {
159                     checkStatus(status1);
160                 } else {
161                     device = receiveCommand();
162                     checkStatus(status1);
163                 }
164             } else if (token != null || device.equals("IOS")) {
165                 try {
166                     do {
167                         //checkIP
168                         ipAddress = receiveCommand();
169                     } while (!ipAddress.equals(ipAddressClient));
170                     do{
171                         if(device.equals("IOS"))
172                             break;
173                         //checkToken
174                         tokenReceive = receiveCommand();
175                     }
176                     while (!tokenReceive.equals(token));
177                     String status = receiveCommand();
178                     checkStatus(status);
179                 } catch (IOException ex) {
180                     Logger.getLogger(Home.class.getName()).log(Level.SEVERE, null
181                 } catch (ClassNotFoundException ex) {
182                     Logger.getLogger(Home.class.getName()).log(Level.SEVERE, null
183                 }
184             } else {break;}
185         }
186     } else {sendCommand("gagal login");}
187 } else {sendCommand("gagal login");}
```

Kode 5. 3 Fungsi Raspberry Pi doInBackground 2

Pada Kode 5.3 terdapat pengecekan token apakah token memiliki nilai null atau tidak. Saat token bernilai null, raspberry pi hanya akan memeriksa perintah berupa status dan jenis perangkat pengguna. Saat token tidak bernilai null, raspberry pi akan memeriksa alamat ip, token dan status perintah dari perangkat pengguna.

```
do{
    String sdf=LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss.SSS'
    System.out.print(sdf);
    a++;
    System.out.println(" :"+a);
    das1.receive(DpReceive);
    String rawString = new String (receive,0,DpReceive.getLength());
    if(!rawString.equals("0")){
        ipAddress = rawString.substring(0,rawString.indexOf('['));
        if(ipAddress.equals(ipAddressClient)){
            tokenGambar = rawString.substring(rawString.indexOf('['), rawString.indexOf('/'));
            if(tokenGambar.equals(token)){
                text =  rawString.substring(rawString.indexOf('/'));
                DpReceive.setLength(receive.length);
                imageByte = decoder.decodeBuffer(text);
                text="";
                inputStream = new ByteArrayInputStream(imageByte);
                image = ImageIO.read(inputStream);
                imageIcon = new ImageIcon(image);
                l.setIcon(imageIcon);
                getImage = true;
            }
        }
    }else if(rawString.equals("0")){
        f.setVisible(false);
        getImage=false;
    }
```

Kode 5. 4 Fungsi Menampilkan Gambar

Kode 5.4 adalah kode untuk menerima gambar dan menampilkan gambar pada tampilan baru. Paket udp yang diterima dipisahkan menjadi 3 bagian. Bagian pertama adalah data berupa alamat ip pengirim gambar. Bagian kedia adalah data berupa token pengirim gambar. Bagian ketiga adalah data gambar berupa teks dengan Base64 encode. Gambar berupa teks ini diubah menjadi bentuk byte gambar lalu menampilkan pada tampilan baru yang telah dibuka.

Gambar berhenti ditampilkan saat paket UDP yang diterima adalah “0”. Saat paket ini diterima, tampilan gambar akan disembunyikan dan menghetinkan *loop* dari penerima gambar.

```
325   private static void checkStatus(String status) throws IOException, ClassNotFoundException {
326       if (status.equalsIgnoreCase("3")) {
327           f.setVisible(false);
328           if (isRecord) {
329               Runtime.getRuntime().exec("sudo reboot");
330           }
331           remote = false;
332           token = null;
333           tcpSocket.close();
334           System.out.println("keluar");
376       } else if (status.equalsIgnoreCase("1") && device.equalsIgnoreCase("IOS")) {
377           System.out.println(device);
378           isRecord = true;
379           Runtime.getRuntime().exec("rpiplay");
380       } else if (status.equalsIgnoreCase("0") && token == null) {
381           System.out.println("token");
382           //send token
383           byte[] tokenValue = new byte[65535];
384           SecureRandom random = new SecureRandom();
385           byte bytes[] = new byte[65535];
386           random.nextBytes(bytes);
387           token = bytes.toString();
388           try {
389               Thread.sleep(2000);
390           } catch (InterruptedException ex) {
391               Logger.getLogger(Home.class.getName()).log(Level.SEVERE, null, ex);
392           }
393           sendCommand(token);
394       } else if (status.equalsIgnoreCase("0") && token != null) {
395           f.setVisible(false);
396
397       } else if (status.equalsIgnoreCase("2")) {
398           f.setVisible(false);
399           token = null;
400       }else if (status.equalsIgnoreCase("4")) {
401           Runtime.getRuntime().exec("sudo shutdown");
402       }
403   }
418       public static String receiveCommand(){
419           String text="";
420           System.out.println("receive..");
421           try {
422               InputStream is = tcpSocket.getInputStream();
423               ByteArrayOutputStream baos = new ByteArrayOutputStream();
424               byte buffer[] = new byte[1024];
425               baos.write(buffer,0,is.read(buffer));
426               byte result[] = baos.toByteArray();
427               text = new String(result);
428               System.out.println("received: "+ text);
429   //
430           } catch (IOException ex) {
431               Logger.getLogger(Home.class.getName()).log(Level.SEVERE, null, ex);
432           }
433           return text;
434       }
435
436       public static void sendCommand(String command){
437           try {
438               System.out.println("send..");
439               byte send[] = command.getBytes();
440               OutputStream os = tcpSocket.getOutputStream();
441               os.write(send);
442           } catch (IOException ex) {
443               Logger.getLogger(Home.class.getName()).log(Level.SEVERE, null, ex);
444           }
445       }
```

```
430   static public String get_SHA_512_SecurePassword(String passwordToHash, String salt){
431       String generatedPassword = null;
432       try {
433           MessageDigest md = MessageDigest.getInstance("SHA-512");
434           md.update(salt.getBytes(StandardCharsets.UTF_8));
435           byte[] bytes = md.digest(passwordToHash.getBytes(StandardCharsets.UTF_8));
436           StringBuilder sb = new StringBuilder();
437           for(int i=0; i< bytes.length ;i++){
438               sb.append(Integer.toString((bytes[i] & 0xff) + 0x100, 16).substring(1));
439           }
440           generatedPassword = sb.toString();
441       } catch (NoSuchAlgorithmException e) {
442           e.printStackTrace();
443       }
444       return generatedPassword;
445   }
```

Kode 5. 5 Method Pendukung (Raspberry Pi)

Method pada Kode 5.5 bekerja berdasarkan status yang dikirimkan oleh perangkat aplikasi pengguna. Serta, terdapat kode untuk membuat token menggunakan class SecureRandoM dan menyimpan hash token menggunakan SHA512. Untuk menjalankan restart, shutdown dan menjalankan Rpi-Play raspberry pi menjalankan perintah layaknya menjalankan perintah pada terminal linux.

Perintah receiveCommand adalah untuk menerima paket TCP dari perangkat pengguna dan perintah sendCommand adalah untuk mengirim paket TCP ke perangkat pengguna. Lalu, untuk melakukan hash token menggunakan SHA512 digunakan metode get_SHA_512_SecurePassword. Hasil dari token ini akan digunakan untuk memeriksa kecocokan dari hash dari client.

5.2.1.1 Implementasi Database

Implementasi database pada aplikasi Raspberry Pi menggunakan sqlite. Pemasangan sqlite dapat menggunakan “sudo apt-get install sqlite3”. Lalu, pembuatan database bernama user dilakukan dengan memasukkan “sqlite3 user.db” seperti pada Gambar 5.1.

```
pi@raspberrypi: -$ sqlite3 user.db
```

Gambar 5. 1 Terminal Pembuatan Database

Lalu, pembuatan tabel baru pada database dilakukan dengan cara memasukkan “CREATE TABLE users(id integer PRIMARY KEY, username TEXT NOT NULL, password TEXT NOT NULL);” untuk tabel dengan nama users seperti pada Gambar 5.2.

```
pi@raspberrypi: -$ sqlite3 user.db
sqlite> CREATE TABLE users(id integer PRIMARY KEY,
username TEXT NOT NULL, password TEXT NOT NULL);
```

Gambar 5. 2 Terminal Pembuatan Tabel

Pada Gambar 5.3 penulis memasukkan data berupa sa untuk username dan sa123 untuk password. Query yang dimasukkan adalah “INSERT INTO users(username, password) VALUES (‘sa’, ‘sa123’);”

```
pi@raspberrypi: -$ sqlite3 user.db
sqlite> INSERT INTO users(username, password)
VALUES (‘sa’,’sa123’);
```

Gambar 5. 3 Terminal untuk Memasukkan Data

5.2.1.2 Implementasi Hotspot

Implementasi hotspot pada Raspberry Pi menggunakan hostapd dnsmasq. Prosesn instalasi dapat dilakukan dengan memasukkan “sudo apt-get install hostapd dnsmasq”.

Setelah proses instalasi selesai, edit file dnsmasq.conf pada path /etc/. Hapus komen pada file pada baris

- interface=lo,uap0
- no-dhcp-interface=lo,wlan0
- dhcp-range = 192.168.4.2, 192.168.4.255, 255.255.255.0, 24h

Selanjutnya, edit file hostapd.conf pada path /etc/hostapd. Ubah data seperti pada detail berikut:

- interface=uap0
- ssid=Pi3-AP

- hw_mode=g
- channel=6
- macaddr_acl=0
- auth_algs=1
- ignore_broadcast_ssid=0
- wpa=2
- wpa_passphrase=Raspberry Pi
- wpa_key_mgmt=WPA-PSK
- wpa_pairwise=TKIP
- rsn_pairwise=CCMP

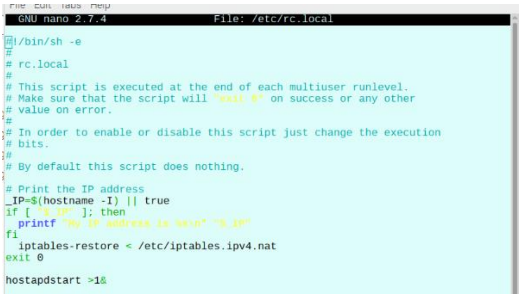
SSID dan password menggunakan “Pi3-AP” dengan password Raspberry Pi. Kemudian bah dan menambhakan pada konfigurasi /etc/network/interfaces menjadi

- auto uap0
- iface uap0 inet static
- address 192.168.4.1
- netmask 255.255.255.0

Selanjutnya, buat file baru dengan cara “sudo nano /usr/local/bin/hostapdstart dengan menambahkan:

- iw dev wlan0 interface add uap0 type __ap
- service dnsmasq restart
- sysctl net.ipv4.ip_forward=1
- iptables -t nat -A POSTROUTING -s 192.168.2.0/24 ! -d 192.168.2.0/24 -j MASQUERADE
- ifup uap0
- hostapd /etc/hostapd/hostapd.conf

Ubah izin pada /usr/local/bin/hostapdstart menggunakan “chmod 667 /usr/local/bin/hostapdstart”. Lalu ubah dan tambahkan baris ke /etc/rc.local dengan tulisan “hostapdstart >1&” pada baris terkahir.



Gambar 5. 4 Terminal Pengaturan rc.local

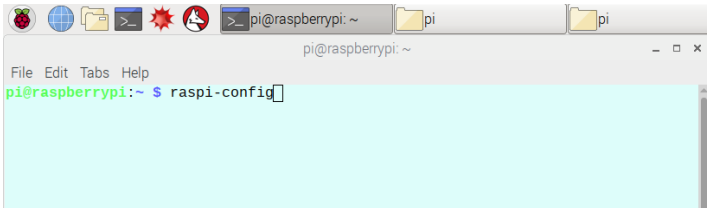
Terakhir tambahkan baris pada file dhcpcd.conf pada path /etc/:

- interface wlan0
- static ip_address = 192.168.4.1/24
- nohook wpa_supplicant

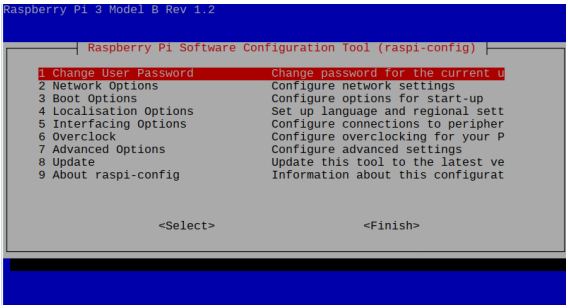
Untuk dapat menyalakan hostapd dan dnsmasq, masukkan "Sudo systemctl start hostapd", "Sudo sytemctl start dnsmasq" dan "Sudo service dhcpd restart"

5.2.1.3 Implementasi SSH

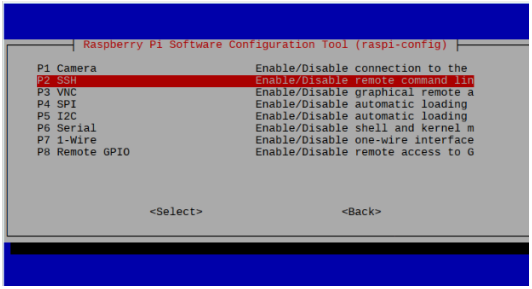
Penggunaan SSH digunakan untuk mengatur user pada database sqlite. Implementasi SSH dengan cara memasukkan "raspi-config" pada terminal. Dalam mengimplementasikan SSH, Raspberry Pi hanya perlu meminta masukkan dari pengguna. SSH dapat digunakan untuk remote dan juga dapat mengaktifkan fitur FTP.



Gambar 5. 5 Terminal raspi-config



Gambar 5. 6 Tampilan raspi-config



Gambar 5. 7 Tampilan Interfacing Options

Pada Gambar 5.10, untuk mengatur SSH pada Raspberry Pi terdapat pada menu nomor 4 Localisation Options. Lalu, pada Gambar 5.11 terdapat pilihan P2 SSH. Saat menu ditekan akan muncul peringatan apakah ingin menyalakan atau mematikan.

Untuk mengganti password SSH dapat dilakukan dengan masuk ke level root pada terminal dan memasukkan input berupa passwd. Sesuai dengan rancangan SSH, username tetap pi dan password diubah menjadi Raspberry Pi. Hal ini dengan tujuan agar tidak semua orang dapat mengakses SSH pada Raspberry Pi secara ilegal. Pengaturan ini dapat dilihat pada Gambar 5.12.

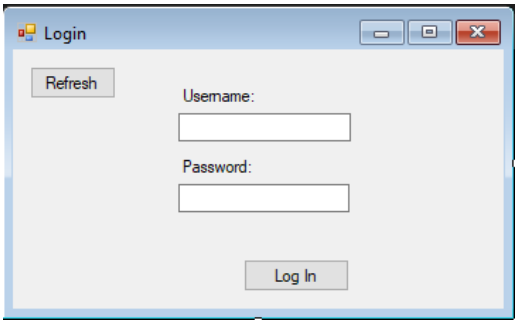
```
pi@raspberrypi: -$ sudo raspi-config
pi@raspberrypi: -$ sudo su
```

```
root@raspberrypi:/home/pi# passwd
```

Gambar 5. 8 Terminal Pengaturan Password Raspberry Pi

5.2.2 Implementasi Aplikasi Windows

Implementasi aplikasi windows menggunakan Visual Studio 2017 dengan menggunakan teknik pengambilan gambar pada layar (*screenshot*) tiap milidetiknya. Berikut hasil implementasi dari aplikasi windows.



Gambar 5. 9 Tampilan Login Aplikasi Windows

Tampilan Login pada aplikasi windows memiliki input berupa username dan password. Input berupa password tidak dapat dilihat dalam bentuk latin / disensor. Serta memiliki tombol refresh dan tombol login.

