TUGAS AKHIR – KI141502

RANCANG BANGUN ALAT BANTU UJI KEAMANAN MAKANAN

MUHAMMAD RUSLAN HAFIZ SYAUQI
NRP 5111100131

Dosen Pembimbing I
Daniel Oranova Siahaan, S.Kom., M.Sc., P.D.Eng.

Dosen Pembimbing II
Ridho Rahman Hariadi, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2015

TUGAS AKHIR – KI141502

DESIGN AND CONSTRUCTION OF FOOD SECURITY TEST TOOLS

MUHAMMAD RUSLAN HAFIZ SYAUQI
NRP 5111100131

Dosen Pembimbing I
Daniel Oranova Siahaan, S.Kom., M.Sc., P.D.Eng.

Dosen Pembimbing II
Ridho Rahman Hariadi, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2015

LEMBAR PENGESAHAN

RANCANG BANGUN ALAT BANTU UJI KEAMANAN MAKANAN

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Rekayasa Perangkat Lunak
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
MUHAMMAD RUSLAN HAFIZ SYAUQI
NRP : 5111 100 131

Disetujui oleh Dosen Pembimbing Tugas Akhir:

DANIEL ORANOVA SIAHAAN, S.Kom., M.Sc., P.D.Eng
NIP: 197411232006041001 ..... (pembimbing 1)

RIDHO RAHMAN HARIADI, S.Kom, M.Sc.
NIP: 198702132014041001 ..... (pembimbing 2)

KEMENTERIAN RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI INSTITUT TEKNOLOGI SEPULUH NOPEMBER FAKULTAS TEKNOLOGI INFORMASI JURUSAN TEKNIK INFORMATIKA

SURABAYA
JULI, 2015

RANCANG BANGUN ALAT BANTU UJI KEAMANAN MAKANAN

Nama Mahasiswa	**: Muhammad Ruslan Hafiz Syauqi**
NRP	**: 5111100131**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Daniel Oranova Siahaan, S.Kom., M.Sc.,P.D.Eng.**
Dosen Pembimbing 2	**: Ridho Rahman Hariadi, S.Kom, M.Sc**

Abstrak

Penggunaan zat kimia berbahaya dalam makanan dapat membahayakn kesehatan konsumen baik dalam jangka pendek maupun jangka panjang. Perubahan tekstur, bentuk maupun warna yang lebih menarik adalah faktor yang mendorong pedagang untuk menggunakan zat kimia ini. Oleh karenanya diperlukan pengawasan terhadap kandungan nutrisi dari produk makanan yang beredar di pasaran.

Food Security Test Kit adalah seperangkat alat untuk menentukan kadar sejumlah zat kimia berbahaya yang terkandung dalam makanan. Peralatan ini terdiri dari reagen yang memberikan perubahan warna dari sampel untuk jenis zat tertentu. Perubahan warna lalu dibandingkan dengan warna referensi yang menentukan kadar dari zat kimia yang terkandung dari sampel tersebut. Proses ini dilakukan secara konvensional yaitu melihat dan membandingkan warna hasil reaksi sampel dengan warna referensi.

Dapat terjadi permasalahan dimana kondisi orang yang melakukan uji coba tidak optimal. Kesalahan dalam membandingkan warna akibat human error dapat terjadi. Oleh karenanya dibutuhkan sebuah alat bantu atau sistem yang dapat mendukung proses pengujian makanan ini.

Pada tugas akhir ini akan dibangun sebuah alat bantu uji keamanan makanan yang mendukung prosedur dari Food

Security Kit. Alat atau sistem yang dibangun terdiri dari komputer kecil untuk melakukan komputasi dari data yang diambil dari sampel, kamera untuk mengambil gambar sampel, lampu untuk mengatur pencahayaan, layar untuk menampilkan antarmuka dengan pengguna dan wadah untuk menampung semua komponen yang telah disebutkan. Desain dari wadah dan posisi lampu akan dibuat sedemikian rupa untuk mengatur variabel cahaya yang tetap.

Sistem akan memiliki aplikasi berbasis desktop yang ditanamkan ke dalam komputer kecil. Aplikasi ini akan menampilkan antarmuka dengan pengguna, menentukan kadar dari zat kimia pada sampel dan mengatur daftar log dari hasil uji makanan yang telah dilakukan. Hasil dari pengujian sistem menunjukkan rata-rata akurasi sebesar 89.047% untuk jenis uji Boraks, Rhodamin B, Formalin, Merkuri dan Methanyl Yellow.

Kata kunci: Aplikasi Desktop , Food Security Kit, Komputer Kecil, Uji Keamanan Makanan

DESIGN AND CONSTRUCTION OF FOOD SECURITY TEST TOOLS

Student's Name	**: Muhammad Ruslan Hafiz Syauqi**
Student's ID	**: 5111100131**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Daniel Oranova Siahaan, S.Kom., M.Sc.,P.D.Eng.**
Second Advisor	**: Ridho Rahman Hariadi, S.Kom, M.Sc.**

Abstract

.

The use of hazardous chemicals in food can be harmful to the health of consumers in both the short and long term. Changes in texture, shape and color to be more attractive are the factors that encourage traders to use these chemicals. Therefore, the necessity in supervision over the nutritional content of food products on the market.

Food Security Test Kit is a set of tools to determine the levels of a number of hazardous chemicals contained in food. This equipment consists of reagents that provide color change of samples for certain types of substances. Discoloration then compared to a reference color that determines the levels of the chemical substances from the sample. This process is done conventionally by viewing and comparing the color of the reaction of the sample with the reference color.

Problems may occur where the condition of the person doing the test is not optimal. Errors in comparing color may occur due to human error. Therefore we need a tool or system that can support the process of testing these foods.

In this final project will be built a tool that supports the food safety testing procedures of the Food Security Kit. Device or system which consists of a small computer built to perform computation on the data taken from the sample, a camera to

take pictures of the sample, the lights to adjust the lighting, the screen to display the user interface and the container to accommodate all components that have been mentioned. The design of the container and position the lamp will be made in such a way to set the variable light remains.

The system will have a desktop-based application that is implanted into a small computer. This application will display the interface with the user, determining the levels of chemicals in the sample and adjust the log list of test results of food that has been done. Results of the testing system showed an average of 89 047% accuracy for Borax, Rhodamine B, Formalin, Mercury and Methanyl Yellow test type.

Keywords: Desktop Application, Food Safety Test, Food Security Kit, Small Computer

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **"RANCANG BANGUN ALAT BANTU UJI KEAMANAN MAKANAN".** Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang sangat baik bagi penulis. Dengan pengerjaan Tugas Akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS. Dengan Tugas Akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari. Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT dan Nabi Muhammad SAW.
2. Kedua orang tua penulis Herman Kadir dan Yuli Mulyanti yang tiada henti-hentinya mencurahkan kasih sayang, perhatian dan doa kepada penulis selama ini.
3. Ketiga saudara kandung penulis, Dhia Zahra Fauziyah, M Irfan Muthohari dan Muhammad Arif F. yang telah memberikan dukungan dan doa kepada penulis dalam menuntut ilmu hingga detik ini.
4. Bapak Daniel Oranova S., S.Kom., M.Sc. ,P.D.Eng. selaku pembimbing I yang telah membantu, membimbing, dan memotivasi penulis dalam menyelesaikan Tugas Akhir ini dengan sabar.
5. Bapak Ridho Rahman H., S.Kom, M.Sc. selaku pembimbing II yang juga telah membantu, membimbing, dan memotivasi kepada penulis dalam mengerjakan Tugas Akhir ini.
6. Ibu Dr. Eng. Nanik Suciati, S.Kom., M.Kom. selaku Kepala Jurusan Teknik Informatika ITS, Bapak Radityo Anggoro,

S.Kom.,M.Sc. selaku koordinator TA, dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.

7. Keluarga besar Laboratorium RPL, Mas Aldy, Mas Iqbal, Mas Yusuf, Mas Najib, Mbak Ami, Mbak Weny, Mbak Yati, Mas Upik, Tommy, Hawari, Ghani, Habibi, Mbak Ervina, Mbak Yenita, Mbak Fadhilah, Mas Gregory, Amanda, Anjar, Ria, Sasa, Dala, DS, Dery, Reva, Ibnu, Rere, Didit, Gilang, Aranda.
8. Teman-teman seperjuangan RMK RPL Andri, Ujik, Hilman, Wati, Galih, Bustan, Jay, Supri,Farhan.
9. Teman-teman satu kontrakan yang tingal bersama penulis selama 3 tahun Rahman, Tommy, Toto, Risal, Bestama, Faris, Punggi, Dapik, Andrie, Yunus, Tev.
10. Teman-teman yang bersedia meluangkan waktunya untuk membantu penulis dalam menyelesaikan masalah Samhid, Andra, Hayam, Hasfi dan teman –teman seperjuangan Tugas Akhir lainnya
11. Angkatan 2011 yang yang telah membantu, berbagi ilmu, menjaga kebersamaan, dan memberi motivasi kepada penulis, kakak-kakak angkatan 2009 dan 2010 serta adik-adik angkatan 2012 dan 2013 yang membuat penulis untuk selalu belajar.
12. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, Januari 2015

Muhammad Ruslan Hafiz Syauqi

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

1.1 Latar Belakang

Penggunaan bahan tambahan pada makanan memerlukan pengawasan yang intensif. Diperlukan adanya pengawasan terhadap penggunaan bahan tambahan yang melebihi batas ataupun mengandung zat kimia yang tidak untuk dikonsumsi. Zat kimia seperti *Rhodamin B*, Boraks, Formalin, *Methanyl Yellow* dan Merkuri dapat meningkatkan resiko terhadap penyakit bahkan langsung menyebabkan konsumen keracunan makanan.

Demi menjaga keamanan pangan BPOM(Badan Pengawasan Obat dan Makanan) setiap tahunnya mengadakan pengawasan terhadap produk pangan yang beredar di masyarakat terutama di sekitar sekolah. Pengawasan ini berupa uji coba terhadap sampel produk untuk menentukan kadar zat kimia yang digunakan. Pada tahun 2013 BPOM mengambil sampel dari 1.601 kantin atau penjaja makanan di sekitar sekolah sehingga didapatkan 15.917 sampel yang menjadi bahan uji coba. Didapatkan 12.859 sampel (80,79%) sampel makanan yang memenuhi syarat dan 3.058 sampel (19,21%) yang tidak memenuhi syarat [1]. Pengawasan seperti ini dibutuhkan untuk terus mengontrol keamanan produk pangan yang beredar di masyarakat.

Pengujian sampel makanan ini membutuhkan waktu yang bervariasi tergantung dari jenis kadar zat kimia yang ingin ditemukan ataupun alat yang digunakan. Contohnya pada uji cairan Formalin dibutuhkan waktu 1 menit hingga cairan bereaksi terhadap sampel. Hal ini menjadi masalah ketika banyaknya jumlah sampel yang akan diuji melebihi kapasistas atau tenaga dari penguji sampel makanan tersebut. Resiko *human error* yang disebabkan kondisi penguji yang tidak optimal atau dalam keadaan lelah dapat mengakibatkan kesalahan dalam pencocokan hasil dari uji sampel.

Terdapat sebuah alat yang dapat menjaga kualitas hasil uji sampel makanan. *RVLM(Royal Vial Lab Multireader)* adalah instrumen untuk menganalisa kandungan mikroorganisme pada makanan secara kuantitatif [2]. Alat ini dapat menghasilkan keluaran dengan akurasi hingga 100% dan datanya tersimpan dalam sistem. Namun penggunaan alat ini sangat mahal dan membutuhkan biaya yang besar. Sehingga dibutuhkan alat alternatif yang lebih murah. Selain itu alat ini dibuat oleh sebuah perusahaan yang berasal dari jerman yang bertolak belakang dengan arti kemandirian bangsa Indonesia dimana seluruh aspek kehidupan bangsa mampu dikelola secara mandiri oleh pemerintah dan seluruh komponen bangsa tanpa campur pihak asing [3]. Dengan kualitas dari sumber daya manusia Indonesia sekarang ini seharusnya dapat membuat alat alternatif yang lebih murah.

Dalam tugas akhir ini akan dibuat sebuah alat yang dapat menjadi solusi alternatif dalam proses uji coba zat kimia pada makanan. Dengan menggunakan kamera sebagai input gambar yang terhubung dengan komputer kecil yang menyimpan program untuk mengolah hasil dari uji coba. Warna dari hasil uji coba yang tertangkap oleh gambar akan dicocokan dengan daftar warna referensi dari uji coba yang sesuai. Setiap warna pada daftar warna referensi menunjukkan prediksi kadar zat kimia yang terkandung. Hasil prediksi kadar zat kimia dari sampel adalah hasil dari pencocokan dengan daftar warna referensi**.** Diharapkan alat ini dapat menghasilkan perbandingan yang akurat sama dengan uji coba zat kimia secara konvensional.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini dapat dipaparkan sebagai berikut.

1. Bagaimana desain alat yang akan dibuat sebagai wadah untuk kamera, lampu, sumber listrik dan komputer kecil?
2. Bagaimana kalibrasi antara lampu dan kamera untuk menghasilkan warna gambar yang natural ?

3. Bagaimana membuat aplikasi pada komputer kecil yang dapat mendeteksi komposisi warna RGB *(Red,Green,Blue)* dari gambar yang diambil ?
4. Bagaimana aplikasi dapat menentukan kelas warna referensi dari warna hasil reaksi sampel ?
5. Bagaimana aplikasi dapat memberikan respon ketika terdapat request untuk mengirim *log* dan menghapus *log* dari hasil uji dalam sistem ?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan yaitu Jenis uji yang dapat diproses oleh sistem hanya uji keamanan makanan yang termasuk dalam parameter uji dari Food Security Kit yang disediakan oleh PT. Sitoho Lamsukses , diantaranya uji Boraks, Formalin, Merkuri , *Methanyl Yellow*, dan *Rhodamin B*.

1.4 Tujuan

Tujuan dari pembuatan alat ini antara lain membuat sebuah produk alat bantu uji coba keamanan makanan yang dapat digunakan.