```
 9   public partial class Login : Form
10   {
11       string status;
12       NetworkStream nwStream;
13       byte[] bytesToSend, bytesToRead;
14       int bytesRead;
15       TcpClient client;
16
17       public Login()
18       {
19           InitializeComponent();
20           showConnected();
21       }
```

Kode 5. 6 Implementasi Cclass Login

Kode 5.6 menjelaskan bahwa dalam inisiasi pertama, aplikasi menggunakan port 9876. Port ini sama dengan port UDP Server

dari aplikasi server. Dan method showConnected digunakan untuk melihat apakah perangkat sudah terhubung dengan jaringan yang telah dibuat oleh Raspberry Pi.

```
23  private void btnLogin_Click(object sender, EventArgs e)
24  {
25      client = new TcpClient("192.168.4.1", 5000);
26      nwStream = client.GetStream();
27      if (textUser.Equals("") || textPassword.Equals(""))
28      {
29          textWarn.Visible = true;
30          textWarn.Text = "Harap masukkan username/Password";
31      }
32      else
33      {
34          textWarn.Visible = false;
35          string username = textUser.Text;
36          string password = textPassword.Text;
37          bytesToSend = ASCIIEncoding.ASCII.GetBytes(username);
38          nwStream.Write(bytesToSend, 0, bytesToSend.Length);
39          bytesToSend = ASCIIEncoding.ASCII.GetBytes(password);
40          nwStream.Write(bytesToSend, 0, bytesToSend.Length);
41          bytesToRead = new byte[client.ReceiveBufferSize];
42          bytesRead = nwStream.Read(bytesToRead, 0, client.ReceiveBufferSize);
43          string returnData = Encoding.ASCII.GetString(bytesToRead, 0, bytesRead);
44          status = returnData.ToString();

46          if (status.Equals("Sukses Login"))
47          {
48              client.Close();
49              Remote m = new Remote();
50              m.Show();
51              this.Hide();
52          }
53          else if (status.Equals("gagal login"))
54          {
55              textUser.Text = "";
56              textPassword.Text = "";
57              textWarn.Visible = true;
58              textWarn.Text = "Harap masukkan username/password dengan benar";
59          }
60      }
61  }
```

Kode 5. 7 Implementasi Button Login (Windows)

Kode 5.7 merupakan kode yang dijalankan saat pengguna menekan tombol “Log In”. Aplikasi akan membuat koneksi TCP pada server lali aplikasi akan mengecek apakah pengguna telah memasukkan username dan password. Jika telah diisi, aplikasi akan mengirimkan username dan password yang telah dimasukkan kepada Raspberry Pi. Jika username dan password

benar. Maka aplikasi akan mendapatkan pesan “Sukses Login” dan saat salah akan mendapatkan pesan “gagal login” dan kembali meminta pengguna untuk memasukkan username dan password.

```
63  private void btnRefresh_Click(object sender, EventArgs e)
64  {
65      showConnected();
66  }
67
68  private void showConnected()
69  {
70      System.Diagnostics.Process p = new System.Diagnostics.Process();
71      p.StartInfo.FileName = "netsh.exe";
72      p.StartInfo.Arguments = "wlan show interface";
73      p.StartInfo.UseShellExecute = false;
74      p.StartInfo.RedirectStandardOutput = true;
75      p.Start();
76
77      string s = p.StandardOutput.ReadToEnd();
78      if (s.Contains("SSID"))
79      {
80          string s1 = s.Substring(s.IndexOf("SSID"));
81          s1 = s1.Substring(s1.IndexOf(":"));
82          s1 = s1.Substring(2, s1.IndexOf("\n")).Trim();
83          p.WaitForExit();
85              if (s1 == "Pi3-AP")
86              {
87                  btnLogin.Enabled = true;
88                  textUser.Enabled = true;
89                  textWarn.Visible = false;
90                  textPassword.Enabled = true;
91              }
92              else
93              {
94                  textUser.Enabled = false;
95                  textPassword.Enabled = false;
96                  textWarn.Visible = true;
97                  textWarn.Text = "Please connect to Pi3-AP!";
98                  btnLogin.Enabled = false;
99              }
```

Kode 5. 8 Button Refresh (Windows)

showConnected pada Kode 5.6 dijelaskan pada Kode 5.8. Method ini mengecek jaringan yang terhubung pada perangkat dengan cara membuka aplikasi netsh dan menginputkan “wlan show interface”. Jika dijalankan menggunakan command prompt akan menghasilkan seperti pada Gambar 5.14. Pengambilan value dari SSID menggunakan fungsi dari substring.

Saat SSID tidak sama dengan “Pi3-AP”, kolom username dan password dapat diisi. Namun, saat SSID bukan “Pi3-AP” kolom username dan password tidak dapat diisi. Tombol Login juga bertindak sama. Tidak dapat ditekan saat SSID tidak benar dan dapat ditekan saat SSID benar.

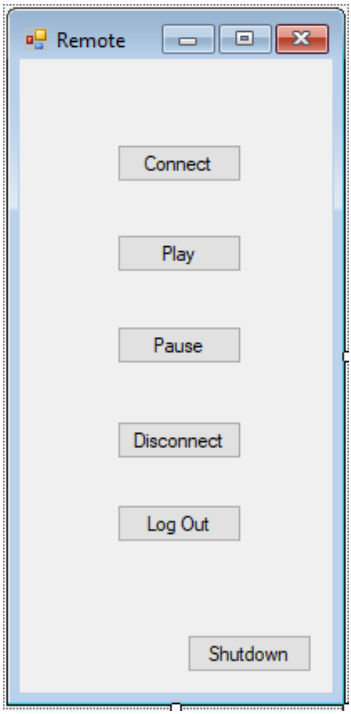
```
Name                   : Wi-Fi
Description            : Intel(R) Dual Band
Wireless-AC 7265
GUID                   : 557e4633-f4a1-48dc-8385-
1fd1667680dd
Physical address       : 88:78:73:b3:d7:ec
State                  : connected
SSID                   : AGUNG
BSSID                  : 54:a6:19:cc:dc:f8
Network type           : Infrastructure
Radio type             : 802.11n
Authentication         : WPA2-Personal
 Cipher                 : CCMP
```

Gambar 5. 10 Informasi Keluaran netsh

```
35          string username = textUser.Text;
36          string password = textPassword.Text;
37          bytesToSend = ASCIIEncoding.ASCII.GetBytes(username);
38          nwStream.Write(bytesToSend, 0, bytesToSend.Length);
39          bytesToSend = ASCIIEncoding.ASCII.GetBytes(password);
40          nwStream.Write(bytesToSend, 0, bytesToSend.Length);
```

Kode 5. 9 Method Send TCP pada Aplikasi Windows

Kode 5.9 adalah method untuk mengirim pesan menggunakan paket TCP. String dari input user diubah menjadi bentuk byte terlebih dahulu. Lalu byte dari string ini akan dikirimkan pada server TCP menggunakan nwStream.Write.



Gambar 5. 11 Tampilan Remote Aplikasi (Windows)

Tampilan remote Gambar 5.15 pada aplikasi windows memiliki 5 tombol sesuai dengan rancangan tampilan remote pada perangkat pengguna.

```
12      public partial class Remote : Form
13      {
14          string token = null;
15          static UdpClient receiveClient;
16          private string ipAddress;
17          private ImageStreamingServer _Server;
18          bool isRecord = false, core;
19          int widthScreen, heightScreen;
20
21          NetworkStream nwStream;
22          byte[] bytesToSend, bytesToRead;
23          int bytesRead;
24          TcpClient client;
25
26          public Remote()
27          {
28              InitializeComponent();
29              checkToken();
30              showIP();
31
32              //---create a TCPClient object at the IP and port no.---
33              client = new TcpClient("192.168.4.1", 5000);
34              nwStream = client.GetStream();
35              receiveClient = new UdpClient(9876);
36              IPEndPoint RemoteIpEndPoint = new IPEndPoint(IPAddress.Any, 0);
37              try
38              {
39                  Byte[] widthPiByte = new byte[client.ReceiveBufferSize];
40                  bytesRead = nwStream.Read(widthPiByte, 0, client.ReceiveBufferSize);
41                  string widthPi = Encoding.ASCII.GetString(widthPiByte, 0, bytesRead);
42                  Byte[] heightPiByte = new byte[client.ReceiveBufferSize];
43                  bytesRead = nwStream.Read(heightPiByte, 0, client.ReceiveBufferSize);
44                  string heightPi = Encoding.ASCII.GetString(heightPiByte, 0, bytesRead);
45                  widthPi = widthPi.Substring(0,widthPi.IndexOf(".")).Trim();
46                  heightPi = heightPi.Substring(0, heightPi.IndexOf(".")).Trim();
47                  widthScreen = int.Parse(widthPi);
48                  heightScreen = int.Parse(heightPi);
49              }
50              catch (Exception error)
51              {
52                  Console.WriteLine(error.ToString());
53              }
54
55              _Server = new ImageStreamingServer(widthScreen, heightScreen);
56          }
```

```
62   private void showIP()
63   {
64       System.Diagnostics.Process p = new System.Diagnostics.Process();
65       p.StartInfo.FileName = "ipconfig.exe";
66       p.StartInfo.UseShellExecute = false;
67       p.StartInfo.RedirectStandardOutput = true;
68       p.Start();
69
70       string s = p.StandardOutput.ReadToEnd();
71       if (s.Contains("Wireless LAN adapter"))
72       {
73           string s1 = s.Substring(s.IndexOf("Wireless LAN adapter"));
74           if (s1.Contains("IPv4 Address"))
75           {
76               s1 = s1.Substring(s1.IndexOf("IPv4 Address. . . . . . . . ."));
77               s1 = s1.Substring(s1.IndexOf(": "));
78               s1 = s1.Substring(2, s1.IndexOf("\n")).Trim();
79               ipAddress = s1;
80               p.WaitForExit();
81           }
82       }
```

Kode 5. 10 Implementasi Class Remote

Pada Kode 5.10 dapat dilihat aplikasi menggunakan method showIP. Method ini sama seperti method showConnected pada Kode 5.8. Yang membedakan adalah method ini mencari alamat ip dari perangkat yagn telah terhubung pada jaringan Raspberry Pi. Jika ipconfig dijalankan pada command prompt akan menghasilkan gambar seperti pada Gambar 5.16.

Wireless LAN adapter Wi-Fi:

```
Connection-specific DNS Suffix  . :
Link-local IPv6 Address . . . . . :
fe80::71db:707e:992f:da14%10
IPv4 Address. . . . . . . . . . . : 192.168.1.3
Subnet Mask . . . . . . . . . . . : 255.255.255.0
Default Gateway . . . . . . . . . : 192.168.1.1
```

Gambar 5. 12 Informasi Ipconfig pada Command Prompt

Tombol Connect mengirimkan pesan “0” kepada TCP Server untuk mendapatkan token dan akan menjalankan method checkToken. Lalu, Tombol Play memeriksa apakah sistem belum menjalankan thread pada class ImageScreenshot. Jika belum menjalankan thread, sistem akan menjalankan thread tersebut. Selain memeriksa thread, tombol play juga mengirimkan paket TCP berupa alamat ip, token dan perintah status. Kemudian, Tombol Pause memeriksa apakah sistem

sedang menjalankan thread atau tidak. Jika iya, kode akan mematikan thread tersebut dan mengirimkan paket UDP berupa nilai 0.

```
85  private void buttonConnect_Click(object sender, EventArgs e)
86  {
87      bytesToSend = ASCIIEncoding.ASCII.GetBytes("0");
88      nwStream.Write(bytesToSend, 0, bytesToSend.Length);
89      bytesToSend = ASCIIEncoding.ASCII.GetBytes("noIOS");
90      nwStream.Write(bytesToSend, 0, bytesToSend.Length);
91      IPEndPoint RemoteIpEndPoint = new IPEndPoint(IPAddress.Any, 0);
92      try
93      {
94          Byte[] receiveBytes = new byte[client.ReceiveBufferSize];
95          bytesRead = nwStream.Read(receiveBytes, 0, client.ReceiveBufferSize);
96          string returnData = Encoding.ASCII.GetString(receiveBytes, 0, bytesRead);
97
98          token = returnData.ToString();
99      }
100     catch (Exception error)
101     {
102         Console.WriteLine(error.ToString());
103     }
104     checkToken();
105 }
107 private void buttonPlay_Click(object sender, EventArgs e)
108 {
109     if (!isRecord)
110     {
111         _Server.Start(1234);
112         ScreenAppNetworking.Streaming.Screen.token = token;
113         ScreenAppNetworking.Streaming.Screen.ipAdrress = ipAddress;
114         ScreenAppNetworking.Streaming.Screen.loop = true;
115         isRecord = true;
116     }
117     bytesToSend = ASCIIEncoding.ASCII.GetBytes(ipAddress);
118     nwStream.Write(bytesToSend, 0, bytesToSend.Length);
119     Thread.Sleep(500);
120     bytesToSend = ASCIIEncoding.ASCII.GetBytes(token);
121     nwStream.Write(bytesToSend, 0, bytesToSend.Length);
122     Thread.Sleep(500);
123     bytesToSend = ASCIIEncoding.ASCII.GetBytes("1");
124     nwStream.Write(bytesToSend, 0, bytesToSend.Length);
125     buttonPlay.Enabled = false;
126     buttonPause.Enabled = true;
127 }
```

```
129   private void buttonPause_Click(object sender, EventArgs e)
130   {
131       if (isRecord)
132       {
133           isRecord = false;
134           ScreenAppNetworking.Streaming.Screen.loop = false;
135       }
136       Thread.Sleep(500);
137       SendUdp(9875, "192.168.4.1", 9875, Encoding.ASCII.GetBytes("0"));
138       buttonPlay.Enabled = true;
139       buttonPause.Enabled = false;
140   }
```

Kode 5. 11 Tombol Connect, Play, dan Pause (Windows)

```
143   private void buttonDisconnect_Click(object sender, EventArgs e)
144   {
145       if (isRecord)
146       {
147           isRecord = false;
148           ScreenAppNetworking.Streaming.Screen.loop = false;
149           Thread.Sleep(500);
150           SendUdp(9875, "192.168.4.1", 9875, Encoding.ASCII.GetBytes("0"));
151       }
152       bytesToSend = ASCIIEncoding.ASCII.GetBytes(ipAddress);
153       nwStream.Write(bytesToSend, 0, bytesToSend.Length);
154       Thread.Sleep(500);
155       bytesToSend = ASCIIEncoding.ASCII.GetBytes(token);
156       nwStream.Write(bytesToSend, 0, bytesToSend.Length);
157       Thread.Sleep(500);
158       bytesToSend = ASCIIEncoding.ASCII.GetBytes("2");
159       nwStream.Write(bytesToSend, 0, bytesToSend.Length);
160       token = null;
161       checkToken();
162   }
164   private void buttonLogout_Click(object sender, EventArgs e)
165   {
166       if (isRecord)
167       {
168           isRecord = false;
169           ScreenAppNetworking.Streaming.Screen.loop = false;
170           Thread.Sleep(500);
171           SendUdp(9875, "192.168.4.1", 9875, Encoding.ASCII.GetBytes("0"));
172       }
173       if (token != null)
174       {
175           bytesToSend = ASCIIEncoding.ASCII.GetBytes(ipAddress);
176           nwStream.Write(bytesToSend, 0, bytesToSend.Length);
177           Thread.Sleep(500);
178           bytesToSend = ASCIIEncoding.ASCII.GetBytes(token);
179           nwStream.Write(bytesToSend, 0, bytesToSend.Length);
180           Thread.Sleep(500);
181       }
```

```
182         bytesToSend = ASCIIEncoding.ASCII.GetBytes("3");
183         nwStream.Write(bytesToSend, 0, bytesToSend.Length);
184         token = null;
185         client.Close();
186         System.Environment.Exit(1);
187     }
```

Kode 5. 12 Tombol Disconnect dan Logout (Windows)

Tombol disconnect memiliki fungsi untuk mereset nilai token dan perintah yang sama seperti tombol pause. Namun memiliki perintah status '2'. Sedangkan tombol Logout memiliki nilai status '3' dan menampilkan aplikasi pada tampilan login

```
195     private void button1_Click(object sender, EventArgs e)
196     {
197         if (token != null)
198         {
199             bytesToSend = ASCIIEncoding.ASCII.GetBytes(ipAddress);
200             nwStream.Write(bytesToSend, 0, bytesToSend.Length);
201             Thread.Sleep(500);
202             bytesToSend = ASCIIEncoding.ASCII.GetBytes(token);
203             nwStream.Write(bytesToSend, 0, bytesToSend.Length);
204             Thread.Sleep(500);
205         }
206         bytesToSend = ASCIIEncoding.ASCII.GetBytes("4");
207         nwStream.Write(bytesToSend, 0, bytesToSend.Length);
208         if (isRecord)
209             token = null;
210         client.Close();
211         Login m = new Login();
212         m.Show();
213         this.Close();
214     }
```

Kode 5. 13 Tombol Shutdown (Windows)

Tombol Shutdown pada Kode 5.13 berfungsi untuk mereset nilai token dan meminta pengguna untuk login kembali.

```
190   static void SendUdp(int srcPort, string dstIp, int dstPort, byte[] data)
191   {
192       receiveClient.Send(data, data.Length, dstIp, dstPort);
193   }
216   private void checkToken()
217   {
218       if (token == null)
219       {
220           buttonConnect.Enabled = true;
221           buttonPlay.Enabled = false;
222           buttonDisconnect.Enabled = false;
223           buttonPause.Enabled = false;
224           buttonLogout.Enabled = true;
225       }
226       else if(token != null)
227       {
228           buttonConnect.Enabled = false;
229           buttonPlay.Enabled = true;
230           buttonDisconnect.Enabled = true;
231           buttonPause.Enabled = true;
232           buttonLogout.Enabled = true;
233       }
234   }
262       public static string GenerateSHA512String(string inputString)
263       {
264           SHA512 sha512 = SHA512Managed.Create();
265           byte[] bytes = Encoding.UTF8.GetBytes(inputString);
266           byte[] hash = sha512.ComputeHash(bytes);
267           return GetStringFromHash(hash);
268       }
269
270       private static string GetStringFromHash(byte[] hash)
271       {
272           StringBuilder result = new StringBuilder();
273           for (int i = 0; i < hash.Length; i++)
274           {
275               result.Append(hash[i].ToString("X2"));
276           }
277           return result.ToString();
278       }
```

Kode 5. 14 Method Pendukung (Windows)

Method check token berfungsi untuk mengecek apakah aplikasi sudah memiliki token atau belum. Saat token tidak null, tombol play, pause dan disconnect dapat ditekan. Sedangkan saat token bernilai null, tombol play, pause dan disconnect tidak dapat ditekan. Dengan adanya checkToken ini mewajibkan pengguna unuk menekan tombol connect atau tombol logout saat tidak memiliki token. Serta untuk mendapatkan hash SHA512 dari token, digunakan method GenerateSHA512String.

```
10  namespace ScreenAppNetworking.Streaming
11  {
12      public class ImageStreamingServer:IDisposable
13      {
14          private List<Socket> _Clients;
15          private Thread _Thread;
16
17          public ImageStreamingServer(int width , int height)
18              :this(Screen.Snapshots(width,height,true))
19          {}
20
21          public ImageStreamingServer(IEnumerable<Image> imagesSource)
22          {
23              _Clients = new List<Socket>();
24              _Thread = null;
25
26              this.ImagesSource = imagesSource;
27              this.Interval = 50;
28          }
29
30
31          public IEnumerable<Image> ImagesSource { get; set; }
32
33          public int Interval { get; set; }
34
35          public IEnumerable<Socket> Clients { get { return _Clients; } }
36
37          public bool IsRunning { get {
38                  return (_Thread != null && _Thread.IsAlive); } }
39
40          /// <param name="port"></param>
41          public void Start(int port)
42          {
43
44              lock (this)
```

Kode 5. 15 Implementasi ImageScreenshot (Windows)

Class ImageStreamingServer adalah implementasi class ImageScreenshot. Class ini berfungsi untuk menjalankan thread dan mengatur interval tiap akan menjalankan thread.

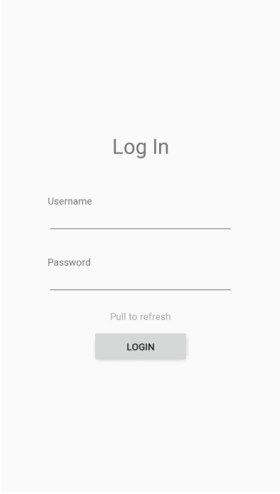
```
156  public static class Screen
157  {
158      static UdpClient receiveClient1;
159      public static string token = ImageStreamingServer.token;
160      public static string ipAdrress = ImageStreamingServer.ipAddress;
161      public static Boolean loop = true;
162
163      public static IEnumerable<Image> Snapshots()
164      {
165          return Screen.Snapshots(System.Windows.Forms.Screen.PrimaryScreen.Bounds.Width,
166          System.Windows.Forms.Screen.PrimaryScreen.Bounds.Height, true);
167      }
174   public static IEnumerable<Image> Snapshots(int width, int height, bool showCursor)
175  {
176
177      Size size = new Size(System.Windows.Forms.Screen.PrimaryScreen.Bounds.Width,
178      System.Windows.Forms.Screen.PrimaryScreen.Bounds.Height);
179
180      Bitmap srcImage = new Bitmap(size.Width, size.Height);
181      Graphics srcGraphics = Graphics.FromImage(srcImage);
182
183      bool scaled = (width != size.Width || height != size.Height);
184
185      Bitmap dstImage = srcImage;
186      Graphics dstGraphics = srcGraphics;
187
188      if (scaled)
189      {
190          dstImage = new Bitmap(width, height);
191          dstGraphics = Graphics.FromImage(dstImage);
192      }
193
194      Rectangle src = new Rectangle(0, 0, size.Width, size.Height);
195      Rectangle dst = new Rectangle(0, 0, width, height);
196      Size curSize = new Size(32, 32);
198     while (loop)
199     {
200         receiveClient1 = new UdpClient(9875);
201         srcGraphics.CopyFromScreen(0, 0, 0, 0, size);
202
203         if (showCursor)
204             Cursors.Default.Draw(srcGraphics, new Rectangle(Cursor.Position, curSize));
205
206         if (scaled)
207             dstGraphics.DrawImage(srcImage, dst, src, GraphicsUnit.Pixel);
208
209         string quarterString = "";
210         string sizeMod = "tidak pas";
211
212         string baseImage = Convert.ToBase64String(VaryQualityLevel(dstImage));
213         baseImage = ipAdrress+token+baseImage;
214         int length = System.Text.Encoding.ASCII.GetBytes(baseImage).Length;
215         Debug.WriteLine(length);
216         if (length<65400)
217         SendUdp(9875, "192.168.4.1", 9875,
218         System.Text.Encoding.ASCII.GetBytes(baseImage));
219
220         receiveClient1.Close();
221         yield return dstImage;
222
223     }
```

Kode 5. 16 Implementasi Screenshot (Windows)

Kelas Screen merupakan inner class dari kelas ImageScreenshot. Kelas ini berfungsi sebagai menangkap gambar dari layar (screenshot). Jika resolusi pada konstruktor ImageScreenshot lebih kecil atau lebih besar daripada layar perangkat. Maka aplikasi melakukan scaling terhadap gambar screenshot. Kemudian aplikasi akan mengirimkan paket UDP berupa alamat ip, hash token dan byte gambar kepada server.

5.2.3 Implementasi Aplikasi Android

Implementasi aplikasi android menggunakan Android Studio dengan menggunakan teknik MediaProjection pada layar. Biasanya MediaProjection ini digunakan aplikasi android saat ingin merekam layar handphone. Namun, file streaming tidak bisa menunggu hasil rekaman layar yang harus selesai terlebih dahulu. Jika menunggu hasil rekaman maka video yang didapat tidaklah realtime. Berikut hasil implementasi dari aplikasi android.



Gambar 5. 13 Tampilan Login (Android)

Tampilan Login pada aplikasi android memiliki input berupa username dan password. Input berupa password tidak dapat dilihat dalam bentuk latin (disensor). Untuk mengecek kembali

jaringan yang terhubung, layar dapat ditarik kebawah (swipe to refresh).

```
public class Login extends AppCompatActivity implements View.OnClickListener {
    String ssid,username,password;
    Button btnLogin;
    TextView label;
    EditText txtUser,txtPass;
    SwipeRefreshLayout pullToRefresh;
    int  port1 = 5000;
    String ip = "192.168.4.1";
    private Socket tcpSocket;
    InetAddress serverAddr;
    SharedPreferences pref;
    Intent intent;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.login);
        StrictMode.ThreadPolicy policy =
        new StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);

        btnLogin = (Button)findViewById(R.id.btnLogin);
        txtUser = (EditText)findViewById(R.id.textUsername);
        txtPass = (EditText)findViewById(R.id.textPassword);
        label = (TextView)findViewById(R.id.labelId);
        pullToRefresh = findViewById(R.id.pullToRefresh);

        WifiManager wifiManager = (WifiManager) getApplicationContext()
        .getSystemService (Context.WIFI_SERVICE);
        WifiInfo info = wifiManager.getConnectionInfo ();
        ssid  = info.getSSID();
        if (!ssid.equals("Pi3-AP")) {
            btnLogin.setEnabled(false);
            Toast.makeText(this, "Please connect to Pi3-AP wifi :)",
            Toast.LENGTH_LONG).show();
        }
        else if(ssid.equals("Pi3-AP")){
            btnLogin.setEnabled(true);
            Toast.makeText(this, "Enter Username and Password correctly",
            Toast.LENGTH_LONG).show();
        }
```

```
77          pullToRefresh.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {
78              @Override
79              public void onRefresh() {
80                  refreshData();
81                  pullToRefresh.setRefreshing(false);
82              }
83          });
84          btnLogin.setOnClickListener(this);
85          pref = getSharedPreferences("user_details",MODE_PRIVATE);
86          intent = new Intent(this,Remote.class);
87          if(pref.contains("username") && pref.contains("password")){
88  //              udpSocket.close();
89              try {
90                  tcpSocket.close();
91              } catch (IOException e) {
92                  e.printStackTrace();
93              }
94              startActivity(intent);
95              this.finish();
96          }
97          refreshData();
```