1.5 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Studi literatur

   Pada tahap ini dilakukan pencarian, pengumpulan, pembelajaran dan pemahaman informasi dan literatur yang diperlukan untuk membangun sistem. Literatur yang digunakan adalah sebagai berikut:

   a. Raspberry Pi
   b. Kamera Raspberry Pi
   c. GPIO(*Genereal Process Input Output*)

d. *Library* Pi4J
e. Food Security Kit
f. Klasifikasi KNN(*K Nearest Neighbor*)
g. Java *Socket*

2. Analisis dan perancangan sistem

   Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap ini mendefinisikan alur dari implementasi. Langkah-langkah yang dikerjakan juga didefinisikan pada tahap ini. Pada tahapan ini dibuat *prototype* sistem, yang merupakan rancangan dasar dari sistem yang akan dibuat. Serta dilakukan desain suatu sistem dan desain proses-proses yang ada.

3. Implementasi

   Implementasi merupakan tahap membangun rancangan program yang telah dibuat. Pada tahapan ini merealisasikan apa yang terdapat pada tahapan sebelumnya, sehingga menjadi sebuah program yang sesuai dengan apa yang telah direncanakan.

4. Pengujian dan evaluasi

   Pada tahapan ini dilakukan uji coba pada data yang telah dikumpulkan. Pengujian dan evaluasi akan dilakukan dengan menggunakan bahasa Java. Tahapan ini dimaksudkan untuk mengevaluasi kesesuaian data dan program serta mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

5. Penyusunan buku Tugas Akhir.

   Pada tahapan ini disusun buku yang memuat dokumentasi mengenai pembuatan serta hasil dari implementasi sistem yang telah dibuat.

1.6 Sistematika Penulisan Laporan Tugas Akhir

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu permasalahan, batasan masalah, metodo*log*i yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

Bab III Analisis dan Perancangan

Bab ini membahas mengenai perancangan perangkat lunak. Perancangan perangkat lunak meliputi perancangan data, arsitektur, prosesdan perancangan antarmuka pada kakas.

Bab IV Implementasi

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa *code* yang digunakan untuk proses implementasi.

Bab V Uji Coba Dan Evaluasi

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

Bab VI Kesimpulan Dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan sistem ke depannya.

Daftar Pustaka

Merupakan daftar referensi yang digunakan dalam pembuatan Tugas Akhir.

Lampiran

Merupakan bab tambahan yang berisi kode–kode sumber yang penting pada aplikasi, panduan pengguna dan panduan pengembang.

BAB II
TINJAUAN PUSTAKA

Pada bab ini dibahas mengenai dasar teori dan literatur yang menjadi dasar pembuatan Tugas Akhir. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap sistem yang dibuat dan berguna sebagai penunjang dalam pengembangan.

2.1 Komputer kecil Raspberry Pi

Raspberry Pi adalah komputer dengan harga yang sangat rendah seukuran kartu kredit yang berjalan dengan sistem operasi Linux dan sistem operasi ringan lainnnya yang dapat berjalan di prosesor ARM [4]. Komponen utama dari Raspberry Pi tertanam pada satu *motherboard* termasuk RAM dengan memori sebesar 512 MB. Raspberry Pi banyak digunakan dalam proyek yang membutuhkan mini *Server* untuk berbagai macam servis [5]. Hal ini dikarenakan Raspberry Pi memiliki keunggulan diantaranya

- Konsumsi Tenaga
  Raspberry Pi hanya membuthkan tenaga 5 hingga 7 watt atau sepersepuluh dari komputer desktop biasa. Karena *Server* berjalan siang dan malam penghematan tenaga listrik sangatlah diperlukan.

- Tidak ada bagian yang bergerak
  Raspberry Pi menggunakan SD(*Secure Digital*) *card* sebagai tempat penyimpanan data yang cepat dan tanpa ada bagian yang bergerak.

- Ukuran Kecil
  Raspberry Pi dapat digenggam dengan tangan. Sehingga Pi dapat diintegrasikan dengan berbagai alat lainnya

- Tanpa Suara
  Raspberry Pi tidak membuat suara sedikitpun

- Lampu Status
  Terdapat lampu status pada *motherboard* yang menandakan kondisi dari aktifitas NIC, disk I/O, status tenaga, dll

- Kapabilitas Ekspansi
  Tterdapat banyak alat tambahan yang dapat dihubungkan dengan Pi karena memiliki 2 port USB dan I/O (GPIO) ke kamera

- Input HDMI
  Raspberry Pi memiliki *port* HDMI untuk menampilkan resolusi hingga 1920x1200

- Terjangkau
  Spesifikasi yang diberikan oleh Raspberry Pi ditawarkan dengan harga yang sangat terjangkau

- Komunitas
  Raspberry Pi memiliki di dukung oleh berbagai macam komunitas. Terdapat banyak forum yang menyediakan bantuan untuk mengatasi masalah pada *hardware* atau *GNU/Linux Software* yang berjalan pada Pi

- Kemampuan *Overclocking*
  Raspberry Pi dapat melakukan *overclocking* namun terdapat beberapa resiko terkait dengan masalah performa.

2.2 Kamera Raspberry Pi

Modul kamera pada Raspberry Pi dapat digunakan untuk mengambil gambar ataupun video dengan kualitas HD*(High-Definition*). Modul ini memiliki *fixed-focus* berukuran 5 *megapixel* yang mendukung mode video 1080p30, 720p60 dan VGA90. Modul kamera ini terpasang dengan pita kabel sepanjang 15 cm yang dapat dihubungkan dengan *port* CSI(*Camera Serial Interface*) pada Raspberry Pi. Penggunaan modul kamera ini dapat

diimplementasikan dengan perintah melalui *terminal* ataupun *library* Picamera Python.

2.3 Raspberry Pi *GPIO(General Process Input Output)*

GPIO pada Raspberry Pi direpresentasikan dengan sejumlah pin yang tertanam pada *board* utama. Setiap pin ini dapat diprogram untuk saling berinteraksi. Input dapat berupa sensor atau sinyal dari perangkat lain. *Output* dari pin dapat digunakan untuk melakukan berbagai hal seperti, menyalakan lampu LED (*Light Emitting Diode*) untuk mengirimkan sinyal atau data ke perangkat lain. Setiap pin memiliki kondisi *on* atau *off*, atau pada terminologi komputer *high* atau *low*. Ketika pin memiliki kondisi high output memiliki tegangan sebesar 3.3 volt, ketika pada kondisi *low* maka mati.

Pin GPIO dipetakan dengan penomoran angka. Terdapat referensi penomoran pin tersendiri untuk setiap jenis *board*. Secara fisik biasanya penomoran diawali dari nomor 1 pada pin yang terletak di paling kiri bawah(posisi dekat SD *card*) .

2.4 *Library Pi4J*

Library Pi4J ditujukan untuk memberikan API(*Application Programming Interface*) I/O(*Input Output*) yang berorientasi objek dan implementasi *library* bagi *programmer* java untuk dapat mengakses kemampuan penuh dari *platform* I/O Raspberry Pi [6]. Proyek ini mengabstraksikan integrasi asing pada low-level dan menginterupsi pengawasan untuk memungkinkan programmer java fokus ke dalam masalah implementasi dari logika bisnis pada aplikasi.

Proyek ini masih dikembangkan lebih lanjut namun penggunaannya masih stabil dalam beberapa tahun terakhir. Fungsionalitas sangat stabil dan perbaikan bug terus menerus dilakukan dengan mengembangkan cabang pada *repository* GitHub Pi4J. Beberapa versi Raspberry Pi yang mendukung dalan penggunaan library ini adalah

- Raspberry Pi - Model A
- Raspberry Pi - Model B (Revision 1)
- Raspberry Pi - Model B (Revision 2)
- Raspberry Pi - Model A+
- Raspberry Pi - Model B+
- Raspberry Pi - Compute Module
- Raspberry Pi 2 - Model B

2.5 Food Security *Kit*

Food Security Kit Adalah alat uji cepat atau *test kit* untuk mengidentifikasi zat kimia yang terkandung pada makanan. Terdapat berbagai macam pengujian untuk mendeteksi jenis zat kimia yang berbeda-beda. Beberapa parameter yang diuji menggunakan Food Security Kit adalah sebagai berikut

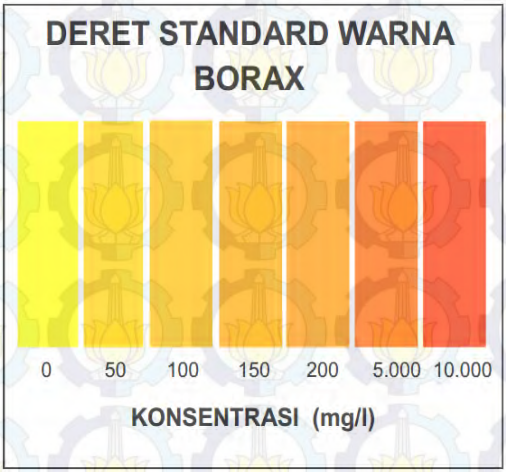
- *Arsenic*

  Arsenic Adalah senyawa kimia yang biasanya terdapat pada sulfur dan berbagai jenis metal. *Arsenic* biasa digunakan untuk memperkuat campuran tembaga dan pelaspis timbal pada baterai. Selain itu *arsenic* juga digunakan pada produksi pestisida, insektisida dan berbagai perabotan kayu. Tanaman lebih mudah menyerap zat *arsenic*, sehingga memungkinkan arsen berada dalam pangan pada konsentrasi tinggi dalam bentuk organik dan anorganik.

- Boraks

  Merupakan garam natrium yang banyak digunakan di berbagai industri non pangan seperti industri kertas, gelas, pengawet kayu dan keramik [7]. Namun boraks banyak disalahgunakan pada makanan seperti bakso, mie dan berbagai

jenis makanan lainnyadengan tujuan agar makanan tersebut bersifat lebih kenyal sehingga menambah tekstur ketika disantap.penggunaan boraks pada makanan telah dilarang oleh pemerintah karena konsumsi boraks pada kadar tertentu dapat menyebabkan kematian.



Gambar 2.1. Warna Standar untuk Uji Boraks

Untuk uji boraks *kit* dilengkapi dengan deret warna standar untuk boraks sesuai pada Gambar 2.1. uji boraks dari kit hanya mampu mendeteksi boraks dari konsentrasi 0-10.000 mg/l. Menggunakana acuan AOAC 959.09 (*Association of Official Analytical Chemists*) [8]. Prosedur uji boraks yang tertera pada Food Security Kit dijelaskan sebagai berikut :

1. Siapkan *beaker glass* dan masukkan sampel makanan 25 gr dalam volume 50 ml *aquadest* dan hancurkan dengan pengaduk sampai larut seluruhnya, untuk sampel minuman yang sudah cair tidak perlu dilakukan perlakuan awal, karena boraks dalamkeadaan dingin membentuk garam sebaiknya sampel sebelum diperiksa dipanaskan terlebih dahulu dengan suhu 80 oC selama 3 - 5 menit setelah tercapai didinginkan

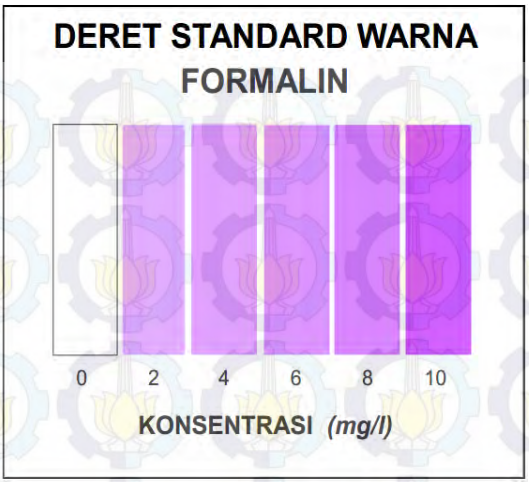
2. Siapkan tabung reaksi masukkan 5 ml sampel serta tambahkan reagen “ Borax - 1 “ sebanyak 3 tetes di aduk hingga merata
3. Siapkan “ Kertas Borax ” teteskan sampel pada perlakuan “ 2 “ pada permukaannya 3 tetes dan diamkan beberapa saat jika ada borax (B4O72-) akan terbentuk perubahan warna dari kuning menjadi merah untuk lebih meyakinkan bandingkan dengan “Standard Borax ” yang di perlakukan sebagai sampel.

- Sianida

  Sianida adalah racun alami yang terdapat pada tanaman singkong yang berbahaya jika dikonsumsi melebihi batas. Terutama pada singkong pahit kandungan sianida didalamnya sangat tinggi dan berbahaya jika dikonsumsi tanpa melalui proses yang semestinya.

- Formalin

  Larutan yang tidak berwarna, berbau tajam yang mengandung 37% *formaldehyde* dalam air dan biasanya ditambahkan methanol 10%-15% sebagai pengawet [7]. Formalin tidak diizinkan untuk ditambahkan ke dalam bahan makanan sebagai pengawet. Pada umumnya formalin digunakan sebagai pembunuh kuman pada pembersih lantai, pembasmi serangga, bahan peledak, pencegah korosi , bahan perekat kayu lapis dan sebagainya.



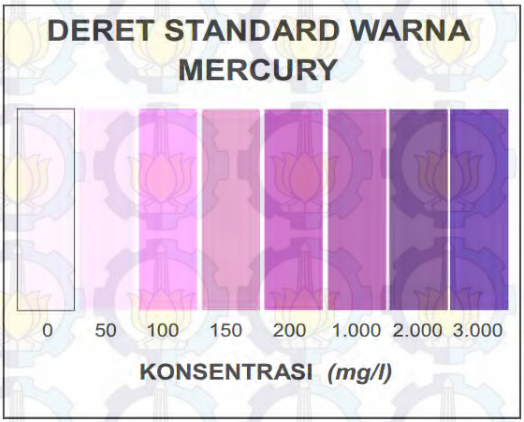
Gambar 2.2. Warna Standar untuk Uji Formalin

Untuk uji formalin *kit* dilengkapi dengan deret warna standar untuk formalin sesuai pada Gambar 2.2. Uji formalin dari kit hanya mampu mendeteksi formalin dari konsentrasi 0-10 mg/l [8]. Prosedur uji formalin yang tertera pada Food Security Kit dijelaskan sebagai berikut :

1. Siapkan *beaker glass* dan masukkan sampel makanan 25 gr dalam volume 50 ml *aquadest* dan hancurkan dengan pengaduk sampai larut seluruhnya, untuk sampel minuman yang sudah cair tidak perlu dilakukan perlakuan awal
2. Siapkan tabung reaksi dan masukkan 2 ml sampel serta tambahkan reagen "Formalin - 1" sebanyak 3 tetes
3. Tambahkan reagen "Formalin - 2" sebanyak 5 tetes
4. Tambahkan reagen "Formalin - 3" sebanyak 3 tetes dan diamkan selama 15 menit, akan terbentuk warna ungu muda seulas sampai ungu tua menunjukkan formalin positif, bandingkan dengan deret standard warna formalin
5. Untuk lebih meyakinkan bandingkan dengan standard formalin yang diperlakukan sebagai sampel.

- Merkuri

Merkuri adalah logam berat berbahaya dan terdapat secara alami di lingkungan sebagai hasil dari perombakan minreal di alam melalui proses cuaca/iklim, dari angin dan air. Senyawa merkuri anorganik berupa serbuk atau larutan putih kecuali merkuri sulfida berwarna merah dan berwarna hitam bila terkena cahaya. Merkuri digunakan untuk termometer, barometer, amalgam gigi, pgimen cat dan pigmen untuk membuat tato.



Gambar 2.3. Warna Standar untuk Uji Merkuri

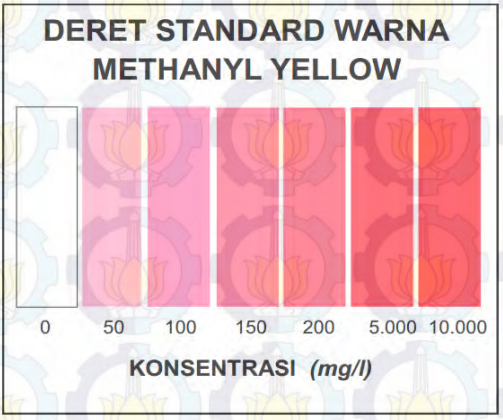
Untuk uji Merkuri *kit* dilengkapi dengan deret warna standar untuk Merkuri sesuai pada Gambar 2.3. Uji Merkuri dari kit hanya mampu mendeteksi Merkuri dari konsentrasi 0-3.000 mg/l [8]. Prosedur uji Merkuri yang tertera pada Food Security Kit dijelaskan sebagai berikut :

1. Siapkan beaker glass dan masukkan sampel makanan 25 gr dalam volume 50 ml *aquadest* dan hancurkan dengan pengaduk sampai larut seluruhnya, untuk sampel minuman yang sudah cair tidak perlu di lakukan perlakuan awal

2. Siapkan tabung reaksi masukkan 5 ml sampel serta tambahkan reagen “Mercury - 1“ sebanyak 3 tetes di aduk hingga merata
3. Siapkan “Kertas Mercury” teteskan sampel pada perlakuan “2“ pada permukaannya 3 tetes dan diamkan beberapa saat akan terjadi perubahan warna putih merah seulas menjadi merah kebiruan hingga menjadi biru semakin tinggi konsentrasi Merkuri semakin pekat warna birunya
4. Untuk lebih meyakinkan bandingkan dengan “Standard Mercury” yang diperlakukan sebagai sampel.

- *Methanyl Yellow*

Methanyl Yellow merupakan bahan pewarna sintetik berbentuk serbuk berwarna kuning kecoklatan , mudah larut dalam air dan alkohol. Pewarna ini umumnya digunakan sebagai pewarna pada tekstil, kertas, tinta, plastik, kulit dan cat serta sebagai indikator asam basa di laboratorium. *Methyl Yellow* bersifat iritan sehingga jika dikonsumsi akan menyebabkan iritasi saluran pencernaan [9].



Gambar 2.4. Warna Standar untuk Uji *Methanyl Yellow*

Untuk uji *Methanyl Yellow kit* dilengkapi dengan deret warna standar untuk *Methanyl Yellow* sesuai pada Gambar 2.4. Uji *Methanyl Yellow* dari kit hanya mampu mendeteksi *Methanyl Yellow* dari konsentrasi 0-10.000 mg/l [8]. Prosedur uji *Methanyl Yellow* yang tertera pada Food Security Kit dijelaskan sebagai berikut :

1. Siapkan beaker glass dan masukkan sampel makanan 25 gr dalam volume 50 ml *aquadest* dan hancurkan dengan pengaduk sampai larut seluruhnya, untuk sampel minuman yang sudah cair tidak perlu dilakukan perlakuan awal
2. Siapkan tabung reaksi dan masukkan 0,5 - 1 ml sampel serta tambahkan reagen "Methanyl Yellow - 1" sebanyak 2 tetes diamkan beberapa saat akan terbentuk warna merah muda seulas sampai pekat (merah tajam) menunjukkan *Methanyl Yellow* positif, bandingkan dengan deret standard warna *Methanyl Yellow*
3. Untuk lebih meyakinkan bandingkan dengan standard *Methanyl Yellow* yang di perlakukan sebagai sampel.

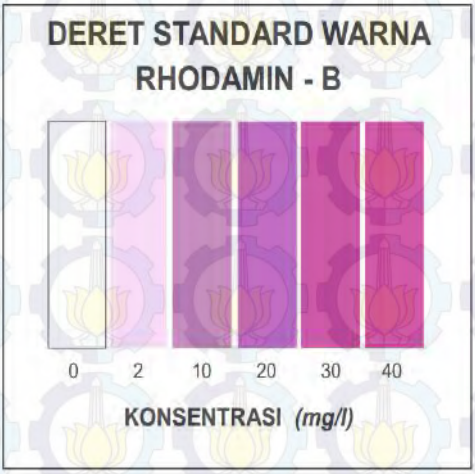
- Timbal

Timbal merupakan logam beracun yang secara alami dapat ditemukan dalam tanah. Timbal tidak memiliki bau dan rasa. Timbal juga dapat ditemukan pada cat usang, debu, udara, air, makanan , tanah yang terkontaminasi dan bahan bakar. Timbal digunakan pada pembuatan gelas, penstabil senyawa PVC, cat minyak dan bahan bakar.

- *Rhodamin B*

Merupakan pewarna sintesis berbentuk kristal berwarna hijau atau ungu kemerahan, tidak berbau dan dalam larutan akan berwarna merah terang. *Rhodamin B* sering digunakan pada industri tekstil dan kertas , pewarna kain , kosmetik, pembersih mulut dan sabun. Konsumsi pewarna ini berbahaya bagi kesehatan karena sifat kimia dan kandungan logam

beratnya yang terakumulasi akan menyebabkan gangguan fungsi hati dan ginjal.



Gambar 2.5. Warna Standar untuk Uji *Rhodamin-B*

Untuk uji *Rhodamin B kit* dilengkapi dengan deret warna standar untuk *Rhodamin B* sesuai pada Gambar 2.5. Uji *Rhodamin B* dari kit hanya mampu *Rhodamin B* dari konsentrasi 0-40 mg/l [8]. Prosedur uji *Rhodamin B* yang tertera pada Food Security Kit dijelaskan sebagai berikut :

1. Siapkan beaker glass dan masukkan sampel makanan 25 gr dalam volume 50 ml *aquadest* dan hancurkan dengan pengaduk sampai larut seluruhnya, untuk sampel minuman yang sudah cair tidak perlu di lakukan perlakuan awal
2. Siapkan tabung reaksi dan masukkan 3 tetes reagen "Rhodamin B - 1" serta tambahkan sampel sebanyak 5 ml secara perlahan dan diamkan beberapa saat akan terjadi perubahan warna merah menjadi merah kebiruan apabila warna *Rhodamin B* tidak pekat warna menjadi putih kebiruan, apabila pekat warna menjadi merah kebiruan bandingkan dengan deret standard warna *Rhodamin B*

3. Untuk lebih meyakinkan bandingkan dengan standard *Rhodamin B* yang di perlakukan sebagai sampel.

- Cyclamate

  Cyclamate merupakan zat berbentuk kristal ,tidak berbau, tidak berwarna dan mudah larut dalam air dan etanol. Intensitas kemanisannya 30 kali lebih manis dibanding sukrosa. Metabolisme *Cyclamate* dalam tubuh bersifat karsinogenik sehingga ekskresi *Cyclamate* dalam urin dapat merangsang tumor dan mempu menyebabkan kerusakan kromosom [8].

Secara umum untuk Food Security Kit yang digunakan, pengguna menentukan kadar dari zat yang diuji dengan membandingkan perubahan warna sampel dengan warna standar. Lalu kadar ditentukan dengan menentukan salah satu warna standar yang paling mirip dengan warna sampel. Pada warna standar tersebut terdapat indikasi berupa konsentrasi dari zat yang terdapat dalam sampel.

2.6 Klasifikasi KNN(*K-Nearest Neighbor)*

KNN merupakan salah satu metode klasifikasi yang menjadi pilihan pertama ketika hanya sedikit atau tidak terdapat pengetahuan sebelumnya tentang distribusi dari data. KNN memiliki algoritma yang simpel dengan menyimpan seluruh data yang tersedia dan mengklasifikasikannya berdasarkan pengukuran kemiripan. Pengukuran kemiripan didapat berdasarkan dengan hasil perhitungan jarak antar data. Jarak dihitung menggunakan fungsi Euclidian *distance*. Namun hasil dari fungsi Euclidian *distance* hanya akan valid ketika nilai dari data bersifat kontinu. Jika terdapat fitur data yang bersifat kategorikal atau numerik maka sebaiknya dilakukan normalisasi terlebih dahulu pada dataset. Fungsi Euclidian *distance* didefinisikan sebagai pada Persamaan 2.1.

$$d(x_i, y_i) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (2.1)$$

Keterangan :

$d(x_i, y_i)$ = jarak euclidian
x_i = data x ke-i
y_i = data y ke-i
i = 0,1,2,3…n

Algoritma KNN memprediksi kelas dari data uji dengan menentukan sejumlah K data latih yang terdekat dengan salah satu dari setiap data uji. Dari sejumlah K data latih tersebut prediksi kelas data uji ditentukan dengan modus kelas pada data latih yang terpilih. Algoritma KNN didefinisikan sebagai berikut :

1. Tentukan nilai K, data latih dan data uji
2. Tentukian sejumlah K data dari data latih yang terdekat dari data uji menggunakan Euclidian *distance*
3. Tentukan modus kelas dari sejumlah data latih yang terpilih
4. Kelas yang merupakan modus dari kelas data latih terpilih menjadi kelas prediksi.

2.7 Socket

Socket adalah titik akhir dari komunikasi 2 arah antara 2 program yang berjalam dalam sebuah jaringan [10]. *Socket* terikat dengan nomor *port* sehingga lapisan TCP(*Tranmission Control Protocol*) dapat mengidentifikasikan aplikasi yang dituju untuk mengirimkan data. Biasanya dalam sebuah jaringan *Socket* diimplementasikan pada 2 program berbeda yaitu *Client* dan *Server*. Pada umumnya *Server* berjalan pada komputer spesifik dan memiliki *Socket* dengan nomor prot yang spesifik.

Tugas *Server* hanya menunggu dan terus mendengarkan *Socket* hingga terdapat permintaan dari *Client* yang terhubung

untuk memberikan respon. Pada sisi *Client* diperlukan alamat dari host yang berperan sebagai *Server* yang dituju dan nomor *port* dari aplikasi *Server*. Ketika *Server* menerima hubungan dari *Client Server* mendapatkan *Socket* baru yang terikat dengan alamat dari *Client* dan nomor *port Client*. Pada sisi *Client* ketika koneksi diterima, sebuah *Socket* berhasil dibuat dan dapat digunakan untuk saling berhubungan dengan *Server*. *Client* dan *Server* saling berhubungan dengan menulis dan membaca pesan yang dikirimkan melalui *Socket* yang dibuat.

Sebuah titik akhir dari kedua sisi merupakan kombinasi antara alamat IP dengan nomor *port*. Setiap koneksi TCP dapat diidentifikasi secara unik berdasarkan kedua titik akhir dari *Socket*.

2.8 JDK(*Java Developer Kit*) 8

JDK adalah lingkungan pengembangan untuk membangun aplikasi dan komponennya menggunakan bahasa pemrograman Java. JDK memiliki kakas yang berguna untuk pengembangan dan pengujian program yang dibuat dengan bahasa pemrograman Java dan berjalan dengan platform Java. JDK memiliki JRE(*Java Runtime Environtment*) berupa kakas bantu seperti *compilers* dan *debuggers* yang dibutuhkan untuk mengembangkan aplikasi perangkat lunak. JDK dapat diunduh dibawah persetujuan Oracle BCL(*Binary Code License*).

2.9 RVLM (*Royal Vial Lab Multireader*)

RVLM adalah alat untuk menganalisa mikroorganisme secara kuantutatif. Analisa dilakukan dengan memperhatikan perubahan warna pada botol kecil yang menjadi wadah untuk sampel. Warna dari sampel pada botol akan berubah ketika terdapat mikroorganisme tertentu, semakin banyak mikroorganisme maka semakin cepat reaksi perubahan warna yang terjadi. Beberapa fitur yang terdapat pada RVLM adalah :

- Kecepatan: waktu analisa dari persiapan hingga mendapatkan hasil analisis membutuhkan waktu 2 sampai 5 kali kurang dari yang digunakan dengan metode tradisional
- Kemudahan dalam penggunaan : siapapun, dimanapun dapat melakukan analisa tanpa reagent atau peralatan khusus
- Sensitifitas : dapat mendeteksi bahkan satu mikroorganisme yang ada di sampel
- Selektifitas : dapat mendeteksi spesies berbeda dari organisme mikroba ke eksperimen dengan batas 99.999%
- Biaya: biaya yang dibutuhkan untuk analisa 2 hingga 4 kali lebih murah dibanding dengan metode tradisional [2]

Gambar 2.6. RVLM(*Royal Vial Lab MultiReader*)

Gambar 2.6 adalah bentuk fisik dari RVLM. Terdiri dari 4 katup untuk tempat sampel. Metode yang digunakan RVLM telah divalidasi sesuai dengan ISO 1614:2003 "*Microbiology of food and animal feeding stuffs- Protocol for the validation of alternative methods*".