```
122  private void login() {
123      try {
124          serverAddr = InetAddress.getByName(ip);
125          tcpSocket = new Socket(serverAddr, port1);
126      } catch (IOException e) {
127          e.printStackTrace();
128      }
129      username = txtUser.getText().toString();
130      password = txtPass.getText().toString();
131      try {
132          sendTCP(username);
133          Thread.sleep(1000);
134          sendTCP(password);
135          String text = receiveMessagge();
136          Log.d("Received data", text);
137          if (text.equals("Sukses Login")){
138              Log.d("Received data", "sini");
139              SharedPreferences.Editor editor = pref.edit();
140              editor.putString("username",username);
141              editor.putString("password",password);
142              editor.apply();
143              Toast.makeText(getApplicationContext(), "Login Successful",
144              Toast.LENGTH_SHORT).show();
145              //udpSocket.close();
146              Log.d("Received data", "sini");
147              tcpSocket.close();
148              Log.d("Received data", "sini");
149              startActivity(intent);
150              this.finish();
151          }else if(text.equals("gagal login")){
152              Toast.makeText(this, "Enter Username and Password correctly",
153              Toast.LENGTH_LONG).show();
154          }
155      } catch (IOException e) {
156          e.printStackTrace();
157      } catch (InterruptedException e) {
158          e.printStackTrace();
159      }
160  }
162  private void sendTCP(String sendMessage){
163      try {
164          OutputStream os = tcpSocket.getOutputStream();
165          byte send[] = sendMessage.getBytes();
166          os.write(send);
167          Thread.sleep(500);
168          Log.d("sendData", sendMessage);
169      } catch (IOException e) {
170          e.printStackTrace();
171      } catch (InterruptedException e) {
172          e.printStackTrace();
173      }
174  }
```

```
176 private String receiveMessagge() throws IOException {
177     String text="";
178         Log.i("TCP client: ", "about to wait to receive");
179     InputStream is = tcpSocket.getInputStream();
180     ByteArrayOutputStream baos = new ByteArrayOutputStream();
181     byte buffer[]= new byte[1024];
182     baos.write(buffer,0,is.read(buffer));
183     byte result[] = baos.toByteArray();
184     text = new String(result);
185     Log.d("Receive","Message received from the server : " +text);
186     return text;
187 }
```

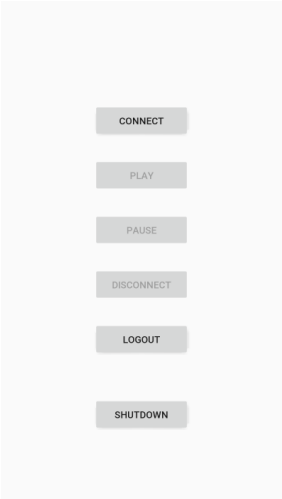
Kode 5. 17 Implementasi Class Login (Android)

Kode 5.17 merupakan implementasi class login pada aplikasi android. Mengecek jaringan pada perangkat menggunakan method refreshData. Pada Android, nama SSID dari WifiManager memiliki tambahan simbol slash (“/”). Maka dari itu untuk memastikan menggunakan if else penulis menggunakan slash.

Jika jaringan dari perangkat pengguna tidak sama dengan “Pi3-AP”. Maka tombol login tidak dapat ditekan. Namun, jika perangkat terhubung dengan jaringan Raspberry Pi. Maka tombol login dapat ditekan.

Untuk mengirim pesan ke TCP Server pada Raspberry Pi, aplikasi android menggunakan port 5000. Pesan username dan password diubah kedalam bentuk byte lalu dikirim.

Saat username dan password yang dimasukkan oleh pengguna benar. Aplikasi akan menerima pesan berupa “Sukses Login”. Dan jika salah akan menerima pesan “gagal login” dan meminta pengguna untuk memasukkan username dan password lagi.



Gambar 5. 14 Tampilan Remote (Android)

Tampilan remote Gambar 5.18 pada aplikasi android memiliki 6 tombol sesuai dengan rancangan tampilan remote pada perangkat pengguna.

```
45  public class Remote extends AppCompatActivity implements View.OnClickListener {
46      private static final String TAG = "Remote";
47      private Button btnConnect, btnPlay, btnPause, btnDisconnect, btnLogout, btn
48      private int port1 = 5000, port2 = 9875;
49      private String ip = "192.168.4.1";
50      private byte[] buf;
51      private Socket tcpSocket;
52      private DatagramPacket packet;
53      private DatagramSocket udpSocket,udpSocket2;
54      private InetAddress serverAddr;
55      private String status, token,clientIp;
56      private SharedPreferences prf;
57      private Intent intent;
58      private double scaleFactor=0;
59      private int mScreenDensity;
60      private boolean isRecord = false;
61      private static final int REQUEST_CODE = 100;
62      private static String STORE_DIRECTORY;
63      private static int IMAGES_PRODUCED;
```

```
109         prf = getSharedPreferences("user_details",MODE_PRIVATE);
110         intent = new Intent(this,Login.class);
111         txtUser.setText("Hello, "+prf.getString("username",null));
112         DisplayMetrics metrics = new DisplayMetrics();
113         getWindowManager().getDefaultDisplay().getMetrics(metrics);
114         mScreenDensity = metrics.densityDpi;
115         if (scaleFactor ==0){
116             try {
117                 scale();
118             } catch (IOException e) {
119                 e.printStackTrace();
120             }
121         }
136     public void onClick(View view) {
137         if (view == btnConnect) {
138             status = "0";
139             sendTCP(status);
140             sendTCP("noIOS");
141             try {
142                 token = receiveMessagg
143                 checkToken();
144             } catch (IOException e) {
145                 e.printStackTrace();
146             }
147         }
148         if (view == btnPlay) {
149             status = "1";
150             if (!isRecord) {
151                 startProjection();
152                 isRecord=true;
153             }
154             sendTCP(clientIp);
155             sendTCP(token);
156             sendTCP(status);
157             btnPlay.setEnabled(false);
158             btnPause.setEnabled(true);
159         }
160         if (view == btnPause) {
161             status = "0";
162             if (isRecord){
163                 stopProjection();
164                 isRecord=false;
165             }
166             try {
167                 Thread.sleep(500);
168             } catch (InterruptedException e) {
169                 e.printStackTrace();
170             }
171             sendCommand1(status);
172             btnPlay.setEnabled(true);
173             btnPause.setEnabled(false);
174         }
```

```
175         if (view == btnDisconnect) {
176             status = "2";
177             if (isRecord){
178                 stopProjection();
179                 isRecord=false;
180                 try {
181                     Thread.sleep(500);
182                 } catch (InterruptedException e) {
183                     e.printStackTrace();
184                 }
185                 sendCommand1("0");
186             }
187             sendTCP(clientIp);
188             sendTCP(token);
189             sendTCP(status);
190             token = null;
191             checkToken();
192         }
```

Kode 5. 18 Implementasi Class Remote (Android)

Implementasi class Remote pada aplikasi android diubah penulis menjadi class AppActivity. Karena sifat aplikasi android dimana saat keluar dari aplikasi, aplikasi akan membuka tampilan pertama. Aplikasi Android pada tugas akhir ini membutuhkan session. Session dibuat agar saat pengguna telah melakukan login tidak melakukan login kembali saat keluar dari aplikasi.

Sama seperti aplikasi pada Windows, setiap tombol pada aplikasi android akan memeriksa apakah alamat ip dan token pada aplikasi pengguna dan Raspberry Pi sama.

```
255     private void sendTCP(String sendMessage){
256         try {
257             OutputStream os = tcpSocket.getOutputStream();
258             byte send[] = sendMessage.getBytes();
259             os.write(send);
260             Thread.sleep(500);
261             Log.d("sendData", sendMessage);
262         } catch (IOException e) {
263             e.printStackTrace();
264         } catch (InterruptedException e) {
265             e.printStackTrace();
266         }
```

```
268     private void sendCommand1(String status) {
269         try {
270             buf = (status).getBytes();
271             packet = new DatagramPacket(buf, buf.length, serverAddr, port2);
272             udpSocket2.send(packet);
273 
274         } catch (IOException e) {
275             e.printStackTrace();
276         }
277     }
317     private void checkToken() {
318         if(token== null){
319             btnConnect.setEnabled(true);
320             btnPlay.setEnabled(false);
321             btnPause.setEnabled(false);
322             btnDisconnect.setEnabled(false);
323             btnLogout.setEnabled(true);
324         }else if(token != null){
325             btnConnect.setEnabled(false);
326             btnPlay.setEnabled(true);
327             btnPause.setEnabled(true);
328             btnDisconnect.setEnabled(true);
329             btnLogout.setEnabled(true);
330         }
331     }
304     private String receiveMessagge() throws IOException {
305         String text="";
306         Log.i("TCP client: ", "about to wait to receive");
307         InputStream is = tcpSocket.getInputStream();
308         ByteArrayOutputStream baos = new ByteArrayOutputStream();
309         byte buffer[]= new byte[1024];
310         baos.write(buffer,0,is.read(buffer));
311         byte result[] = baos.toByteArray();
312         text = new String(result);
313         Log.d("Receive","Message received from the server : " +text);
314         return text;
315     }
318 static public String get_SHA_512_SecurePassword(String passwordToHash, String salt){
319     String generatedPassword = null;
320     try {
321         MessageDigest md = MessageDigest.getInstance("SHA-512");
322         md.update(salt.getBytes(StandardCharsets.UTF_8));
323         byte[] bytes = md.digest(passwordToHash.getBytes(StandardCharsets.UTF_8));
324         StringBuilder sb = new StringBuilder();
325         for(int i=0; i< bytes.length ;i++){
326             sb.append(Integer.toString((bytes[i] & 0xff) + 0x100, 16).substring(1));
327         }
328         generatedPassword = sb.toString();
329     } catch (NoSuchAlgorithmException e) {
330         e.printStackTrace();
331     }
332     return generatedPassword;
333 }
```

Kode 5. 19 Method Pendukung (Android)

Method check token berfungsi untuk mengecek apakah aplikasi sudah memiliki token atau belum. Saat token tidak null, tombol play, pause dan disconnect dapat ditekan. Sedangkan saat token bernilai null, tombol play, pause dan disconnect tidak dapat ditekan. Dengan adanya checkToken ini mewajibkan pengguna unuk menekan tombol connect atau tombol logout saat tidak memiliki token. Serta untuk mendapatkan hash SHA512 dari token, digunakan method get_SHA_512_SecurePassword.

```
279 private void scale() throws IOException {
280     String widthPi = receiveMessagge();
281     String heightPi = receiveMessagge();
282     widthPi = widthPi.substring(0,widthPi.indexOf(".")).trim();
283     heightPi = heightPi.substring(0,heightPi.indexOf(".")).trim();
284     int widthPiScreen = Integer.parseInt(widthPi);
285     int heightPiScreen = Integer.parseInt(heightPi);
286     DisplayMetrics displayMetrics = new DisplayMetrics();
287     getWindowManager().getDefaultDisplay().getMetrics(displayMetrics);
288     int widthDevice = displayMetrics.widthPixels;
289     int heigthDevice = displayMetrics.heightPixels;
290     Boolean scaled = (widthDevice!=widthPiScreen || heigthDevice!=heightPiScreen);
291     if (scaled){
292         double x = (double)widthDevice/(double)widthPiScreen;
293         double y = (double)heigthDevice/(double)heightPiScreen;
294         if (x>y)
295             scaleFactor = ((double)widthDevice /(double)widthPiScreen);
296         else if (y>x)
297             scaleFactor = ((double)heigthDevice /(double)heightPiScreen);
298     }
299 
300     Log.e(TAG, "scale: " + scaleFactor);
301 
302 }
```

Kode 5. 20 Method Scale (Android)

Fungsi scale bertujuan untuk memeriksa apakah perlu dilakukan pengaturan skala gambar atau tidak. Perbandingan skala menggunakan resolusi layar Raspberry Pi dengan layar perangkat.

```
393 private void createVirtualDisplay() {
394     // get width and height
395     Point size = new Point();
396     mDisplay.getSize(size);
397     mWidth = size.x;
398     mHeight = size.y;
399     // start capture reader
400     mImageReader = ImageReader.newInstance(mWidth, mHeight, PixelFormat.RGBA_8888, 2);
401     mVirtualDisplay = sMediaProjection
402     .createVirtualDisplay(SCREENCAP_NAME, mWidth, mHeight,
403     mDensity, VIRTUAL_DISPLAY_FLAGS, mImageReader.getSurface(), null, mHandler);
404     mImageReader.setOnImageAvailableListener(new ImageAvailableListener(), mHandler);
405 }
```

```
425   Bitmap scaledBitmap = null;
426   bitmap = Bitmap.createBitmap(mWidth + rowPadding / pixelStride, mHeight,
427   Bitmap.Config.ARGB_8888);
428   bitmap.copyPixelsFromBuffer(buffer);
429   ByteArrayOutputStream bos = new ByteArrayOutputStream();
430   scaledBitmap = Bitmap.createScaledBitmap(bitmap, (int) (mWidth / scaleFactor),
431   (int) (mHeight / scaleFactor), false);
432   scaledBitmap.compress(CompressFormat.JPEG, 50, bos);
433   text = clientIp+token+Base64.encodeToString(bos.toByteArray(), Base64.NO_WRAP);
434   IMAGES_PRODUCED++;
435   Log.e(TAG, "captured image: " + IMAGES_PRODUCED);
436   Log.e(TAG, "ImageSize: " + text.length());
437   sendCommand1(text);
461   private class OrientationChangeCallback extends OrientationEventListener {
462       OrientationChangeCallback(Context context) {
463           super(context);
464       }
465       @Override
466       public void onOrientationChanged(int orientation) {
467           final int rotation = mDisplay.getRotation();
468           if (rotation != mRotation) {
469               mRotation = rotation;
470               try {
471                   // clean up
472                   if (mVirtualDisplay != null) mVirtualDisplay.release();
473                   if (mImageReader != null) mImageReader.setOnImageAvailableListener(
474
475                   // re-create virtual display depending on device width / height
476                   createVirtualDisplay();
477               } catch (Exception e) {
478                   e.printStackTrace();
479               }
480           }
481       }
482   }
```

Kode 5. 21 Implementasi Screenshot (Android)

Aplikasi Android memerlukan bantuan dari MediaProjection untuk mendapatkan gambar pada layar perangkat. Hal ini dikarenakan jika aplikasi menangkap gambar secara langsung dapat melanggar ketentuan yang dimiliki oleh Android. Maka dari itu untuk mengambil gambar pada layar harus memakai MediaProjection dan membuat VirtualDisplay untuk menyalin layar pada perangkat pada layar virtual. Gambar yang dibuat kemudian diubah tingkat kualitasnya seperti pada baris 432. Gambar ini kemudian diubah menjadi base64 dan didapatkan bytenya. Paket UDP yang akan dikirim adalah gabungan dari alamat ip perangkat, token dan byte gambar.

Inner class OrientationChangeCallback adalah class untuk mengatur gambar saat terjadi perubahan orientasi pada perangkat android.

5.2.4 Implementasi Aplikasi iOS

Implementasi aplikasi iOS menggunakan XCode versi 11.2.1 dengan menggunakan fungsi bawaan dari iPhone, airPlay. Dikarenakan fungsi dari airPlay adalah untuk terhubung dengan perangkat dari Apple. Perlu adanya modifikasi terhadap Raspberry Pi agar dapat bertindak seperti AppleTV. Penambahan software RPi-Play open source dapat mengatasi kebutuhan tersebut. Namun kekurangan dari software ini sering terjadi bug pengguna terjebak didalamnya dan jalan terakhir untuk keluar adalah shutdown/reboot. Berikut adalah langkah penambahan RPi-Play pada Raspberry Pi.

Unduh software dari GitHub dengan memasukkan perintah dibawah menggunakan terminal.