2.10 PrimeLab Photometer

PrimeLab Photometer merupakan alat bantu untuk menentukan nilai dari parameter air yang merupakan peralatan standar di setiap laboratorium [10]. Terdiri dari seperangkat alat

yang terdapat sensor didalamnya dan aplikasi desktop yang dapat terhubung melalui koneksi *bluetooth*. Aplikasi perangkat lunak bawaan dari alat bantu ini dapat digunakan untuk mengatur dan mengevaluasi hasil pengukuran kandungan dalam sampel dengan mudah. Aplikasi dapat terhubung secara *remote* memudahkan penguna untuk ,emgaktifkan parameter uji tambahan yang dapat diunduh diluar parameter uji yang terdapat dalam sistem sebelumnya. Alat terdiri dari lampu dan sensor JenColor yang mampu menangkap warna dengan variabel cahaya.

Gambar 2.7. PrimeLab Photometer

Gambar 2.7 merupakan bentuk fisik dari PrimeLab Photometer. Terdiri dari satu lubang katup untuk kontainer *vial* sampel. Pada kontainer sampel terdapat satu sensor JenColor dan satu lampu untuk pencahayaan. Empat tombol dan layar *analog* untuk tampilan dan interaksi pengguna.

BAB III
ANALISIS DAN PERANCANGAN

Pada bab ini dibahas analisa kebutuhan, proses bisnis dan desain dari sistem yang dibangun dalam Tugas Akhir ini. Bagian awal bab akan dibahas tentang analisa permasalahan, deskripsi umum sistem dan kebutuhan dari sistem yang akan dibangun. Bagian berikutnya akan dibahas rancangan sistem yang ditujukan untuk memberikan gambaran tentang perangkat lunak yang dibuat.

3.1 Analisis Perancangan Sistem

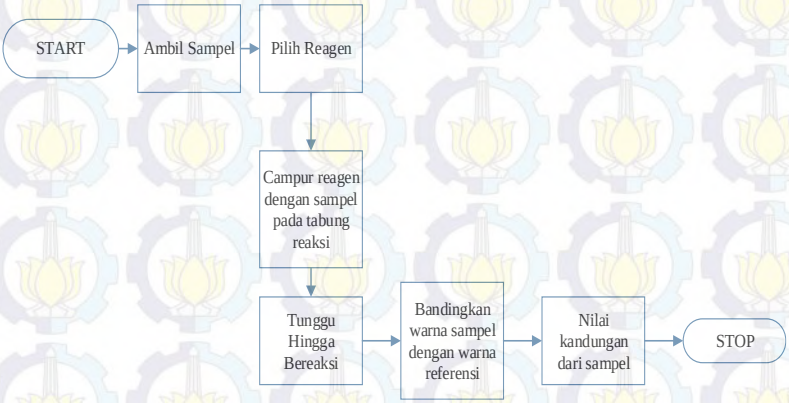
Pada subbab ini akan dijelaskan analisa dalam pembuatan alat bantu uji keamanan makanan. Analisis meliputi permasalahan yang diangkat, deskripsi secara umum dari sistem dan penjabaran dari kebutuhan sistem yang akan dibangun.

3.1.1 Analisis Permasalahan

Pangan merupakan salah satu kebutuhan pokok manusia yang harus dipenuhi secara kuantitatif ataupun kualitatif. Secara kualitatif setiap makanan haruslah memiliki tingkat keamanan dalam mengkonsumsi makanan tersebut. Oleh karenanya keamanan pangan telah menjadi salah satu perhatian pemerintah dalam memastikan kebutuhan pokok masyarakat terpenuhi.

BPOM(Badan Pengawasan Obat dan Makanan) melakukan pengawasan pada setiap produk makanan yang beredar di masyarakat. Pengawasan ini dilakukan dengan pengambilan sampel pada setiap makanan yang akan diuji lalu pengujian akan dilakukan sesuai dengan jenis pengujian yang dipilih. Pengujian dilakukan untuk menentukan jumlah kandungan zat kimia radikal yang mungkin terdapat pada sampel. Terdapat banyak jenis zat kimia yang menjadi parameter seperti *Methanyl Yellow* yang biasanya digunakan untuk pewarna makanan ataupun seperti formalin yang digunakan untuk merubah tekstur dari makanan.

Biasanya pengujian ini digunakan menggunakan peralatan kimia yang terdiri dari berbagai reagen tertentu sesuai dengan jenis uji zat kimia yang diinginkan.



Gambar 3.1. Proses Uji Makanan Secara Umum

Food Security Kit merupakan seperangkat alat yang digunakan untuk melakukan uji zat kimia pada makanan. Terdiri dari berbagai macam reagen dan peralatan seperti vial atau tabung reaksi. Seperangkat alat ini memiliki panduan tersendiri untuk setiap jenis uji yang dilakukan. Pada umumnya sampel dimasukan dengan reagen ke dalam tabung reaksi. Setelah ditunggu untuk beberapa saat reagen akan bekerja dengan sampel dan merubah warna dari cairan reagen. Setiap warna yang ditunjukkan dapat mengindikasikan jumlah kandungan zat kimia dari sampel yang diuji. Gambar 3.1 menunjukan proses bisnis yang terjadi dalam proses uji makanan. Proses ini dilakukan untuk semua sampel yang ingin diuji. Terdapat permasalahan ketika membandingkan warna hasil uji sampel dengan daftar warna referensi dari jenis uji yang dipilih. Kemungkinan penguji dapat melakukan kesalahan dalam membandingkan warna dikarenakan berbagai macam faktor, seperti misalnya kelelahan ataupun tidak fokus. Hal ini dapat terjadi karena jumlah sampel yang terlalu banyak ataupun jam

kerja yang panjang. Ketidakakuratan yang signifikan dapat berakibat fatal dalam pengujian makanan. Oleh karenanya dibutuhkan sistem yang dapat menangkap warna dan membandingkannya dengan daftar warna referensi.

3.1.2 Deskripsi Umum Sistem

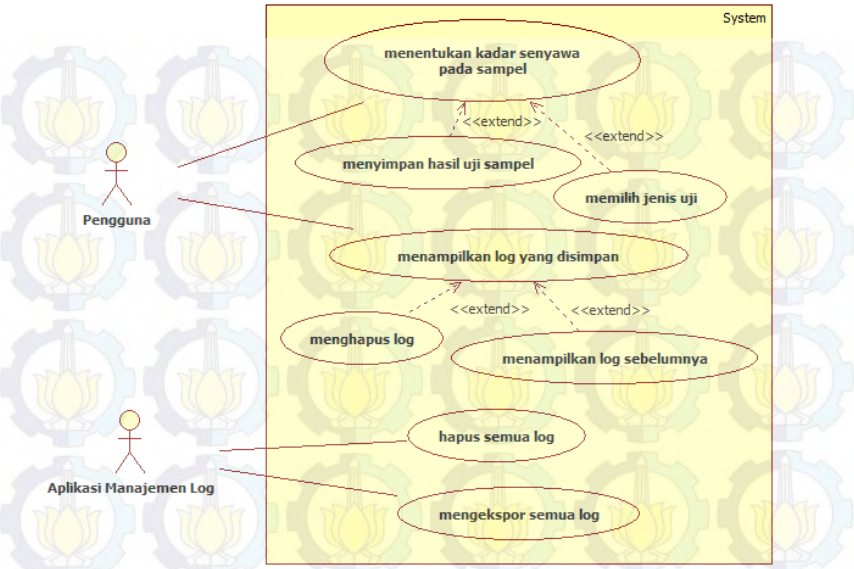
Sistem dibuat untuk menurunkan tingkat kesalahan dalam pencocokan warna hasil sampel uji makanan yang diakibatkan oleh *human error*. Pengguna dapat memilih jenis uji makanan yang akan dilakukan sesuai reagen yang sampel gunakan. Sistem akan mengekstrak warna dari gambar sampel yang diambil. Warna yang diambil akan dibandingkan dengan daftar warna referensi dari jenis uji yang dilakukan. Hasil perbandingan akan ditampilkan oleh sistem sesuai dengan reagen yang digunakan. Daftar warna referensi telah disimpan sebelumnya oleh sistem sesuai dengan referensi dari jenis Food Security Kit yang digunakan.

Fungsionalitas dari sistem apabila diringkas adalah sebagai berikut.

1. Sistem dapat mengambil gambar dan melakukan perbandingan warna uji sampel.
2. Sistem dapat menyimpan,menampilkan dan menghapus *log* berupa uji sampel yang telah dilakukan.

3.1.3 Kebutuhan Fungsional Sistem

Sistem digunakan untuk mendapatkan hasil dari perbandingan warna pada uji sampel dengan daftar warna referensi. Secara keseluruhan kebutuhan fungsional dari sistem dapat digambarkan dengan diagram kasus penggunaan pada Gambar 3.2. Penjelasan lengkap mengenai kasus penggunaan pada sistem dijelaskan pada Tabel 3.1.



Gambar 3.2. Diagram Kasus Penggunaan Sistem

Tabel 3.1. Daftar Kasus Penggunaan

No	Kode	Nama	Keterangan
1	UC-01	Menentukan kadar senyawa dalam sampel	Pengguna dapat menentukan kadar senyawa zat kimia dari sebuah sampel sesuai dengan uji zat kimia yang dipilih
2	UC-02	Menampilkan *log*	Pengguna dapat melihat *log* dari uji sampel yang disimpan
3	UC-03	Hapus Semua Log	Aplikasi Manajemen *log* dapat mengirimkan permintaan ke sistem

			untuk menghapus semua *log*
4	UC-04	Mengekspor semua *log*	Aplikasi manajemen *log* dapat mengirimkan permintaan ke sistem untuk mengekspor log

3.1.3.1 Deskripsi Kasus Penggunaan UC-01

Kasus penggunaan kode UC-01 merupakan kasus penggunaan menentukan kadar senyawa dalam sampel. Rincian alur kasus dijelaskan pada Tabel 3.2.

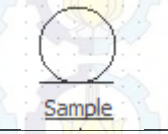
Tabel 3.2. Spesifikasi Kasus Penggunaan Menentukan Kadar Senyawa dalam Sampel

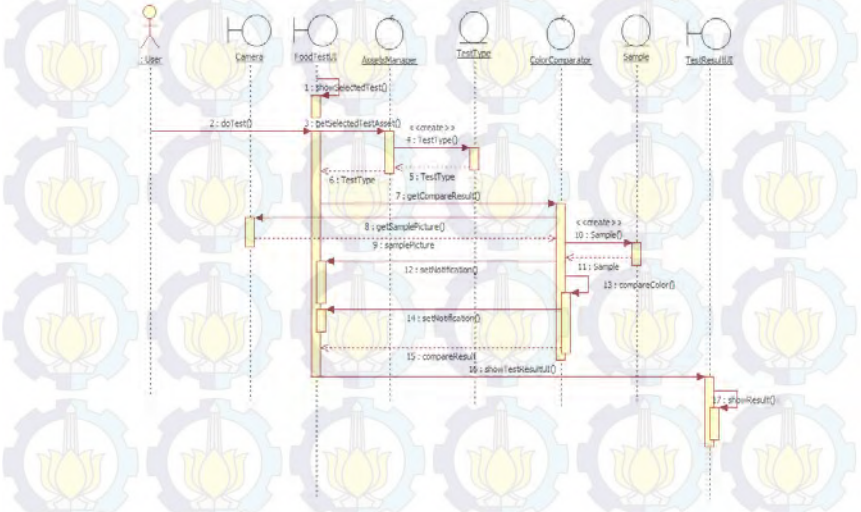
Nama Use Case	Menentukan kadar senyawa dalam sampel
Nomor	UC-01
Aktor	Pengguna
Kondisi Awal	Uji sampel yang belum dibandingkan dengan warna referensi
Kondisi Akhir	Perkiraan kandungan zat kimia pada uji sampel sesuai dengan jenis uji yang dipilih
Alur Normal	1. Sistem menampilkan antarmuka pengujian dengan jenis uji pertama 2. Pengguna menekan tombol mulai untuk melakukan pengujian A1. Pengguna memilih jenis uji lain 3. Sistem menampilkan notifikasi proses pengambilan gambar 4. Sistem menampilkan notifikasi proses penghitungan nilai senyawa 5. Sistem menampilkan hasil dari jenis uji yang dilakukan

	6. Pengguna memilih kembali ke antarmuka pengujian, *log* tidak disimpan. A2. Pengguna memilih untuk menyimpan hasil uji ke*log* 7. Sistem menampilkan antarmuka pengujian dengan jenis uji pertama 8. Sistem menampilkan notifikasi *log* tidak tersimpan
Alur Alternatif	A1. Pengguna memilih jenis uji lain A1.1 Sistem menampilkan jenis uji berikutnya A1.2 Kembali ke alur nomor 2 A2. Pengguna memilih untuk menyimpan hasil uji ke *log* A2.1 Sistem menampilkan antarmuka pengujian dengan jenis uji pertama A2.2.1 Sistem menampilkan notifikasi

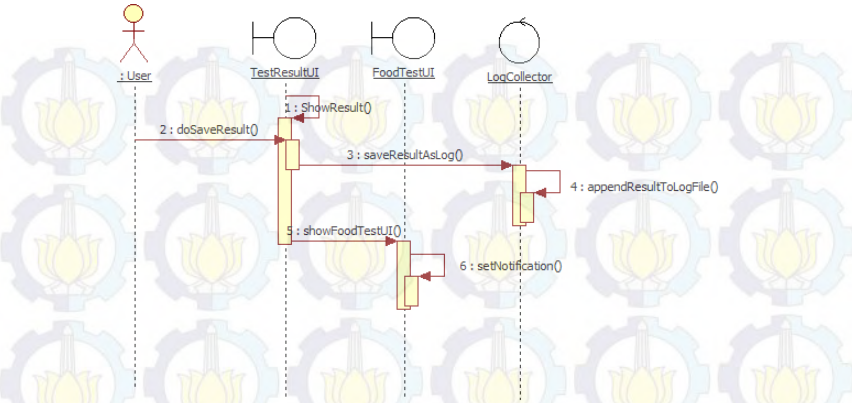
Tabel 3.3. Analisa Kelas untuk Kasus Penggunaan Menentukan Kadar Senyawa dalam Sampel

Kelas	Keterangan
	Kelas *boundary* Camera digunakan untuk menghubungkan dengan antarmuka dari kamera dengan sistem
	Kelas *boundary* FoodTestUI untuk mengatur tampilan dari antarmuka pengujian

	Kelas *control* AssetManager untuk mengatur aset warna referensi yang akan digunakan sesuai pengujian yang dilakukan
	Kelas *entity* TestType untuk representasi kelas dari jenis uji yang digunakan dalam pengujian
	Kelas *control* ColorComparator untuk mengatur proses perbandingan sampel yang diambil kamera dengan warna referensi jenis uji yang dipilih
	Kelas *entity* Sample untuk merepresentasikan sampel yang diambil dalam pengujian



Gambar 3.3. Diagram Alir Kasus Penggunaan Menentukan Kadar Senyawa dalam Sampel

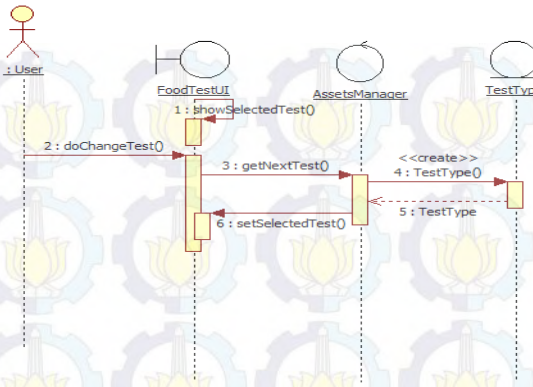


Gambar 3.4. Diagram Alir Kasus Penggunaan UC-01 Aliran Alternatif A1

Analisis kelas untuk kasus penggunaan UC-01 ditunjukkan pada Tabel 3.3. Tabel 3.3 menunjukkan analisa representasi kelas yang digunakan pada kasus penggunaan UC-01. Diagram alir untuk kasus penggunaan UC-01 aliran normal ditunjukkan Gambar 3.3.