```
git clone https://github.com/FD-/RPiPlay.git
cd RPiPlay
```

Install software pendukung dari RPi-Play dengan memasukkan perintah dibawah menggunakan terminal. Lalu masuk ke dalam folder build.

```
sudo apt-get install cmake
sudo apt-get install libavahi-compat-libdnssd-dev
sudo apt-get install libssl-dev
cd build
cmake ..
make
```

Untuk memulai service, dapat dilakukan didalam folder build dengan memasukkan perintah pada terminal “./rpiplay”. Agar service dapat dijalankan langsung memalui terminal. Perlu dibuat perintah bash di dalam bin pada Raspberry Pi. Buat file

dan masukkan perintah dibawah lalu simpan dengan nama rpiplay.

```
#!/bin/bash
Cd RPi-Play
Cd build
./rpiplay
```

Lalu ubah permission file tersebut agar dapat dijalankan dengan perintah chmod “+x rpiplay”.

Setelah melakukan penambahan software RPi-Play kedalam Raspberry Pi. Berikut hasil implementasi dari aplikasi iOS agar dapat terhubung dengan sistem yang dibuat.

Gambar 5. 15 Tampilan Login (iOS)

Tampilan Login pada aplikasi iOS memiliki input berupa username dan password. Input berupa password tidak dapat dilihat dalam bentuk latin (disensor). Untuk mengecek kembali

jaringan yang terhubung, layar dapat ditarik kebawah (swipe to refresh).

```
16  class HomeController: UIViewController,CLLocationManagerDelegate {
17      @IBOutlet weak var userTexxt: UITextField!
18      @IBOutlet weak var passText: UITextField!
19      @IBOutlet weak var loginButton: UIButton!
20      @IBOutlet var gestureControl: UISwipeGestureRecognizer!
21
22      var ssid: String? = nil
23      var locationManager: CLLocationManager?
24      var refreshControl = UIRefreshControl()
25      var message: String = ""
26      var client = TCPClient(address: "192.168.4.1", port: 5000)
27      override func viewDidLoad() {
28          super.viewDidLoad()
29          // Do any additional setup after loading the view.
30          navigationItem.hidesBackButton = true
31
32          let swipeLeft = UISwipeGestureRecognizer(target: self,
33          action: #selector(handleGesture))
34          swipeLeft.direction = .left
35          self.view.addGestureRecognizer(swipeLeft)
36
37          let swipeRight = UISwipeGestureRecognizer(target: self,
38          action: #selector(handleGesture))
39          swipeRight.direction = .right
40          self.view.addGestureRecognizer(swipeRight)
41
42          let swipeUp = UISwipeGestureRecognizer(target: self,
43          action: #selector(handleGesture))
44          swipeUp.direction = .up
45          self.view.addGestureRecognizer(swipeUp)
46
47          let swipeDown = UISwipeGestureRecognizer(target: self,
48          action: #selector(handleGesture))
49          swipeDown.direction = .down
50          self.view.addGestureRecognizer(swipeDown)
51
52          locationManager = CLLocationManager()
53          locationManager?.delegate = self
54          locationManager?.requestAlwaysAuthorization()
55          let tap: UITapGestureRecognizer = UITapGestureRecognizer(target: self,
56          action: #selector(UIInputViewController.dismissKeyboard))
57          view.addGestureRecognizer(tap)
58          checkWifi()
```

```
62  @objc func dismissKeyboard() {
63      //Causes the view (or one of its embedded text fields) to resign t
64      view.endEditing(true)
65  }
66  @objc func handleGesture(gesture: UISwipeGestureRecognizer) -> Void {
67      if gestureControl.direction == .right {
68              showToast(message: "Pull to refresh")
69      }
70      else if gestureControl.direction == .left {
71          showToast(message: "Pull to refresh")
72      }
73      else if gestureControl.direction == .up {
74          showToast(message: "Pull to refresh")
75      }
76      else if gestureControl.direction == .down {
77          checkWifi()
78      }
79  }
81  @IBAction func loginButtonPressed(_ sender: Any) {
82      switch client.connect(timeout: 10){
83      case .success:
84          let defaults = UserDefaults.standard
85          let username = userTexxt.text
86          let password = passText.text
87
88
89          //send username & password
90          sendTCP(username!)
91          sendTCP(password!)
92
93          print("sini")
94          message = receiveTCP()
95          var x = 0
96          while(x<500000000) {
97              x+=1
98          }
99          print(message)
100
101         if message == "Sukses Login"{
102             defaults.set(username, forKey: "username")
103             defaults.set(password, forKey: "password")
104             defaults.synchronize()
105             showToast(message: "Login Successfull")
106             //self.connectionHome = nil
107             client.close()
108             navigationController?.popViewController(animated: true)
109             dismiss(animated: true, completion: nil)
```

```
111             }
112             else if message == "gagal login"{
113                 showToast(message: "Enter username and password correctly")
114                 userTexxt.text = ""
115                 passText.text = ""
116             }
117         case .failure(let error):
118             print(error)
119         }
123     func getWiFiSsid() -> String? {
124         var ssid: String?
125         if let interfaces = CNCopySupportedInterfaces() as NSArray? {
126             for interface in interfaces {
127                 if let interfaceInfo = CNCopyCurrentNetworkInfo(
128                     interface as! CFString) as NSDictionary? {
129                     ssid = interfaceInfo[kCNNetworkInfoKeySSID as String]
130                         as? String
131                     break
132                 }
133             }
134         }
135         return ssid
136     }
137
138     func showToast(message : String) {
139
140         let toastLabel = UILabel(frame: CGRect(
141             x: self.view.frame.size.width/2 - 75,
142             y: self.view.frame.size.height-150, width: 150, height: 100))
143         toastLabel.backgroundColor = UIColor.black.withAlphaComponent(0.6)
144         toastLabel.textColor = UIColor.white
145         toastLabel.textAlignment = .center;
146         toastLabel.text = message
147         toastLabel.alpha = 1.0
148         toastLabel.layer.cornerRadius = 10;
149         toastLabel.clipsToBounds  =  true
150         toastLabel.lineBreakMode = .byWordWrapping
151         toastLabel.numberOfLines = 3
152         self.view.addSubview(toastLabel)
153         UIView.animate(withDuration: 4.0, delay: 0.1,
154                        options: .curveEaseOut, animations: {
155             toastLabel.alpha = 0.0
156         }, completion: {(isCompleted) in
157             toastLabel.removeFromSuperview()
158         })
159     }
161     func checkWifi(){
162         ssid = getWiFiSsid()
163         if ssid != "Pi3-AP"{
164             self.loginButton.isEnabled = true
165             showToast(message: "Please connect to Pi3-AP wifi")
166         }
167         else if ssid == "Pi3-AP"{
168             self.loginButton.isEnabled = true
169             showToast(message: "Enter username and password")
```

```
172  func sendTCP(_ content: String) {
173      let contentToSendUDP = content.data(using: String.Encoding.utf8)
174      _ = client.send(data: contentToSendUDP!)
175      sleep(1)
176  }
177
178
179  func receiveTCP() ->String{
180      let data = client.read(1024*10)
181      let text = String(bytes: data!, encoding: .utf8)!
182      print(text)
183      return text
184  }
```

Kode 5. 22 Implementasi Class Login (iOS)

Kode 5.22 merupakan implementasi class login pada aplikasi iOS. Untuk mengirim pesan ke TCP Server pada Raspberry Pi, aplikasi android menggunakan port 5000. Pesan username dan password diubah kedalam bentuk byte lau dikirim.

Saat username dan password yang dimasukkan oleh pengguna benar. Aplikasi akan menerima pesan berupa “Sukses Login”. Dan jika salah akan menerima pesan “gagal login” dan meminta pengguna untuk memasukkan username dan password lagi.

Jika jaringan dari perangkat pengguna tidak sama dengan “Pi3-AP”. Maka tombol login tidak dapat ditekan. Namun, jika perangkat terhubung dengan jaringan Raspberry Pi. Maka tombol login dapat ditekan.

Tampilan remote Gambar 5.20 pada aplikasi iOS memiliki 5 tombol berbeda dengan rancangan tampilan remote pada perangkat pengguna. Hal ini karena tombol untuk men-*trigger* service RPi-Play hanya dapat dilakukan sekali saja. Yaitu saat pengguna pertama kali menekan tombol play. Karena kekurangan dari software RPi-Play yang sering terjadi adanya bug. Pada aplikasi ini tidak dapat memberhentikan service rpiplay pada Raspberry Pi. Jalan satu-satunya adalah saat pengguna menekan logout, Raspberry Pi akan melakukan reboot.

Gambar 5. 16 Tampilan Remote (iOS)

```
16  class RemoteController: UIViewController {
17      @IBOutlet weak var connectButton: UIButton!
18      @IBOutlet weak var playButton: UIButton!
19      @IBOutlet weak var pauseButton: UIButton!
20      @IBOutlet weak var logoutButton: UIButton!
21      var message: String = ""
22      let defaults = UserDefaults.standard
23      var isRecord = false
24      var status: String = ""
25      var isConnected = false
26      var clientIp: String = ""
27      let device = "IOS"
28      var client = TCPClient(address: "192.168.4.1", port: 5000)
29      override func viewDidLoad() {
30          super.viewDidLoad()
31      }
```

```
33      override func viewWillAppear(_ animated: Bool) {
34          //setiap mau bukaÍ
35          let name = defaults.string(forKey: "username") ?? ""
36          print(name)
37          if name == "" {
38              let storyBoard: UIStoryboard = UIStoryboard(name: "Main", bundle: nil)
39              let homeController = storyBoard.
40              instantiateViewController(withIdentifier: "HomeController") as! HomeContr
41              navigationController?.pushViewController(homeController, animated: true)
42          }else{
43              //self.helloLabel.text = "Hello " + name
44          }
45          clientIp = getWiFiAddress()!
46      }
48  @IBAction func connectButtonPressed(_ sender: Any) {
49      switch client.connect(timeout: 10){
50      case .success:
51          print("Success")
52      case .failure(let error):
53          print(error)
54      }
55      status = "0"
56      sendTCP(status)
57      sendTCP(device)
58      isConnected = true
59      checkConnection()
60  }
61  @IBAction func playButtonPressed(_ sender: Any) {
62      status = "1"
63      sendTCP(clientIp)
64      sendTCP(status)
65      isRecord = true
66      showToast(message: "Please open control center & start screen mirroring")
67      checkConnection()
68  }
69  @IBAction func pauseButtonPressed(_ sender: Any) {
70      status = "0"
71      sendTCP(clientIp)
72      sendTCP(status)
73      isRecord = false
74      showToast(message: "Please open control center & stop screen mirroring")
75      checkConnection()
76  }
```

```
@IBAction func logoutButtonPressed(_ sender: Any) {
    sendTCP(clientIp)
    status = "3"
    sendTCP(status)

    let storyBoard: UIStoryboard = UIStoryboard(name: "Main", bundle: nil)
    let homeController = storyBoard.
    instantiateViewController(withIdentifier: "HomeController") as! HomeController
    defaults.set("", forKey: "username")
    defaults.set("", forKey: "password")
    client.close()
    navigationController?.pushViewController(homeController, animated: true)
}

@IBAction func shutdownButtonPressed(_ sender: Any) {
    sendTCP(clientIp)
    status = "4"
    sendTCP(status)

    let storyBoard: UIStoryboard = UIStoryboard(name: "Main", bundle: nil)
    let homeController = storyBoard.
    instantiateViewController(withIdentifier: "HomeController") as! HomeController
    defaults.set("", forKey: "username")
    defaults.set("", forKey: "password")
    client.close()
    navigationController?.pushViewController(homeController, animated: true)
}
```

```
    func sendTCP(_ content: String) {
        let contentToSendUDP = content.data(using: String.Encoding.utf8)
        _ = client.send(data: contentToSendUDP!)
        sleep(5)
    }


    func receiveTCP() ->String{
        let data = client.read(1024*10)
        let text = String(bytes: data!, encoding: .utf8)!
        print(text)
        return text
    }
```

Kode 5. 23 Implementasi Class Remote (iOS)

Implementasi class Remote pada aplikasi iOS diubah penulis menjadi class RemoteController. Karena sifat aplikasi android dimana saat keluar dari aplikasi, aplikasi akan membuka tampilan pertama. Aplikasi iOS pada tugas akhir ini membutuhkan session. Session dibuat agar saat pengguna telah melakukan login tidak melakukan login kembali saat keluar dari aplikasi.

Berbeda dengan aplikasi pada Windows dan Android, setiap tombol pada aplikasi iOS hanya akan memeriksa apakah alamat ip pada aplikasi pengguna dan Raspberry Pi sama.

```
func checkConnection(){
    if isConnected == false {
        connectButton.isEnabled = true
        playButton.isEnabled = false
        pauseButton.isEnabled = false
        logoutButton.isEnabled = true
    }
    else if isConnected == true {
        connectButton.isEnabled = false
        if isRecord{
            playButton.isEnabled = false
            pauseButton.isEnabled = true
        }
        else{
            playButton.isEnabled = true
            pauseButton.isEnabled = false
        }
        logoutButton.isEnabled = true
    }
}
// Return IP address of WiFi interface (en0) as a String, or `nil`
func getWiFiAddress() -> String? {
    var address : String?

    // Get list of all interfaces on the local machine:
    var ifaddr : UnsafeMutablePointer<ifaddrs>?
    guard getifaddrs(&ifaddr) == 0 else { return nil }
    guard let firstAddr = ifaddr else { return nil }

    // For each interface ...
    for ifptr in sequence(first: firstAddr, next: { $0.pointee.ifa_next }) {
        let interface = ifptr.pointee

        // Check for IPv4 or IPv6 interface:
        let addrFamily = interface.ifa_addr.pointee.sa_family
        if addrFamily == UInt8(AF_INET) || addrFamily == UInt8(AF_INET6) {

            // Check interface name:
            let name = String(cString: interface.ifa_name)
            if  name == "en0" {

                // Convert interface address to a human readable string:
                var hostname = [CChar](repeating: 0, count: Int(NI_MAXHOST))
                getnameinfo(interface.ifa_addr, socklen_t(interface.ifa_addr.pointee.sa_len),
                            &hostname, socklen_t(hostname.count),
                            nil, socklen_t(0), NI_NUMERICHOST)
                address = String(cString: hostname)
            }
        }
    }
    freeifaddrs(ifaddr)
    return address
```

Kode 5. 24 Method Pendukung (iOS)

Pada Kode 5.24 aplikasi akan memerika apakah tombol connect sudah ditekan atau belum. Perintah checkConnection digunakan saat tombol connect ditekan. Seta perintah getWifiAddress digunakan untuk mendapatkan alamat ip dari perangkat.

5.2.5 Implementasi Testing

Metode pengujian yang akan dilakukan pada pengembangan sistem menggunakan metode *integration testing* dan *performance testing (black box testing)*. Berikut adalah daftar perangkat yang digunakan untuk melakukan testing dari sistem yang telah dibuat.

Tabel 5. 8 Spesifikasi Raspberry Pi

Tipe	Rasberry Pi 3 Model B V1.2
Processor	Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
Memory (RAM)	1GB LPDDR2 (900 MHz)
Sistem Operasi	Raspbian
GPU	Broadcom VideoCore IV @ 250 MHz

Tabel 5. 9 Spesifikasi Perangkat Windows

Tipe	ASUS ROG GL553VD
Processor	Intel® Core™ i7-7700HQ CPU @2.8GHz
Memory (RAM)	16 GB
Sistem Operasi	Windows 10 Education 64Bit
Graphic Card	Nvidia GeForce GTX 1050

Tabel 5. 10 Spesifikasi Perangkat Android

Tipe	Samsung J500G
Processor	Quad-core 1.2 GHz Cortex-A53
Memory (RAM)	8GB 1.5GB RAM
Sistem Operasi	Android 6.0.1
Graphic Card	Adreno 306

Tabel 5. 11 Spesifikasi Perangkat iOS

Tipe	iPhone 6s
Processor	64-bit Apple A9 - M9 motion coprocessor
Memory (RAM)	2 GB LPDDR4 RAM
Sistem Operasi	OS 13.2.3
Graphic Card	PowerVR GT7600 (hexa-core)

Tabel 5. 12 Spesifikasi Jammer

Nama	Portable Cell Phone Signal Blocker Jammer (CDMA / GSM / DCS / PHS / 3G)
Isolating Signal Bandwidth	CDMA:850-894 MHz GSM: 925-960 MHz DCS: 1805-1880 MHz PHS/CDMA1900: 1900-1980 MHz 3G: 2110-2170 MHz
Jangkauan	3-5 meter

5.2.5.1 Pengujian Fungional Aplikasi Raspberry Pi

Pengujian fungsional aplikasi raspberry pi dilakukan sesuai dengan gambar 4. 4 diagram alur aplikasi raspberry pi. Penjelasan dari masing-masing fungsional akan dijelaskan didalam tabel beserta penjelasannya.

Tabel 5. 13 Uji Fungsional Buat dan Kirim Token

No	Perintah	Ekspektasi	Hasil
1	Membuat token	Token berhasil dibuat	OK
2	Mengirim token	Token terkirim menggunakan protokol TCP	OK
3	Melakukan *hash* token	Hash token berhasil dibuat	OK

Pada tabel 5. 13, buat dan kirim token, diawali dengan mendapatkan token dengan method SecureRandom yang disediakan pada bahasa pemrograman Java. Token berhasil dikirim menggunakan protokol TCP ke Aplikasi Client .
Setelah token berhasil dikirim, aplikasi Raspberry pi berhasil melakukan dan menyimpan hash token menggunakan metode SHA512.

Tabel 5. 14 Uji Fungsional Menerima Status 0

No	Perintah	Ekspektasi	Hasil
1	Menerima status	Pesan diterima menggunakan protokol TCP	OK
2	Memeriksa token	Dapat membedakan perintah connect atau pause	OK
3	Menutup frame gambar	Frame gambar tertutup	OK

Pada tabel 5. 14. Saat aplikasi Raspberry pi mendapatkan status berupa 0. Aplikasi berhasil melakukan pengecekan terhadap ketersediaan token. Jika token bernilai null, maka aplikasi akan menganggap perintah tersebut sebagai perintah connect dengan arti aplikasi harus membuat token atau sama seperti pada poin pertama. Jika token sudah memiliki nilai, maka aplikasi akan

menganggap perintah tersebut sebagai tombol pause dengan arti aplikasi harus menutup frame gambar yang sedang terbuka.

Tabel 5. 15 Fungsional Menerima Status 1

No	Perintah	Ekspektasi	Hasil
1	Menerima status	Pesan terkirim menggunakan protokol TCP	OK
2	Menerima paket gambar	Pesan diterima menggunakan protokol UDP	OK
3	Membagi paket gambar	Terdapat 3 bagian paket, yang terdiri dari alamat ip, token dan base64 gambar	OK
4	Mencocokkan alamat ip	Perbandingan menghasilkan nilai benar atau salah	OK
5	Mencocokkan token	Perbandingan menghasilkan nilai benar atau salah	OK
6	Menampilkan frame	Frame terbuka dan memunculkan gambar	OK

Pada tabel 5. 15. Saat aplikasi Raspberry pi mendapat status berupa 1. Aplikasi akan menganggap perintah ini berupa perintah play dan mulai menerima paket gambar berupa data UDP dan membuka frame gambar. Paket gambar yang diterima dipotong menjadi 3 bagian. Saat alamat ip dan/atau token yang diterima tidak sama seperti milik pengguna saat melakukan autentifikasi. Maka gambar tidak akan ditampilkan di dalam frame dan tidak akan mengganggu jalannya sistem. Saat alamat ip dan token bernilai sama dengan pengguna saat melakukan autentifikasi. Maka gambar akan ditampilkan di dalam frame.

Tabel 5. 16 Fungsional Menerima Status 2

No	Perintah	Ekspektasi	Hasil
1	Menerima status	Pesan terkirim menggunakan protokol TCP	OK
2	Menghapus nilai token	Token bernilai null	OK
3	Menutup frame gambar	Frame gambar tertutup	OK

Pada tabel 5. 16. Saat aplikasi Raspberry pi mendapat status berupa 2. Aplikasi akan menganggap perintah ini berupa perintah disconnect dan menghapus nilai token yang telah disimpan. Serta akan menutup frame gambar jika frame sedang terbuka.

Tabel 5. 17 Fungsional Menerima Status 3

No	Perintah	Ekspektasi	Hasil
1	Menerima status	Pesan terkirim menggunakan protokol TCP	OK
2	Menghapus nilai token	Token bernilai null	OK
3	Menghapus nilai alamat ip	Alamat ip bernilai null	OK
4	Berada pada kondisi untuk menerima username dan password	Sedang dalam kondisi untuk menerima masukan TCP	OK

Pada tabel 5. 17. Saat aplikasi Raspberry pi mendapat status berupa 3. Aplikasi akan menggap perintah ini berupa perintah logout. Aplikasi akan menghapus nilai token jika nilai token tidak bernilai null. Selain menghapus nilai token, aplikasi akan menghapus alamat ip dari pengguna yang telah melakukan

autentifikasi. Setelah menghapus alamat ip dan token, aplikasi akan berada dalam kondisi untuk menunggu username dan password.

Tabel 5. 18 Fungsional Menerima Status 4

No	Perintah	Ekspektasi	Hasil
1	Menerima status	Pesan terkirim menggunakan protokol TCP	OK
2	Shutdown	Raspberry pi melakukan shutdown	OK

Pada tabel 5. 18. Saat aplikasi Raspberry pi mendapat status berupa 4. Aplikasi akan menganggap perintah ini berupa perintah shutdown. Aplikasi berhasil menjalankan perintah shutdown pada terminal Raspbian.

Maka dapat disimpulkan seperti pada tabel 5. 19 bahwa dari keenam poin fungsional aplikasi Raspberry pi telah berhasil dilakukan semua.

Tabel 5. 19 Hasil Uji Fungsional Aplikasi Raspberry Pi

No	Fungsi	Ekspektasi	Hasil
1	Buat dan Kirim Token	Token terkirim dan melakukan hash token	OK
2	Status = 0	Buat token atau Tutup Frame Gambar	OK
3	Status = 1	Menerima Gambar dan membuka frame gambar	OK
4	Status = 2	Tutup frame gambar dan hapus token	OK
5	Status = 3	Tutup frame gambar dan reset token dan alamat ip	OK
6	Status = 4	Shutdown Raspberry pi	OK

5.2.5.2 Pengujian Fungional Aplikasi *Client*

Pengujian fungsional aplikasi *client* dilakukan sesuai dengan gambar 4. 7 diagram alur aplikasi *client*. Penjelasan dari masing-masing fungsional akan dijelaskan didalam tabel beserta penjelasannya.

Tabel 5. 20 Fungsional Cek Jaringan Perangkat

No	Perintah	Ekspektasi	Hasil
1	Meminta izin akses lokasi	Muncul pop-up izin akses lokasi untuk aplikasi	OK
2	Mengambil nama wi-fi yang sedang terhubung	Nama wi-fi dapat dibaca dan dapat diperiksa	OK
3	Terhubung dengan Raspberry Pi	Form login dan tombol login aktif	OK
4	Tidak terhubung dengan Raspberry Pi	Form login dan tombol login tidak aktif	OK