Kasus penggunaan diawali dengan menampilkan jenis uji pertama. Lalu diinisiasi oleh pengguna yang menekan tombol mulai pengujian. Hingga hasil pengujian didapatkan dan antarmuka hasil uji ditampilkan. Diagram alir untuk kasus penggunaan UC-01 aliran alternatif A1 ditunjukkan pada Gambar 3.4.

Kasus kebutuhan diawali dengan menampilkan jenis uji pertama. Lalu diinisiasi oleh pengguna yang menekan tombil ganti jenis uji. Hingga jenis uji yang berikutnya terpilih ditampilkan.



Gambar 3.5. Diagram Alir Kasus Penggunaan UC-01 Aliran Alternatif A2

Diagram alir untuk kasus penggunaan UC-01 aliran alternatif A2 ditunjukkan Gambar 3.5. Kasus kebutuhan diawali ketika hasil uji ditampilkan. Lalu diinisiasi oleh pengguna yang menekan tombil simpan hasil uji. Hingga antarmuka pengujian kembali ditampilkan dan menunjukkan notifikasi.

3.1.3.2 Deskripsi Kasus Penggunaan UC-02

Kasus penggunaan kode UC-02 merupakan kasus penggunaan menampikan *log*. Rincian alur kasus dijelaskan pada Tabel 3.4.

Tabel 3.4. Spesifikasi Kasus Penggunaan Menampilkan *Log*

Nama Use Case	Menampilkan *log*
Nomor	UC-02
Aktor	Pengguna
Kondisi Awal	Data dari *log* yang tersimpan
Kondisi Akhir	Data dari *log* yang terakhir disimpan ditampilkan
Alur Normal	1. Pengguna menekan tombol Log

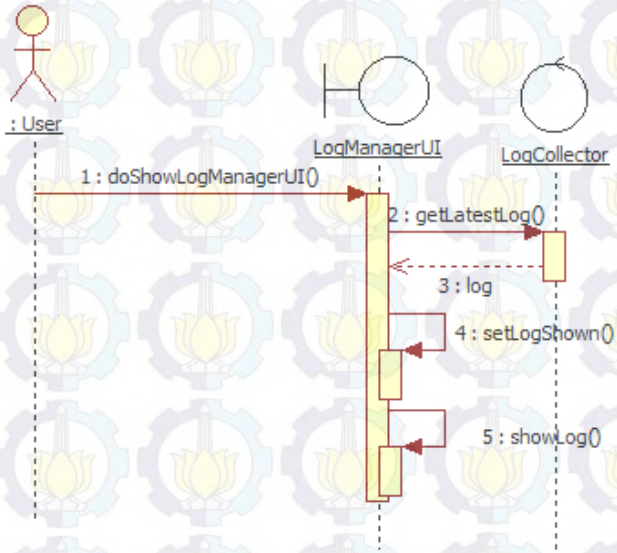
	2. Sistem menampilkan *log* terakhir yang disimpan A1. Tidak ada *log* A2. Pengguna memilih untuk menghapus *log* yang ditampilkan A3. Pengguna memilih untuk menampilkan *log* sebelumnya
Alur Alternatif	A1. Tidak ada *log* A1.1 Sistem menampilkan notifikasi tidak ada *log* A2. Pengguna memilih untuk menghapus *log* yang ditampilkan A2.1 Pengguna menekan tombol hapus untuk menghapus *log* yang ditampilkan A2.2 Sistem menampilkan *log* yang disimpan sebelumnya A2.3 Sistem menampilkan notifikasi *log* telah terhapus A3. Pengguna memilih untuk menampilkan *log* sebelumnya A3.1 Pengguna menekan tombol *log* A3.2 Sistem menampilkan *log* yang sebelumnya disimpan

Tabel 3.5. Analisa Kelas untuk Kasus Penggunaan Menampilkan *log*

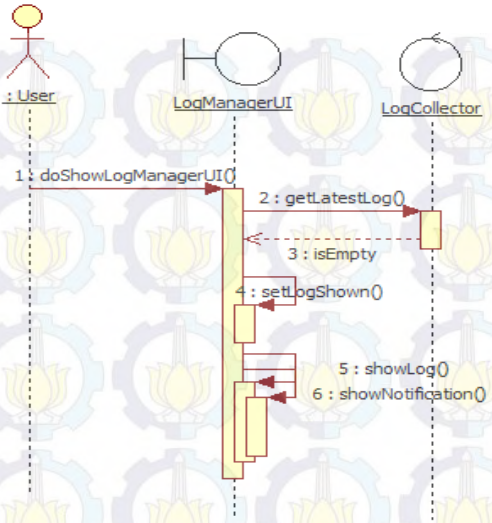
Kelas	Keterangan
	Kelas boundary *Log*ManagerUI yang mengatur tampilan pada antarmuka manajemen *log*

	Kelas control untuk mengatur *log*

Analisa kelas untuk kasus penggunaan UC-02 ditunjukkan padaTabel 3.5. Tabel 3.5 menunjukkan analisa representasi kelas yang digunakan pada kasus penggunaan UC-02. Diagram alir untuk kasus kebutuhan UC-02 aliran normal ditunjukkan Gambar 3.6. Kasus kebutuhan diawali dengan pengguna menekan tombol *Log* untuk menampilkan antarmuka manajemen *log*. Hingga *log* yang paling terakhir disimpan ditampilkan.



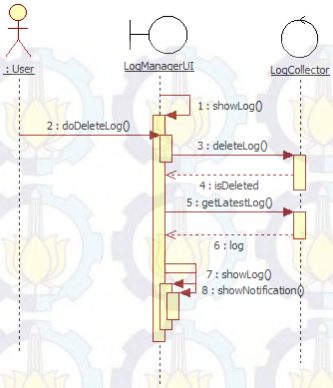
Gambar 3.6. Diagram Alir Kasus Penggunaan Menampilkan *Log*



Gambar 3.7. Diagram Alir Kasus Penggunaan UC-02 Aliran Alternatif A1.

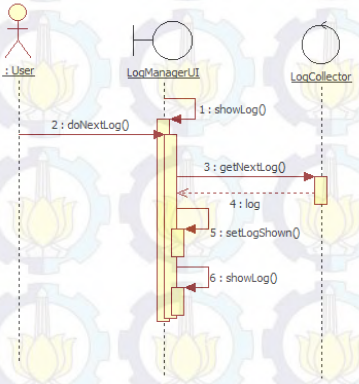
Diagram alir untuk kasus penggunaan UC-02 aliran alternatif A1 ditunjukkan Gambar 3.7. Kasus kebutuhan diawali dengan pengguna menekan tombol Log untuk menampilkan antarmuka manajemen *log*. Lalu sistem menampilkan notifikasi karena tidak ada *log* yang tersedia.

Diagram alir untuk kasus penggunaan UC-02 aliran alternatif A2 ditunjukkan Gambar 3.8. Kasus penggunaan diawali dengan sistem menampilkan *log*. Lalu diinisiasi oleh pengguna yang menekan tombol hapus *log*. Hingga sistem menampilkan *log* yang disimpan paling terakhir dan menunjukan notifikasi *log* terhapus.



Gambar 3.8. Diagram Alir untuk Kasus Penggunaan UC-02 Aliran Alternatif A2.

Diagram alir untuk kasus penggunaan UC-02 aliran alternatif A3 ditunjukkan Gambar 3.9. Kasus penggunaan diawali dengan sistem menampilkan *log*. Lalu diinisiasi oleh pengguna yang menekan tombol *log* berikutnya. Hingga sistem menampilkan *log* berikutnya yang disimpan terakhir.



Gambar 3.9. Diagram Alir Kasus Penggunaan UC-03 Aliran Alternatif A3

3.1.3.3 Deskripsi Kasus Penggunaan UC-03

Kasus penggunaan UC-03 merupakan kasus penggunaan hapus semua *log*. Pada kasus penggunaan ini aktor utama adalah aplikasi manajemen log yang merupakan aplikasi di luar sistem yang memberikan permintaan ke sistem. Dalam kasus penggunaan ini perintah yang dikirimkan berupa perintah untuk menghapus semual *log* di sistem. Tabel 3.6 menunjukkan rincian alur kasus penggunaan.

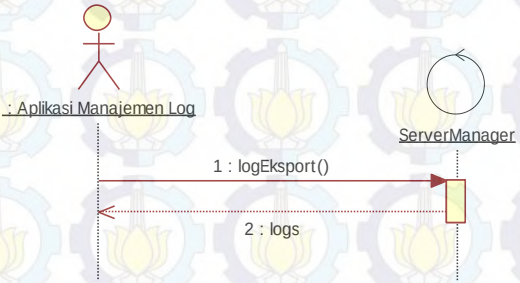
Tabel 3.6. Spesifikasi Kasus Penggunaan Hapus Semua *Log*

Nama Use Case	Hapus Semua *Log*
Nomor	UC-03
Aktor	Aplikasi Manajemen Log
Kondisi Awal	Data dari *log* yang tersimpan
Kondisi Akhir	*File* penyimpanan *log* kososng
Alur Normal	1. Aplikasi Manajemen Log mengirimkan permintaan hapus *log*. 2. Sistem menghapus semua log yang disimpan

Tabel 3.7. Analisis Kelas untuk Kasus Penggunaan Hapus Semua *Log*

Kelas	Keterangan
	Kelas control ServerManager akan berjalan di belakang aplikasi sistem sebagai *thread*. Kelas ini akan mengatur permintaan kelas Aplikasi Manajemen Log dan menjalankan permintaan tersebut.

Analisa kelas untuk kasus penggunaan UC-03 ditunjukkan pada Tabel 3.7. Tabel 3.7 menunjukkan analisa representasi kelas yang digunakan pada kasus penggunaan UC-03. Diagram alir untuk kasus kebutuhan UC-03 aliran normal ditunjukkan Gambar 3.10. Kasus kebutuhan diawali dengan Aplikasi Manajemen Log mengirimkan perintah untuk memanghapus semua *log*. Hingga semua *log* yang disimpan sistem terhapus.



Gambar 3.10. Diagram Alir Kasus Penggunaan UC-03

3.1.3.4 Deskripsi Kasus Penggunaan UC-04

Kasus penggunaan UC-04 merupakan kasus penggunaan mengekspor semua log. Pada kasus penggunaan ini aktor utama adalah aplikasi manajemen log yang merupakan aplikasi di luar sistem yang memberikan permintaan ke sistem. Dalam kasus penggunaan ini perintah yang dikirimkan berupa perintah untuk mengekspor *file log* dari sistem ke Aplikasi Manajemen Log. Tabel 3.8 menunjukkan rincian alur kasus penggunaan.

Tabel 3.8. Spesifikasi Kasus Penggunaan Mengekspor Semua *Log*

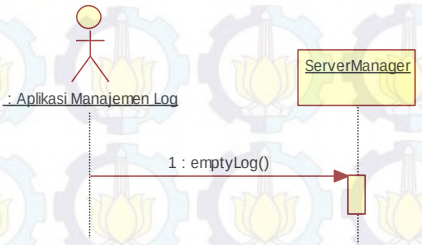
Nama Use Case	Mengekspor Semua *Log*
Nomor	UC-04
Aktor	Aplikasi Manajemen Log

Kondisi Awal	Data dari *log* yang tersimpan
Kondisi Akhir	*File* penyimpanan *log* terkirim ke Aplikasi Manajemen Log
Alur Normal	1. Aplikasi Manajemen Log mengirimkan permintaan ekspor *file log*. 2. Sistem mengirim *file log* yang disimpan

Tabel 3.9. Analisis Kelas untuk Kasus Penggunaan Ekspor *File Log*

Kelas	Keterangan
	Kelas control ServerManager akan berjalan di belakang aplikasi sistem sebagai *thread*. Kelas ini akan mengatur permintaan kelas Aplikasi Manajemen Log dan menjalankan permintaan tersebut.

Analisa kelas untuk kasus penggunaan UC-04 ditunjukkan pada Tabel 3.9. Tabel 3.9 menunjukkan analisa representasi kelas yang digunakan pada kasus penggunaan UC-04. Diagram alir untuk kasus kebutuhan UC-04 aliran normal ditunjukkan Gambar 3.11. Kasus kebutuhan diawali dengan Aplikasi Manajemen Log mengirimkan perintah untuk memanghapus semua *log*. Hingga semua *log* yang disimpan sistem terhapus.