Pada tabel 5. 20, cek jaringan perangkat, hal pertama yang dilakukan aplikasi adalah meminta izin dari pengguna untuk mengaktifkan izin mengakses lokasi untuk aplikasi yang dibuat. Langkah selanjutnya, aplikasi akan memeriksa apakah perangkat telah terhubung dengan wi-fi dari Raspberry pi. Saat perangkat telah terhubung dengan wi-fi Raspberry pi, aplikasi mengaktifkan form username, form password dan tombol login. Saat perangkat tidak terhubung dengan wi-fi dari Raspberry pi, aplikasi akan menonaktifkan form usename, form password dan tombol login.

Tabel 5. 21 Fungsional Tombol Connect

No	Perintah	Ekspektasi	Hasil
1	Mengirim status	Pesan terkirim menggunakan protokol TCP	OK
2	Menerima token	Token dapat diterima menggunakan protokol TCP	OK
3	Melakukan *hash* dari *plain text* token	Hash token berhasil dibuat	OK
4	Memeriksa ketersediaan token	Jika token tidak bernilai null. Maka tombol play, pause dan disconnect aktif.	OK

Pada tabel 5. 21, tombol connect, aplikasi client mengirimkan status berupa 0 dengan menggunakan protokol TCP. Saat status telah dikirimkan, aplikasi dalam kondisi untuk mendapatkan token. Token yang didapat dilakukan hash dengan metode SHA512. Hash token ini digunakan untuk potongan paket gambar. Selanjutnya, aplikasi memeriksa ketersediaan token. Jika token sudah tidak bernilai null. Maka tombol play, pause, dan disconnect aktif.

Tabel 5. 22 Fungsional Tombol Play

No	Perintah	Ekspektasi	Hasil
1	Mengirim status	Pesan terkirim menggunakan protokol TCP	OK
2	Meminta izin *screenshot*	Muncul pop-up izin screenshot untuk aplikasi	OK
3	Melakukan *screenshot*	Gambar dapat dihasilkan	OK

Pada tabel 5. 22, tombol play, aplikasi client mengirimkan status berupa 1 dengan menggunakan protokol TCP. Aplikasi pada Android akan meminta izin untuk melakukan screenshot dan untuk aplikasi pada Windows langsung melakukan screenshot. Saat izin tidak diberikan aplikasi tidak melakukan screenshot dan tidak mengirip paket gambar.

Tabel 5. 23 Fungsional Tombol Pause

No	Perintah	Ekspektasi	Hasil
1	Mengirim status	Pesan terkirim menggunakan protokol TCP	OK
2	Berhenti melakukan *screenshot*	Tidak melakukan *screenshot* dan tidak menghasilkan gambar	OK

Pada tabel 5. 23, tombol pause, aplikasi client mengirimkan status berupa 2 dengan menggunakan protokol TCP. Aplikasi berhenti melakukan screenshot dan berada pada status tidak melakukan screenshot.

Tabel 5. 24 Fungsional Tombol Logout

No	Perintah	Ekspektasi	Hasil
1	Mengirim status	Pesan terkirim menggunakan protokol TCP	OK
2	Mengubah token dan alamat ip menjadi null	Token dan alamat ip bernilai null	OK
3	Merubah tampilan remote menjadi tampilan login	Tampilan berubah menjadi tampilan login	OK

Pada tabel 5. 24, tombol logout, aplikasi client mengirimkan status berupa 3 dengan menggunakan protokol TCP. Aplikasi mengubah nilai token dan alamat ip menjadi null. Kemudian aplikasi mengubah tampilan menjadi tampilan login dan menunggu masukan username dan password.

Tabel 5. 25 Fungsional Tombol Shutdown

No	Perintah	Ekspektasi	Hasil
1	Mengirim status	Pesan terkirim menggunakan protokol TCP	OK
2	Mengubah token dan alamat ip menjadi null	Token dan alamat ip bernilai null	OK
3	Merubah tampilan remote menjadi tampilan login	Tampilan berubah menjadi tampilan login	OK

Pada tabel 5. 25, tombol shutdown, aplikasi client akan mengirimkan status berupa 4 menggunakan protokol TCP. Aplikasi mengubah nilai token dan alamat ip menjadi null. Kemudian aplikasi mengubah tampilan menjadi tampilan login dan menunggu masukan username dan password.

Tabel 5. 26 Fungsional Mengirim Gambar

No	Perintah	Ekspektasi	Hasil
1	Mengirim status	Pesan terkirim menggunakan protokol TCP	OK
2	Encoding gambar menggunakan Base64	Gambar dalam bentuk data Base64	OK
3	Memebuat paket UDP gambar	Paket UDP memiliki bagian alamat ip,	OK

		token dan base64 dari gambar	
4	Mengirim paket gambar	Paket gambar terkirim menggunakan protokol UDP	OK

Pada tabel 5. 26, mengirim gambar, aplikasi mengirim gambar setelah melakukan screenshot dan langsung mengubah gambar menjadi data berupa Base64, menjadikan 1 paket dengan alamat ip dan hash token. Lalu, Paket gambar dikirim menggunakan protokol UDP. Saat gambar memiliki ukuran gambar lebih dari 65kb gambar tidak dikirim dan tidak mengganggu jalannya sistem.

Maka dapat disimpulkan seperti pada tabel 5. 27 bahwa dari ketujuh poin fungsional aplikasi client telah berhasil dilakukan semua.

Tabel 5. 27 Hasil Uji Fungsional Aplikasi Client

No	Fungsional	Ekspektasi	Hasil
1	Cek Jaringan Perangkat	Wifi terdeteksi	OK
2	Tombol connect	Status = 0, menerima token dan melakukan hash token	OK
3	Tombol Play	Status = 1, screenshot	OK
4	Tombol Pause	Status = 2 dan berhenti melakukan screenshot	OK
5	Tombol Logout	Status = 3, mereset token dan membuka tampilan login	OK
6	Tombol Shutdown	Status = 4, mereset token dan membuka tampilan login	OK

7	Mengirim Gambar	Paket gambar terkirim dengan 3 bagian	OK

5.2.5.3 Pengujian Waktu Delay

Pengujian dilakukan dengaan output Raspberry Pi menggukan layar monitor resolusi 1024 x 768 dan kualitas yang berbeda untuk perangkat windows dan android. Perangkat iphone tidak ada manipulasi kualitas gambar

Tabel 5. 28 Hasil Pengujian Waktu Delay (Windows)

Perangkat:	**Windows**		
Kualitas Gambar	15 %	20%	25%
No	Delay (detik)		
1	1.31	1.2	0.97
2	1.21	1.3	1.27
3	1.33	1.7	1.14
4	1.27	1.2	1.07
5	1.33	1.43	1.04
Rata-rata	1.29	1.366	1.098

Tabel 5. 29 Hasil Pengujian Waktu Delay (Android)

Perangkat:	**Android**			
Kualitas Gambar	15 %	20%	25%	50%
No	Delay (detik)			
1	0.98	0.91	0.85	0.79
2	0.79	0.71	0.92	0.7
3	0.92	0.92	0.84	0.78
4	0.78	0.91	0.92	0.88
5	1.17	1.04	0.84	0.72
Rata-rata	0.928	0.898	0.874	0.774

Tabel 5. 30 Hasil Pengujian Waktu Delay (iOS)

Perangkat:	**iphone**
Kualitas Gambar	-
No	Delay
1	0.16 detik
2	0.21 detik
3	0.22 detik
4	0.19 detik
5	0.15 detik
Rata-rata	0.186 detik

5.2.5.4 Pengujian Multiuser pada Sistem.

Tabel 5. 31 Hasil Pengujian Multiuser pada Sistem

No	**Kondisi Raspberry Pi**	**Perintah Perangkat 2**	**Respon**
1	Telah terhubung dengan perangkat (tcp)	Login	Tidak ada respon
2		Connect	Tidak ada respon
3		Play	Tidak ada respon
4		Pause	Tidak ada respon
5		Disconnect	Tidak ada respon
6		Logout	Tidak ada respon
7		Shutdown	Tidak ada respon

5.2.5.5 Pengujian Hasil Fps

- Uji coba 1: Jenis perangkat keras dan port yang digunakan pada output display.

Tabel 5. 32 Hasil Uji Coba 1

Perangkat	**Port**	**Jenis Perangkat**		
		Windows	Android	iOS
Monitor	HDMI	5 fps	8-9 fps	55-60 fps

1024 x 768 60Hz				
Proyektor 1024 x 768	VGA	5 fps	8-9 fps	55-60 fps

- Uji Coba 2: Resolusi yang digunakan pada resolusi output display.

Tabel 5. 33 Hasil Uji Coba 2

Resolusi	**Jenis Perangkat**		
	Windows	Android	iOS
1024 x 768 60 Hz	5 fps	8-9 fps	55-60 fps

- Uji Coba 3: Pengujian hasil gambar yang didapat terhadap adanya jammer sinyal Wi-Fi.

Tabel 5. 34 Hasil Uji Coba 3

	Hasil fps		
Jarak Jammer – Raspberry pi	Perangkat Windows	Android	iOS
1 meter	0 fps	0 fps	0 fps
2 meter	0 fps	0 fps	0 fps
3 meter	2-4 fps	1-3 fps	0-2 fps
4 meter	5 fps	8 fps	55 fps

Pengujian hasil fps terhadap adanya jammer dilakukan dalam jarak total 5 meter. Jarak pada tabel menunjukan jarak jammer terhadap raspberry pi. Jika jarak menunjukkan 1 meter, maka jarak jammer terhadap perangkat adalah 4 meter.

5.2.5.6 Pengujian Jarak Jaringan

Tabel 5. 35 Hasil Pengujian Jarak Jaringan

No	**Jarak**	**Kondisi**		
		Windows	Android	iOS

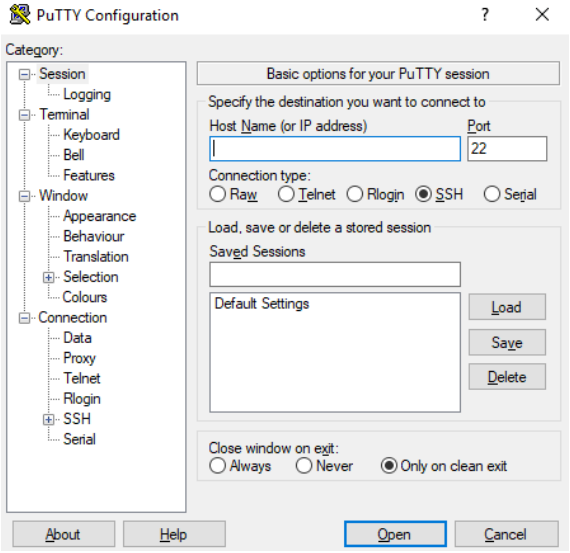
1	5 Meter	Terdeteksi	Terdeteksi	Terdeteksi
2	10 Meter	Terdeteksi	Terdeteksi	Terdeteksi
3	15 Meter	Tidak Terdeteksi	Tidak Terdeteksi	Tidak Terdeteksi

BAB VI
HASIL DAN PEMBAHASAN

Pada bagian ini akan menjelaskan mengenai hasil dan pembahasan dari pengembangan sistem wireless display yang telah dibuat dalam pengerjaan tugas akhir ini.

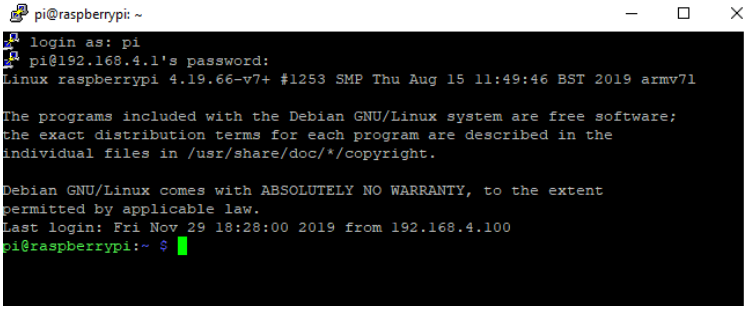
6.1 Hasil Implementasi SSH

Implementasi SSH menggunakan sistem operasi Windows dengan software Putty. SSH digunakan untuk menambahkan user pada database. Gambar 6.1 adalah bentuk tampilan dari Putty. Pada baris "Host Name" diisi dengan alamat ip dari Raspberry Pi, 192.168.4.1. Lalu isi port dengan port 22.

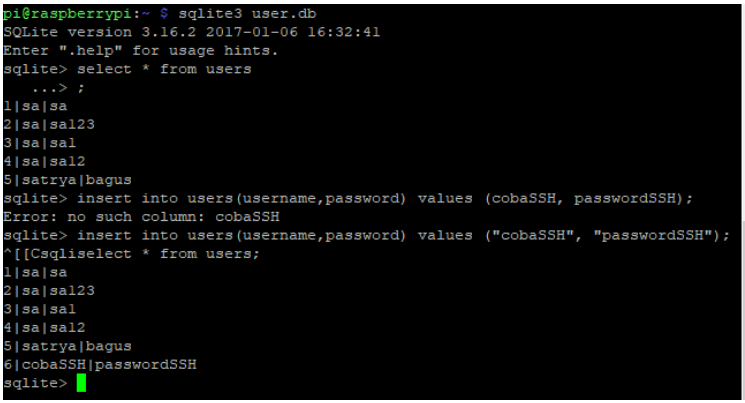


Gambar 6. 1 Tampilan Putty

Tampilan seperti pada Gambar 6.2 akan muncul. Tampilan ini menunjukan bahwa koneksi dengan Raspberry Pi berhasil. Berikutnya login dengan username dan password seperti pada bab perancangan. Jika berhasil melakukan login, akan muncul baris seperti pada tampilan terminal linux.



Gambar 6. 2 Login SSH Berhasil

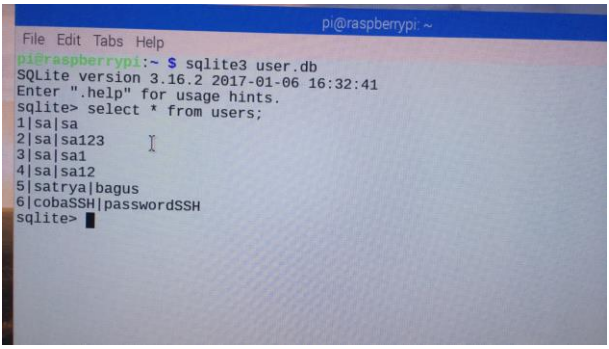


Gambar 6. 3 Perintah SQLite3

Detail perintah SQLite untuk menambahkan user pada Gambar 6.3 adalah sebagai berikut.