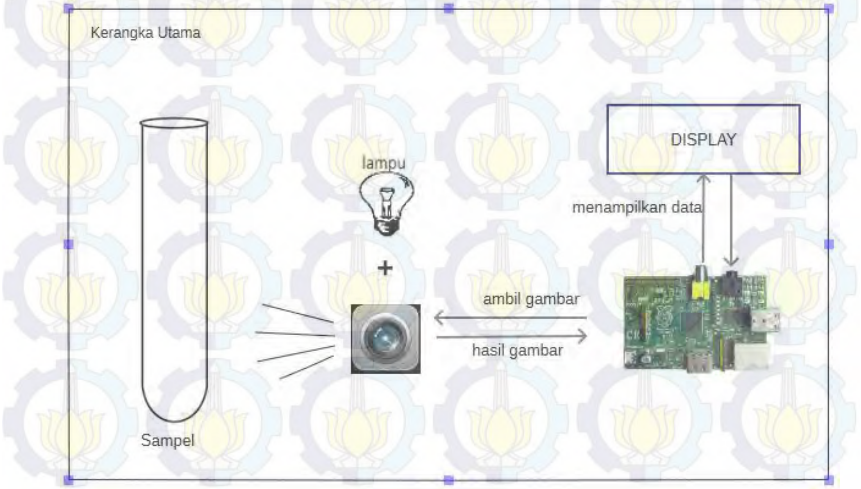
Gambar 3.11. Diagram Alir untuk Kasus Penggunaan UC-04

3.2 Perancangan Sistem

Tahap perancangan dari system dibagi menjadi beberapa bagian yaitu perancangan arsitektur sistem, perancangan diagram kelas, dan perancangan antarmuka aplikasi sistem.

3.2.1 Arsitektur Sistem

Sistem dibuat mengikuti kebuthan pengguna dalam proses uji makanan. Tujuan akhir yang dalam pembuatan sistem adalah menghasilkan sebuah produk berupa alat bantu dalam proses bisnis uji makana dengan Food Security Kit. Oleh karenanya dibutuhkan beberapa jenis perangkat keras yang menjadi pendukung sistem untuk dapat memenuhi proses bisnis yang terjadi. Sebagai *core* atau inti dari sistem adalah computer kecil yang mampu menjalankan komputasi untuk melakukan perbandingan terhadap warna yang diambil. Dibutuhkan sebuah kamera kecil untuk mengekstrak warna dari sampel. Kerangka luar akan dibuat sebagai wadah dari sistem secara keseluruhan.



Gambar 3.12. Rancangan Arsitektur Perangkat Keras Sistem

Kerangka luar ini juga digunakan untuk menyesuaikan kondisi cahaya yang dapat mempengaruhi hasil penangkapan gambar oleh kamera. Kondisi dalam kerangka memiliki cahaya yang sangat minimum, oleh karenanya dibuthukan pencahayaan dari lampu untuk memberikan penglihatan ke kamera. Aplikasi dari sistem akan ditanamkan pada komputer kecil. Penambahan *display* untuk komputer kecil digunakan untuk menampilkan antarmuka dari aplikasi. Secara keseluruhan arsitektur dari sistem yang ingin dibuat sesuai dengan Gambar 3.12.

Sesuai dengan alur dari proses bisnis yang terjadi pada uji makanan. Antarmuka aplikasi pada komputer kecil akan ditampilkan oleh *display*. Pengguna memasukan sampel ke dalam alat lalu memilih untuk memulai uji. Maka tabung berisi sampel yang dimasukan ke sistem akan diambil gambarnya dengan kamera. Lalu dari gambar tersebut akan diambil warna pada sampel yang akan dibandingkan dengan warna standar jenis uji yang dipilih. Hasil uji akan ditampilkan pada *display*.

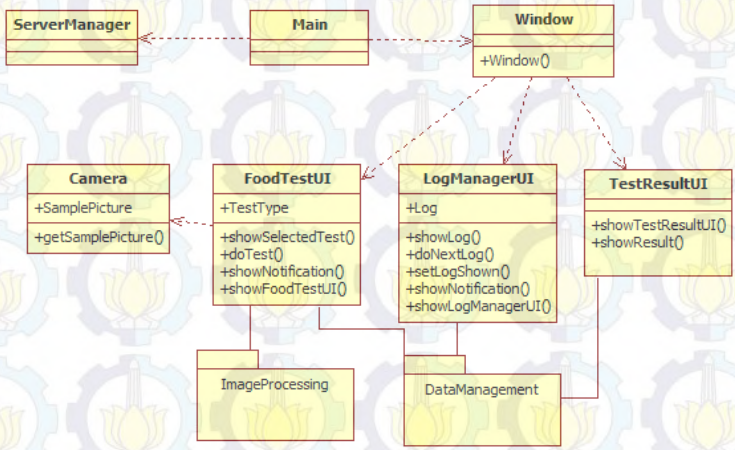
3.2.2 Perancangan Diagram Kelas

Perancangan diagram kelas berisi perancangan dari kelas-kelas yang digunakan untuk membuat aplikasi dari sistem. Hubungan dan perilaku antar kelas akan dijabarkan dengan lebih jelas. Modul yang terdapat pada sistem ini terdiri dari modul antarmuka, modul manajemen data dan modul pemrosesan gambar.

3.2.2.1 Modul Antarmuka

Pada modul ini terdapat kelas-kelas yang mengatur tampilan dari aplikasi sistem. Hubungan antar kelas ditunjukkan dengan diagram kelas sesuai pada Gambar 3.13. Kelas-kelas tersebut diantaranya adalah kelas Main, Window, FoodTestUI, LogManagerUI, TestResultUI dan ShutDownUI. Kelas Main berperan menjadi inisiator untuk jalannya aplikasi. Window merupakan penampung dari kelas tampilan untuk mengatur transisi

perpindahan tampilan aplikasi. FoodTestUI mengatur tampilan untuk uji makanan. LogManagertUI mengatur tampilan pengaturan *log*. TestResultUi mengatur tampilan hasil dari uji makanan. Kelas FoodTestUI akan menggunakan kelas dari modul pemrosesan gambar dan manajemen data untuk menginisiasi proses uji makanan dan pengambilan aset yang dibutuhkan dalam uji makanan. Kelas LogManagerUI akan menggunakan modul dari manajemen data untuk mengambil *log* yang disimpan. Kelas TestResultUI akan mengunakan modul data manajemen untuk menyimpan hasil uji ke *log*.

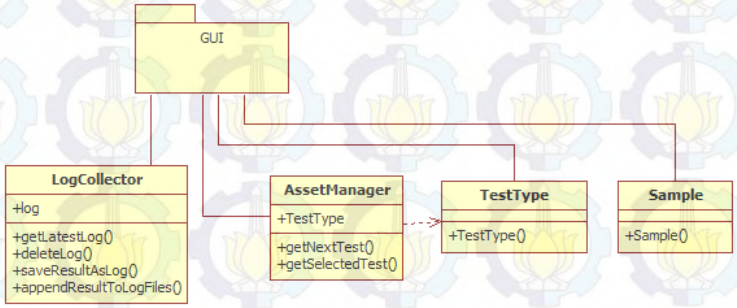


Gambar 3.13. Diagram Kelas Modul Antarmuka

3.2.2.2 Modul Manajemen Data

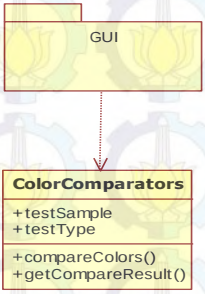
Modul ini akan mengatur penyimpanan dan distribusi dari data yang disimpan. Data dapat berupa *log* dari setiap hasil uji yang disimpan ataupun aset dari sistem. Aset dari sistem dapat berupa daftar warna referensi untuk setiap uji makanan yang disimpan. Hubungan antar kelas digambarkan dengan diagram kelas sesuai Gambar 3.14. Kelas-kelas pada modul ini diantranya adalah

AssetManager, LogCollector, TestType, dan Sample. Kelas AssetManager akan menampung seluruh aset yang digunakan oleh sistem. Kelas LogCollector akan mengatur segala sesuatu yang berhubungan dengan *log*. TestType digunakan untuk mendefinisikan uji makanan yang dilakukan ke dalam sebuah kelas. Kelas Sample akan merepresentasikan sampel yang diuji menjadi sebuah kelas pada aplikasi sistem.



Gambar 3.14. Diagram Kelas Modul Manajemen Data

3.2.2.3 Modul Pemrosesan Gambar

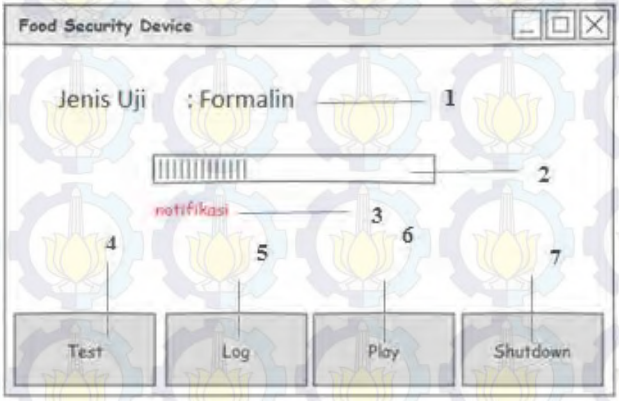


Gambar 3.15. Diagram Kelas Modul Pemrosesan Gambar

Pada modul ini terdapat kelas-kelas yang menunjang pemrosesan gambar dari sampel. Sesuai pada arsitektur pada Gambar 3.12, sampel yang dimasukan akan diambil gambarnya lalu aplikasi akan memproses gambar tersebut. Hubungan antar kelas pada modul ini digambarkan dengan diagram kelas sesuai Gambar 3.15. Kelas pada modul ini adalah ColorComparators. Kelas ini digunakan untuk mengekstrak warna sampel lalu membandingkan warna dari sampel dengan warna referensi.

3.2.3 Perancangan Antarmuka Aplikasi Sistem

Pada sub bab ini akan dibahas rancangan dari antarmuka aplikasi sistem yang akan dibuat. Antarmuka yang dibangun diantaranya adalah antarmuka pengujian makanan, antarmuka manajemen *log*, antarmuka hasil uji makanan, dan antarmuka matikan sistem.



Gambar 3.16. Rancangan Desain Antarmuka Pengujian Makanan

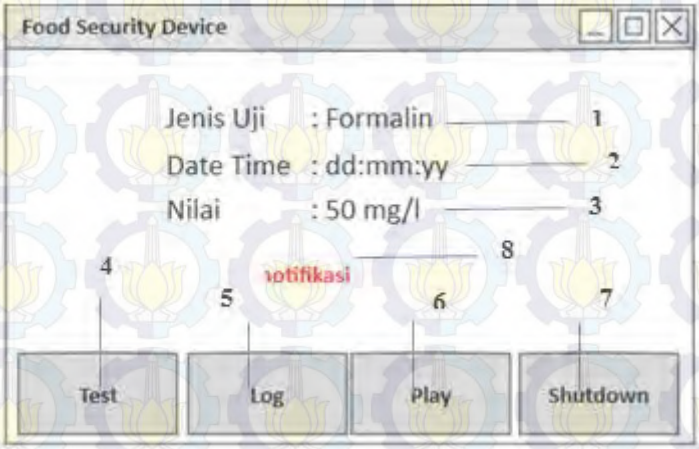
3.2.3.1 Perancangan Antarmuka Pengujian Makanan

Antarmuka ini ditampilkan ketika pengguna pertama kali menggunakan sistem. Gambar 3.16 merupakan rancangan dari

antarmuka yang akan dibangun. Pada bagian ini pengguna dapat memilih jenis uji yang akan dilakukan dan memulai pengujia makanan. Berikut penjelasan masing-masing nomor yang tertera pada Gambar 3.16.

1. Jenis uji yang dipilih oleh pengguna, berupa label yang akan berganti ketika tombl Test ditekan.
2. *Progress bar* mengindikasikan proses perbandingan sampel dengan warna referensi. Hanya akan tampil ketika pengujian telah dilakukan
3. Label notifikasi dari setiap kejadian yang terjadi
4. Tombol Test untuk memilih jenis uji yang ingin dilakukan, akan mengganti label Jenis Uji ketika ditekan.
5. Tombol Log untuk berpindah ke antarmuka Manajemen Log.
6. Tombol Play untuk memulai pengujian sesuai jenis uji yang dipilih.
7. Tombol Shutdown untuk mematikan sistem.

3.2.3.2 Perancangan Antarmuka Manajemen Log

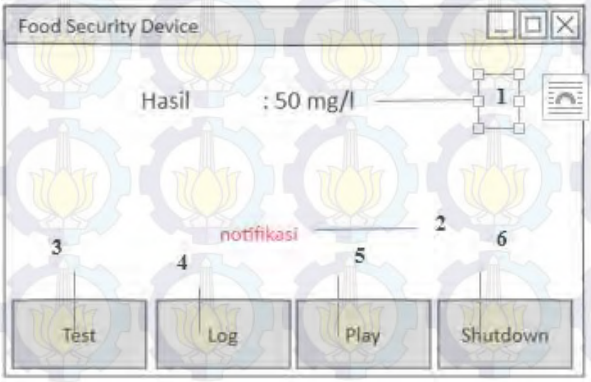


Gambar 3.17 Rancangan Antarmuka Manajemen Log

Antarmuka ini ditampilkan ketika tombol *Log* ditekan. Menampilkan *log* yang terakhir kali dilakukan oleh pengguna. Pengguna juga dapat mencari *log* lain yang telah sebelumnya disimpan serta menghapus *log* yang sedang ditampilkan. Gambar 3.17 menunjukkan rancangan dari antarmuka manajemen *log*. Berikut adalah penjelasan untuk nomor yang tertera pada Gambar 3.17

1. Label jenis uji yang disimpan pada *log*.
2. Label tanggal dan waktu dari sistem.
3. Label nilai hasil uji yang dilakukan.
4. Tombol Test untuk pindah ke antarmuka uji makanan
5. Tombol Log untuk mengganti *log* ke *log* lain yang disimpan. Log yang ditampilkan berikutnya adalah *log* yang disimpan sebelum *log* yang sedang ditampilkan.
6. Tombol play untuk menghapus *log* yang ditampilkan.
7. Tombol Shutdown untuk berpindah ke antarmuka matikan sistem.
8. Label notifikasi untuk setiap kejadian yang terjadi.