```
Sqlite3 user.db
INSERT INTO users(username, password) VALUES
(“cobaSSH”,”passwordSSH”)
```

Perintah tersebut digunakan untuk menambahkan user dengan username cobaSSH dan passwordSSH sebagai passwordnya. Untuk mengetahui apakah sudah masuk pada Raspberry Pi. Dapat dilakukan pengecekan dengan masuk pada sqlite3 langsung di Raspberry Pi seperti pada Gambar 6.4.



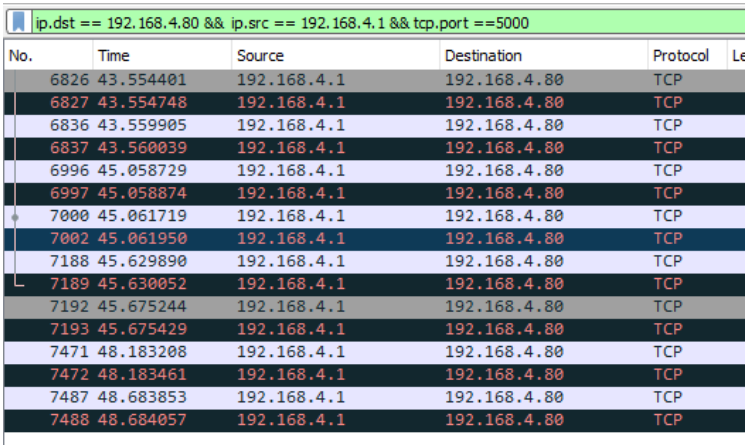
Gambar 6. 4 Tampilan Tabel Users

6.2 Implementasi Protokol

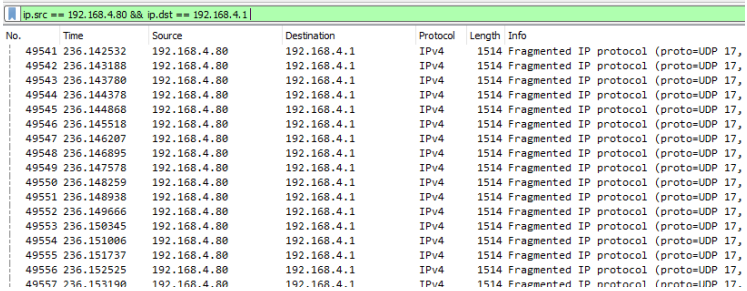
Untuk mengetahui hasil implementasi protokol yang digunakan pada tugas akhir digunakan tools cain & abel untuk menjadikan komputer berperan sebagai *man in the middle* dalam jalur yang akan di-*poison*. Lalu untuk mendapatkan packet yang sedang melintas, digunakan tools wireshark.

Sesuai dengan metodologi yang digunakan untuk protokol yang digunakan. Protokol untuk mengirim pesan adalah protokol TCP dan Protokol yang digunakan untuk pengiriman gambar adalah menggunakan port UDP.

Pengujian pertama dilakukan dengan device Android dan Raspberry Pi. Laptop windows berperan sebagai MITM pada jalur tersebut.



Gambar 6. 5 Tampilan Paket TCP Login (Android)

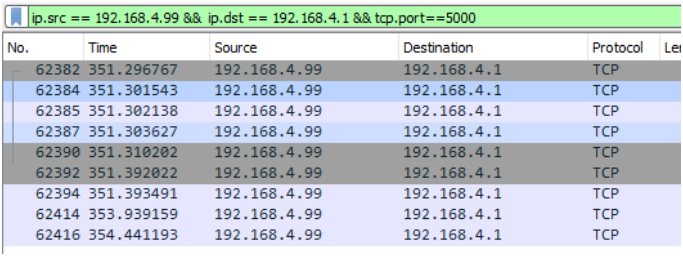


Gambar 6. 6 Tampilan Paket UDP Gambar (Android)

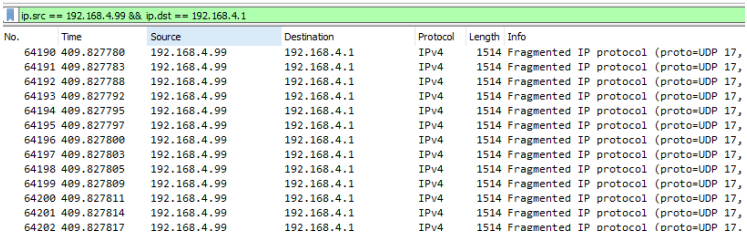
Pada Gambar 6.5 dengan menggunakan filter “ip.src == 192.168.4.83 && ip.dst == 192.168.4.1 && tcp.port == 5000” didapatkan paket dengan protokol TCP. Alamat ip sumber adalah alamat ip pengirim paket. Dan alamat ip tujuan adalah alamat ip dari penerima paket.

Pada Gambar 6.6 didapatkan pula bahwa gambar yang dikirim menggunakan protokol udp. Protokol pada wireshark menunjukan bahwa protokol yang digunakan adalah protokol ipv4. Penjelasan bahwa paket diterima menggunakan protokol udp adalah pada kolom info yang menjelaskan protokol = UDP.

Pengujian kedua adalah menggunakan perangkat Windows. Didapatkan hasil sama seperti hasil tangkapan paket pada perangkat Android.

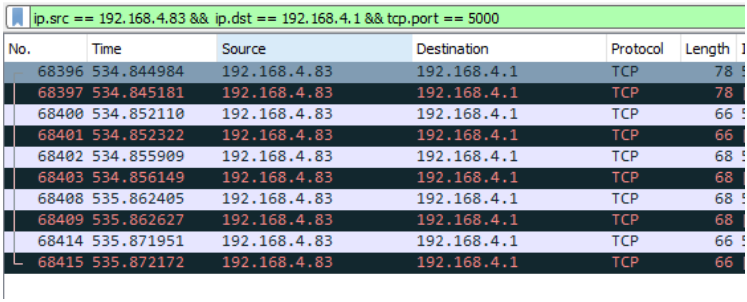


Gambar 6. 7 Tampilan Paket TCP Login (Windows)

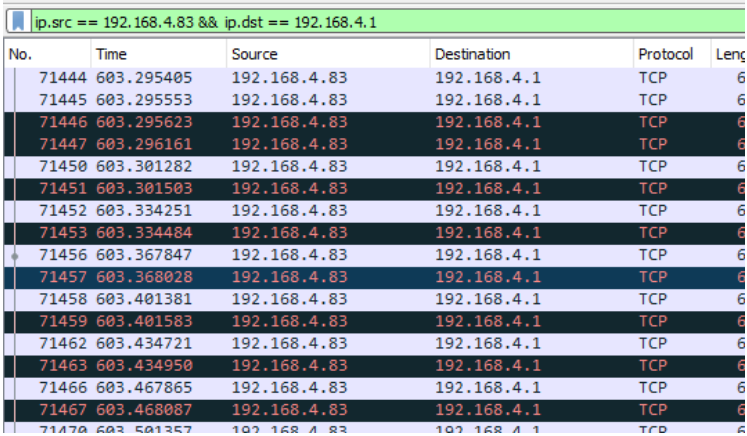


Gambar 6. 8 Tampilan Paket UDP Gambar (Windows)

Pengujian Ketiga adalah menggunakan perangkat iPhone. Didapatkan hasil login yang sama seperti pada perangkat lainnya. Bahwa pada saat login protokol yang digunakan adalah protokol TCP. Namun ada perpedaan saat screen-mirroring dilakukan. Protokol yang digunakan oleh Rpi-Play menggunakan TCP.



Gambar 6. 9 Tampilan Paket TCP Login (iPhone)



Gambar 6. 10 Tampilan Paket TCP Gambar (iPhone)

6.3 Performa Sistem

Berdasarkan implementasi testing yang dilakukan terhadap waktu delay. Dapat dikatakan bahwa sistem memiliki delay rata-rata 1 detik untuk perangkat Windows dan Android. Sedangkan pada perangkat iphone memiliki waktu delay 0.1-0.2 detik saja.

Kualitas gambar yang digunakan agar performa stabil pada windows adalah dengan kualitas sebesar 15%. Hal ini karena saat kualitas gambar diatur menggunakan 20%, ukuran byte gambar mendekati angka 65000 (batas ukuran UDP). Hal ini dapat menyebabkan gambar tidak dapat dikirim. Sedangkan

dengan ukuran 25% ukuran gambar mendekati batas ukuran UDP dan kadang melebihi batas tersebut.

Kualitas gambar yang digunakan agar performa stabil pada Android adalah dengan kualitas 20% - 25%. Saat kualitas gambar dinaikkan menjadi 50%, ukuran gambar mendekati batas ukuran UDP dan kadang melebihi batas tersebut. Sedangkan saat diturunkan menjadi 15%, kenaikan fps tidak terlalu signifikan.

Berdasarkan uji coba multiuser. Sistem dapat dikatakan reliable karena sistem hanya akan menerima nilai alamat ip dan token yang telah ditetapkan. Saat alamat ip yang lain dengan token yang sama mencoba untuk masuk kedalam sistem. Penyusup akan dihiraukan. Serta sistem menggunakan TCP dari awal login hingga pengguna melakukan logout.

Pada uji coba fps pertama (tabel 5. 32), Sistem tidak tergantung pada jenis perangkat keras yang terhubung dengan Raspberry Pi. Dari hasil tampilan yang dikeluarkan juga tidak memiliki perbedaan. Lalu pada uji coba fps kedua (tabel 5.33), dapat diketahui bahwa sistem tergantung pada resolusi layar raspberry pi. Pengaruh resolusi terhadap sistem adalah semakin kecil resolusi layar perangkat keras keluaran Raspberry Pi. Maka semakin kecil pula ukuran data yang akan dikirimkan. Pengujian dilakukan hanya pada resolusi 1024 x 768 saja. Selanjutnya, berdasarkan uji coba fps ketiga (tabel 5.34), dapat diketahui bahwa saat adanya perusak sinyal disekitar sistem. Sistem akan terdisrupsi pada jarak 1 hingga 3 meter terhadap *jammer* dan saat lebih dari 3-meter sistem tidak mengalami gangguan.

Halaman ini sengaja dikosongkan

BAB VII
KESIMPULAN DAN SARAN

Pada bagian ini akan menjelaskan mengenai kesimpulan dari pengerjaan tugas akhir ini dan saran untuk pengembangan perangkat lunak selanjutnya yang relevan.

7.1. Kesimpulan

Berikut merupakan kesimpulan yang dapat diambil dari pengembangan sistem wireless display yang telah dilakukan.

1. Aplikasi yang dikembangkan dapat menggunakan sistem tanpa adanya gangguan dari aplikasi atau pengguna lain
2. Sistem menghasilkan *scaling* resolusi yang tepat dengan membandingkan resolusi tiap layar.
3. Aplikasi yang dikembangkan memiliki fps dengan rata-rata 5 fps untuk windows, 8 fps untuk android dan 55 fps untuk iPhone.
4. Aplikasi yang dikembangka memiliki rata-rata delay sebesar 1 detik.
5. Besar perbedaan resolusi akan menimbulkan penurunan kualitas gambar.
6. Semakin kecil resolusi yang digunakan. Hasil fps yang didapat akan semakin tinggi.
7. Protokol untuk message control menggunakan protokol TCP dan transfer gambar menggunakan protokol UDP.
8. Untuk penambahan user baru untuk sistem. Hanya pengguna Raspberry Pi saja yang dapat menambahkan.

7.2. Saran

Berikut merupakan saran untuk pengembangan lebih lanjut yang dapat dilakukan dalam menyempurnakan pengembangan sistem.

1. Pengembangan perangkat lunak untuk Android dan Windows dapat mengimplementasikan protokol RTP dan dapat menggunakan audio.
2. Meningkatkan fps yang dihasilkan.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] A. Supriyanto, "Analisis Kelemahan Keamanan pada Jaringan Wireless," *J. Teknol. Infromasi Din.*, vol. 11, no. 1, pp. 38–46, 2006.

[2] "Pengguna Smartphone di Indonesia 2016-2019 | Databoks." [Online]. Available: https://databoks.katadata.co.id/datapublish/2016/08/08/pengguna-smartphone-di-indonesia-2016-2019. [Accessed: 28-Mar-2019].

[3] "Consumer Barometer - TRENDING." [Online]. Available: https://www.consumerbarometer.com/en/trending/?countryCode=ID&category=TRN-NOFILTER-ALL. [Accessed: 28-Mar-2019].

[4] D. Rusmiyati, Ida, "Penggunaan Multimedia Dalam Pembelajaran Bahasa Sastra Indonesia di SMPN 2 Bawen Kabupaten Semarang," *J. Teknol. Pendidik. Dan Pembelajaran*, vol. 2, no. 2, pp. 1–14, 2014.

[5] B. P. A, M. S. Qirom, and D. Hermanto, "Automatisasi Smart Home Dengan Raspberry Pi Dan Smartphone Android," *Konf. Nas. Ilmu Komput.*, vol. 1, no. December 2014, pp. 1–5, 2014.

[6] Y. Fauzan and B. Kurniawan, "Rancang Bangun Perangkat Wireless untuk Projector Konvensional Design of Wireless Devices for Conventional Projector," vol. 4, no. 1, pp. 43–51, 2016.

[7] A. I. Yusuf and B. T. Penelitian, "Rancang Bangun Sistem Printer Tanpa Kabel Berbasis Bluetooth dan WIFI Communication System Design Wireless Printer Based on Bluetooth and WI-FI," *Telekontran, Vol. 2, No. 1, Novemb. 2014*, vol. 2, no. 1, pp. 54–60, 2014.

[8] D. K. Salim, J. Andjawirawan, and L. P. Dewi, "Penerapan Screen Mirroring Android Pada Projector Menggunakan Raspberry Pi," *J. Infra Petra*, 2019.

[9] H. A. Harahap, "Rancan Bangun Jaringan Peer to Peer Webcam dengan Menggunakan Protokol JXTA," 2012.

[10] C. H. Ding, S. Nutanong, and R. Buyya, "Peer-to-Peer

Networks for Content Sharing," *Peer-to-Peer Comput.*, pp. 28–65, 2011.

[11] S. Raspberry, P. I. Menggunakan, and H. Protocol, "Implementasi Video Streaming Lalu Lintas Kendaraan Dengan Implement Video Streaming Traffic Using Raspberry Pi With," vol. 5, no. 5, pp. 629–634, 2018.

[12] H. Jusuf, "Penggunaan Secure Shell (SSH) Sebagai Sistem Komunikasi Aman Pada Web Ujian Online," *Bina Insa. Ict J.*, vol. 2, no. 2, pp. 75–84, 2015.

[13] S. Sukaridhoto, "Buku Jaringan Komputer I," 2014.

[14] Y. Mardiana and J. Sahputra, "Analisa Performansi Protokol TCP , UDP dan SCTP," *J. Media Infotama*, vol. 13, no. 2, pp. 73–84, 2017.

[15] D. A. N. Presentasi, A. Setiyadi, and T. Harihayati, "Vol.13 No. 2," vol. 13, no. 2, pp. 221–226.

[16] W. E. Winanto, "APLIKASI KEAMANAN EMAIL DATA PRODUKSI PT KUNYUN GRAVURE INDUSTRIES INDONESIA DENGAN RC4 DAN BASE64," vol. 1, no. 1, pp. 303–308, 2018.

[17] P. Algoritma, G. Rc, D. A. N. Base, and P. S. K. E-commerce, "Penerapan Algoritma Gabungan RC4 dan Base64 pada Sistem Keamanan E- Commerce (Application of Joint RC4 and BASE64 Algorithm for E-Commerce Security System)," no. June 2012, 2016.

[18] A. Rahmatulloh, H. Sulastri, and R. Nugroho, "Keamanan RESTful Web Service Menggunakan JSON Web Token (JWT) HMAC SHA-512," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 7, no. 2, 2018.

[19] "FD-/RPiPlay: An open-source AirPlay mirroring server for the Raspberry Pi. Supports iOS 9 and up." [Online]. Available: https://github.com/FD-/RPiPlay. [Accessed: 17-Jan-2020].

[20] "KqSMea8/AirplayServer: Android server for Airplay2." [Online]. Available: https://github.com/KqSMea8/AirplayServer. [Accessed: 17-Jan-2020].

BIODATA PENULIS

Penulis bernama Satrya Bagus Nurawan, lahir di Sidoarjo pada tanggal 19 Januari 1998, merupakan anak ketiga dari tiga bersaudara. Penulis telah menempuh jenjang pendidikan formal di beberapa sekolah, yaitu : SD Negeri Pucang 1 Sidoarjo (2006 - 2011), SMP Negeri 3 Sidoarjo (2011 - 2014), dan SMA Negeri 1 Sidoarjo (2014 - 2016). Penulis melanjutkan pendidikan sarjana di Departemen Sistem Informasi Fakultas Teknologi Informasi dan Komunikasi (FTIK) Institut Teknologi Sepuluh Nopember (ITS) pada tahun 2016 yang terdaftar sebagai mahasiswa dengan NRP 05211640000079.

Selama menjadi mahasiswa, penulis aktif mengikuti organisasi kemahasiswaan, seperti BEM FTIF pada tahun 2017-2018, BEM FTIK pada tahun 2019, serta kegiatan kepanitiaan, seperti Information Systems Expo 2017, Information Systems Expo 2018, Technoprenuership 2018, dan kegiatan lainnya. Penulis juga pernah meraih penghargaan pada kompetisi karya tulis bidang MIPA yang berskala nasional, seperti LKTIN 2019 di Yogyakarta. Selain organisasi dan kepanitiaan, penulis menjadi asisten praktikum mata kuliah “Desain Manajemen Jaringan Komputer” selama satu semester pada tahun 2019.

Pada Juli - Agustus 2019, penulis melaksanakan kegiatan magang di PT GMF Aero Asia Tbk sebagai *System Analyst* pada Unit TO. Penulis dapat dihubungi melalui email satrya16@mhs.is.its.ac.id.

Halaman ini sengaja dikosongkan