3.2.3.3 Perancangan Antarmuka Hasil Uji Makanan



Gambar 3.18. Rancangan Antarmuka Hasil Uji Makanan

Antarmuka ini ditampilkan ketika pengguna telah memilih jenis uji dan melakukan pengujian pada antarmuka pengujian makanan. Antarmuka ini menampilkan hasil dari perbandingan sampel sesuai dengan jenis uji yang dipilih. Gambar 3.18 menunjukkan rancangan dari antarmuka yang akan dibuat. Berikut adalah penjelasan dari setiap nomor yang tertera pada Gambar 3.18.

1. Label dari hasil uji yang ditampilkan.
2. Label notifikasi untuk setiap kejadian yang terjadi.
3. Tombol Test untuk pindah ke antarmuka pengujian makanan
4. Tombol Log untuk pindah ke antarmuka manajemen *log*.
5. Tombol Play untuk menyimpan hasil uji ke *log*
6. Tombol Shutdown untuk pindah ke antarmuka matikan sistem

3.2.3.4 Perancangan Antarmuka Matikan Sistem

Gambar 3.19. Rancangan Antarmuka Matikan Sistem

Antarmuka ini ditampilkan ketika tombol Shutdown ditekan dari antarmuka lainnya. Antarmuka ini memberikan konfirmasi

untuk perintah mematikan sistem. Gambar 3.19 menunjukkan rancangan dari antarmuka yang akan dibuat.

Berikut adalah penjelasan untuk nomor yang tertera pada Gambar 3.19.

1. Label notifikasi untuk menkonfirmasi akan mematikan sistem
2. Tombol Test untuk pindah ke antarmuka pengujian makanan
3. Tombol Log untuk pindah ke antarmuka manajemen *log*.
4. Tombol Play tidak aktif.
5. Tombol Shutdown untuk mematikan system.

BAB IV
IMPLEMENTASI

Bab ini membahas tentang implementasi dari perancangan sistem. Bab ini berisi proses dari implementasi arsitektur perangkat keras, implementasi kelas pada setiap modul dan implementasi antarmuka rancangan aplikasi..

4.1 Implementasi Arsitektur Sistem

Pada Sub bab ini akan dijelaskan implementasi dari rancangan arsitektur sistem yang telah dibuat. Implementasi meliputi daftar perangkat keras yang digunakan untuk membangun sistem ,proses perakitan dari perangkat keras yang digunakan dan implementasi dari setiap modul aplikasi.

4.1.1 Perangkat Keras Sistem

Sejumlah perangkat keras untuk membangun sistem yang diinginkan diantaranya adalah kerangka wadah penampung sistem, komputer kecil, lampu LED, layar display, dan kamera.

4.1.1.1. Raspberry Pi Model B *Revision* 2

Gambar 4.1. Raspberry Pi Model B Revision 2 yang Digunakan

Komputer kecil yang digunakan adalah Raspberry Pi *model* B *revision* 2. Pada Raspberry Pi akan digunakan sistem operasi *Raspbian*. *Raspbian* merupakan sistem operasi ringan yang dikembangkan dari *distro* Debian. Gambar 4.1 adalah tampilan dari Raspberry Pi Model B Revision 2. Raspberry Pi akan berperan sebagai inti dari sistem. Raspberry Pi juga memiliki GPIO(*General Process Input Output*) yang dapat digunakan untuk menambah berbagai macam perangkat lain seperti *display* dan lampu. Semua data termasuk sistem operasi disimpan dalam SD *card*.

4.1.1.2. Wadah Akrilik Hitam

Wadah penampung dibuat dengan berbahan dasar akrilik berwarna hitam. Wadah digunakan untuk menjaga kondisi cahaya agar tidak berubah-ubah. Wadah akan meminimalkan cahaya yang masuk sehingga sumber cahaya berasal dari lampu. Hal ini digunakan untuk mendapatkan warna natural dari gambar sampel yang diambil.

4.1.1.3. Kamera Raspberry Pi

Gambar 4.2. Kamera Raspberry Pi yang Digunakan

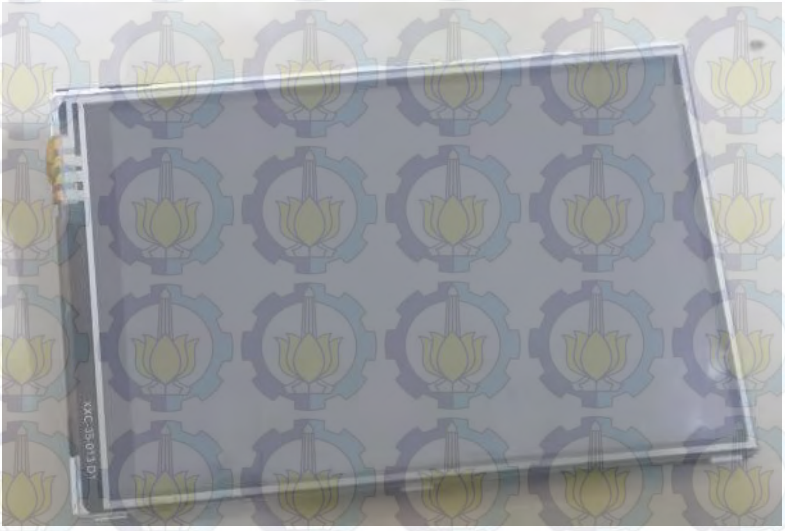
Kamera yang digunakan adalah modul kamera dari Raspberry Pi. Kamera ini memiliki spesifikasi ukuran hinga 5 *megapixel*. Kamera digunakan untuk mengambil gambar dari sampel yang akan diuji. Hasil gambar akan disimpan pada Raspberry Pi. Gambar 4.2 merupakan kamera Raspberry Pi yang digunakan.

4.1.1.4. Lampu LED putih

Lampu digunakan untuk memberikan cahaya pada kondisi wadah yang memiliki cahaya minimum. Lampu yang digunakan berwarna putih untuk menghindari perubahan warna pada sampel yang signifikan. Digunakan sebanyak dua buah lampu yang terhubung dengan kabel *jumper female to female*. Salah satu ujung kabel akan terpasang pada lampu dan ujung yang lain akan terpasang pada GPIO Raspberry Pi.

4.1.1.5. Display Waveshare LCD TFT 3.5" + Layar Sentuh

Layar *display* yang digunakan berukuran 3,5 inci dengan kapabilitas layar sentuh. LCD ini akan menampilkan antarmuka dari aplikasi sistem. Penggunaan LCD dengan layar sentuh menghilangkan kebutuhan penambahan perangkat lain untuk interaksi pengguna dengan aplikasi. Gambar 4.3 merupakan LCD yang digunakan.



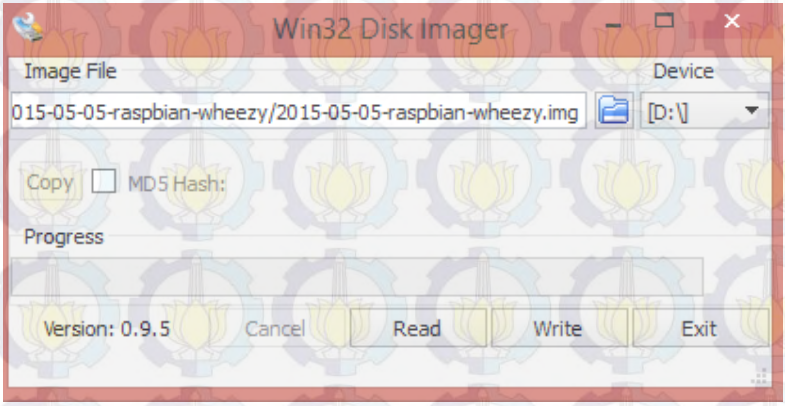
Gambar 4.3. Layar Sentuh LCD TFT Waveshare 3.5 inci

4.1.2 Perakitan Sistem

Bagian ini akan menjelaskan bagaimana semua perangkat keras yang digunakan akan dirakit. Dibutuhkan juga beberapa modul tambahan yang harus diinstalasi pada Rasberry pi untuk mendukung pemasangan dari perangkat tambahan

4.1.2.1 Instalasi Sistem Operasi Raspbian

Sistem operasi yang digunakan adalah Raspbian dengan versi *kernel* 3.18 dan di keluarkan pada tanggal 16-02-2015. File *stand-alone image* untuk Raspbian ini dapat diunduh pada link berikut *https://www.raspberrypi.org/downloads/* dalam bentuk *zip* atau *file torrent*.

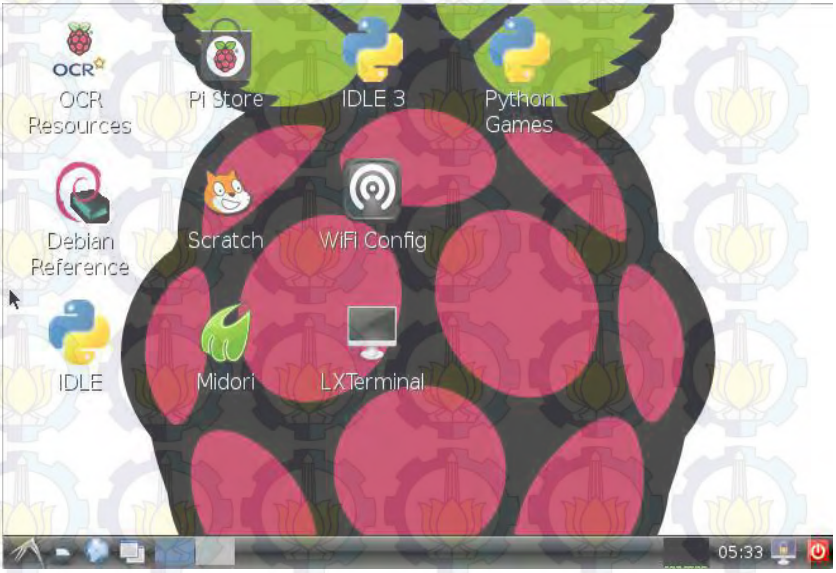


Gambar 4.4. Penanaman Sistem Operasi Raspbian ke SD Card dengan Win32DiskImager

SD *card* yang digunakan adalah Sandisk berukuran 8 GB lalu pasang pada komputer dengan alat pembaca SD card. Ekstrak *file zip* yang berisi *file image*. Untuk memindahkan *file image* ini ke SD *card* digunakan aplikasi Win32DiskImager. Win32DiskImager ini hanya untuk pengguna Windows. Jalankan Win32DiskImager sebagai admin lalu pilih lokasi *file image* Raspbian yang telah

diunduh tadi dan juga pilih lokasi dari SD Card yang telah dimasukan sesuai pada gambar 4.4.

Jendela yang muncul akan memiliki 2 panel Image File untuk memilih target file image dan panel Progress menunjukkan proses yang berjalan. Tekan tombol Read pada panel Progress untuk memindahkan *file image* ke SD *card*. Pasang SD *Card* di Raspberry Pi dan pasang kabel HDMI, lalu sambungkan ke listrik. Tampilan yang pertama kali keluar berupa konsol. Akan keluar perintah untuk mengatur konfigurasi pada pertama kali Raspberry Pi dinyalakan, lewati dengan memilih *finish*. *Raspbian* memiliki LXDE(*Lightweight X11 Desktop Environment*) bawaan sehingga memiliki tampilan antarmuka desktop. Untuk masuk ke desktop maka *log*in dengan username "pi" dan *password* "raspberry" sebagai akun *default*. Lalu jalankan perintah startx pada terminal untuk memunculkan *desktop* LXDE. Gambar 4.5 menunjukkan tampilan yang keluar.



Gambar 4.5. Tampilan Desktop Raspbian

4.1.2.2 Instalasi Modul Kamera Raspberry Pi

Modul kamera Raspberry Pi dapat mengambil gambar dengan resolusi hingga 1080 *pixel* dan keluaran video dapat diatur secara pemrograman. Kamera memiliki kabel *flex* yang dipasang pada konektor di Raspberry Pi. Untuk menggunakan kamera ubah konfigurasi dengan menjalankan perintah raspi-config lalu pilih enable camera dan tekan enter. Mengambil gambar dapat dilakukan dengan perintah pada Gambar 4.6.

```
raspistill –o /direktori tujuan/namafile.format
```

Gambar 4.6. Contoh Format Perintah Raspistill

Perintah raspistill untuk menginisiasi perintah kamera. –o adalah target keluaran dari kamera. /direktori_tujuan/namafile.format adalah path dari gambar yang disimpan dengan format yang dipilih.

4.1.2.3 Instalasi Layar LCD

Layar LCD TFT Waveshare menggunakan semua pin pada P1 header yang terdapat di Raspberry Pi model A dan B. Pada umumnya penyedia layar LCD akan memberikan *file* image berupa sistem operasi yang telah kompatibel dengan layar yang disediakan [11]. Namun pada implementasi ini akan digunakan sistem operasi *Raspbian* yang telah diinstal sebelumnya. Oleh karenanya diperlukan beberapa konfigurasi dan instalasi modul untuk layar LCD [12]. Driver yang digunakan untuk layar LCD ini adalah driver fbtft yang tersedia pada repositori ini *https://github.com/notro/fbtf*t.

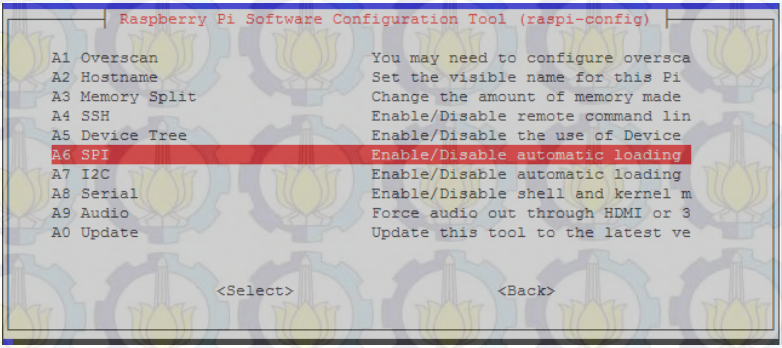
Langkah pertama dalam menginstall driver fbtft adalah dengan mengganti keluaran tampilan Raspberry Pi dari HDMI ke SPI(Serial Peripheral Interface). Dengan merubah baris konfigurasi dari *file* yang terdapat pada /usr/share/X11/xorg.conf.d/99-

fbturbo.conf. Pada baris dengan option “fbdev” ganti fb0 dengan fb1. Isi dari *file* tersebut sekarang sesuai dengan Gambar 4.7.

```
Section "Device"
        Identifier      "Allwinner A10/A13 FBDEV"
        Driver          "fbturbo"
        Option          "fbdev" "/dev/fb1"

        Option          "SwapbuffersWait" "true"
EndSection
```

Gambar 4.7. Pengaturan pada *File* 99-fbturbo.conf

Berikutnya mengaktifkan SPI dengan membuka perintah raspi-config lalu pilih Advanced Option. Pilih opsi SPI lalu pilih aktifkan sesuai dengan Gambar 4.8.



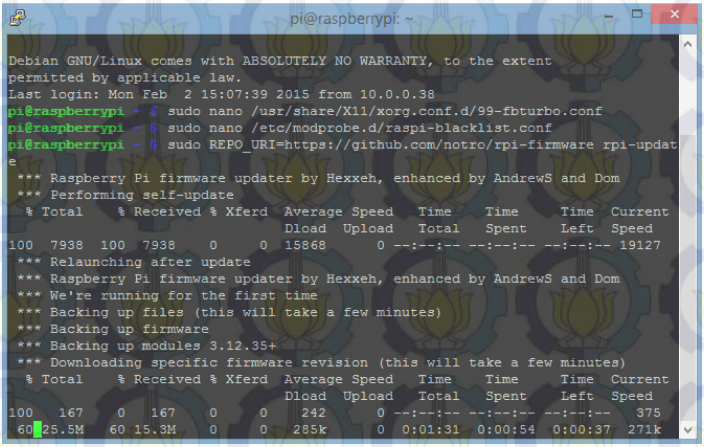
Gambar 4.9. Konfigurasi SPI pada raspi-config

```
sudo REPO_URI=https://github.com/notro/rpi-firmware
```

Gambar 4.8. Perintah Mengunduh Modul *Kernel*

Sekarang driver LCD dapat diunduh dan jalankan perintah pada Gambar 4.8 pada *terminal*. Perintah tersebut akan mengunduh modul kernel dari layar dan menambahkan *driver* fbtft yang dibutuhkan. Jika akses internet menggunakan *proxy* maka set variabel *proxy* terlebih dahulu. *Reboot* Raspberry Pi dengan

menjalankan perintah sudo reboot. Gambar 4.10 menunjukkan *driver* telah berhasil diunduh.



Gambar 4.10. Pengunduhan Driver fbtft Berhasil

```
flexfb width=480 height=320 regwidth=16 init=-
1,0xb0,0x0,-1,0x11,-2,250,-1,0x3A,0x55,-1,0xC2,0x44,-
1,0xC5,0x00,0x00,0x00,0x00,-
1,0xE0,0x0F,0x1F,0x1C,0x0C,0x0F,0x08,0x48,0x98,0x37,0x
0A,0x13,0x04,0x11,0x0D,0x00,-
1,0xE1,0x0F,0x32,0x2E,0x0B,0x0D,0x05,0x47,0x75,0x37,0x
06,0x10,0x03,0x24,0x20,0x00,-1,
0xE2,0x0F,0x32,0x2E,0x0B,0x0D,0x05,0x47,0x75,0x37,0x06
,0x10,0x03,0x24,0x20,0x00,-1,0x36,0x28,-1,0x11,-
1,0x29,-
fbtft_device debug=3 rotate=0 name=flexfb
speed=16000000 gpios=reset:25,dc:24.
```

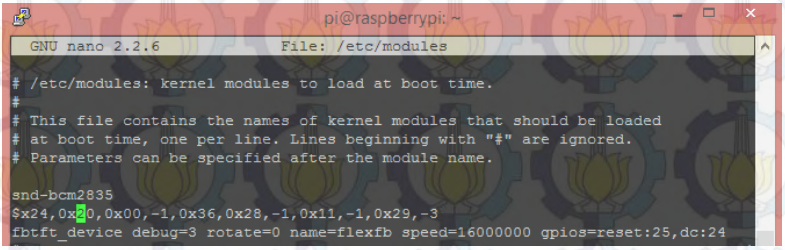
Gambar 4.11. Konfigurasi pada *File* /etc/modules

Setelah *driver* diunduh konfigurasi dapat dilakukan dengan mengubah *file* pada /etc/modules. Pada *file* ini tambahkan satu baris perintah pada Gambar 4.11.

Kode flexfb adalah nama dari jenis monitor yang digunakan dan kode seterusnya adalah pengaturan sesuai dengan spesifikasi

monitornya yang terdapat pada daftar di laman https://github.com/notro/fbtft/blob/master/fbtft_device.c [13].

fbtft_device adalah modul kernel yang digunakan untuk pengaturan layar LCD. Gambar 4.11 menunjukkan *file* /etc/modules setelah dikonfigurasi. LCD sudah dapat menampilkan desktop pada tahap ini.



Gambar 4.12. Isi file /etc/module

Terdapat kekurangan dimana modul untuk ads7846_device yang merupakan modul layar sentuh tidak lagi terdapat pada modul kernel yang diberikan. Mengakibatkan layar sentuh tidak dapat digunakan. Oleh karenanya diperlukan pengaturan untuk DTOverlay(*Device Tree Overlay*) pada /boot/config.txt [14]. DTOverlay mendukung untuk banyak konfigurasi perangkat keras dengan satu kernel tanpa perlu menambahkan modul kernel lain secara eksplisit. Tambahkan baris kode sumber 4.12 pada *file* /boot/config.txt di baris paling bawah.

```
dtoverlay=ads7846,speed=1000000,penirq=17,swapxy=1,xmin=2
00,xmax=3900,ymin=200,ymax=3900
```

Gambar 4.13. Konfigurasi DTOverlay untuk Modul ads784

Variabel DTOverlay menunjukan modul yang ingin digunakan. Nilai penirq merupakan pin yang digunakan untuk layar sentuh yaitu pin nomor 17. Gambar 4.14 menunjukan konfigurasi pada /boot/config.txt untu pengaturan DTOverlay.

Gambar 4.14. Konfigurasi pada /boot/config.txt

Pada tahap ini layar dapat menampilkan antarmuka dan layar sentuh bekerja. Namun layar sentuh perlu dikalibrasi lagi dengan *xiput_calibrator*. *Xinput_calibrator* merupakan sebuah kakas bantu untuk kalibrasi layar sentuh dengan memberikan keluaran berupa konfigurasi yang diperlukan untuk layar sentuh lebih akurat. Pertama unduh modul yang menjadi dependensi dari *xinut_calibrator* dengan menjalankan perintah Gambar 4.15 pada *terminal*.

```
sudo apt-get install libx11-dev libxext-dev libxi-dev
x11proto-input-dev
```

Gambar 4.15. Perintah untuk Instalasi Modul Dependensi xinput_calibrator

```
wget
http://github.com/downloads/tias/xinput_calibrator/xinput_
calibrator-0.7.5.tar.gz.
```

Gambar 4.16. Perintah untuk Mengunduh xinput_calibrator

```
tar xzvf xinput_calibrator-0.7.5.tar.gz dan cd
xinput_calibrator-0.7.5/ .
```

Gambar 4.17. Perintah untuk Mengekstrak *File* xinput_calibrator-0.7.5.tar.gz

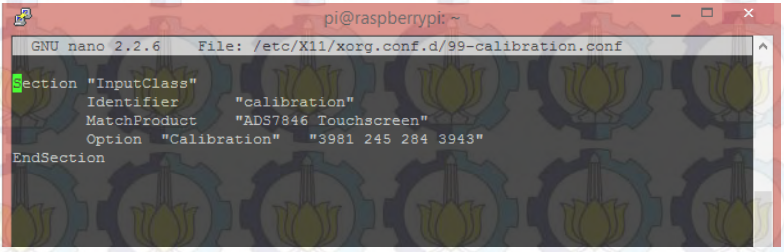
```
./configure
make
sudo make install
```

Gambar 4.18. Perintah untuk Instalasi xinput_calibrator

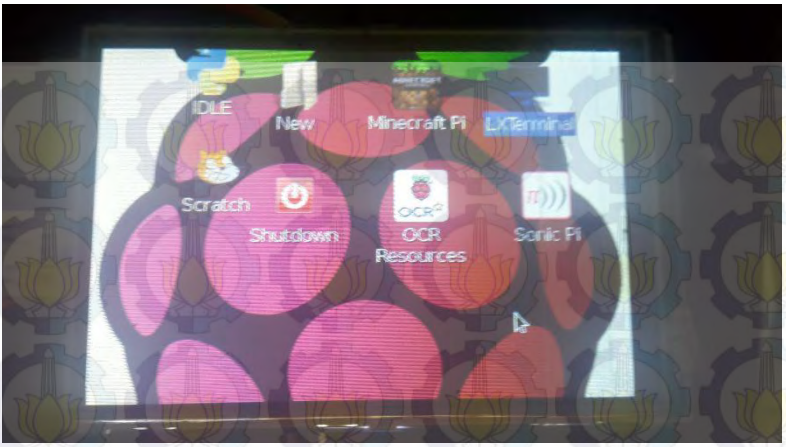
```
DISPLAY=":0.0" /usr/local/bin/xinput_calibrator
```

Gambar 4.19. Perintah untuk Kalibrasi dengan xinput_calibrator

Lalu unduh *xinput_calibrator* dengan menjalankan perintah Gambar 4.16 pada *terminal*. .Ekstrak lalu pindah kedalam direktori yang telah diekstrak dengan perintah kode Gambar 4.17 pada *terminal*. *Install* modul yang telah diunduh dengan menjalankan perintah Gambar 4.18 pada *terminal*. Jalankan xinput_calibrator dengan menjalankan perintah Gambar 4.18 Proses kalibrasi akan berlangsung dan ikuti instruksi yang diberikan. Keluaran yang dihasilkan oleh kakas ini ditambahkan pada *file* /etc/X11/xorg.conf.d/99-calibration.conf [15].



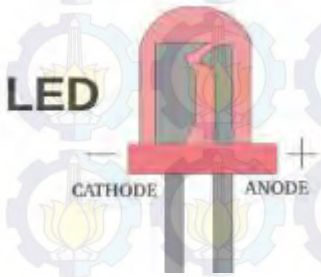
Gambar 4.20. Konfigurasi pada File 99-calibration.conf



Gambar 4.21. Tampilan Desktop Raspbian dari Layar LCD

Gambar 4.20 menunjukkan hasil dari *file* yang telah ditambahkan dengan konfigurasi dengan *xinput_calibrator*. Gambar 4.21. Menunjukkan layar sentuh dari LCD TFT Waveshare telah dapat digunakan.

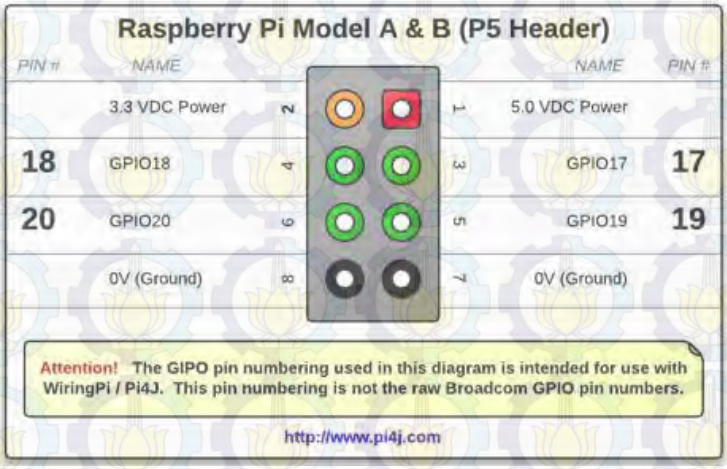
4.1.2.4 Instalasi Lampu LED



Gambar 4.22. Model Lampu LED

Lampu LED yang digunakan sebanyak dua buah dengan warna lampu berwarna putih. Lampu dipasang dengan kabel *female to female* dengan salah satu ujungnya terpasang dengan lampu dan lainnya terpasang dengan GPIO. Setiap kabel terhubung dengan

cathode dan *anode* dari lampu. *cathode* dan *anode* dari lampu ditunjukkan pada Gambar 4.22.



Gambar 4.23. Posisi Pin pada P5 Header

Cathode dari lampu dihubungkan dengan kabel ke *ground* dan *anode* dihubungkan dengan pin untuk GPIO dengan nomor berapapun. Namun karena seluruh pin pada papan utama Raspberry Pi telah digunakan layar LCD maka pin pada P5 Header digunakan. P5 Header merupakan sejumlah pin tambahan yang terdapat pada Raspberry Pi model A dan B. Gambar 4.23 menunjukkan peta dari P5 Header dan penomorannya. Pada implementasi sistem ini pin GPIO nomor 17 dan 18 yang digunakan.

4.1.2.5 Instalasi Sistem pada Wadah

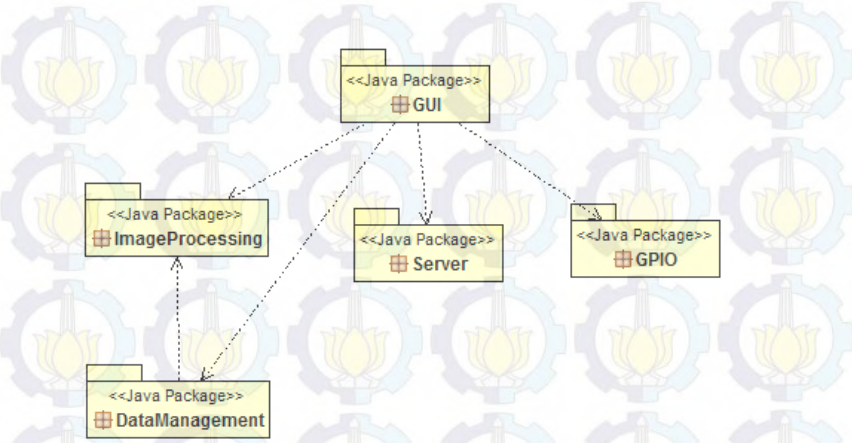
Wadah yang dibuat berbahan akrilik dan berwarna dasar hitam. Desain badan utama dari wadah berbentuk prisma segi empat terbuka di atas dengan panjang 25 cm , lebar 9 cm dan tinggi 7.5 cm. Tutup dari wadah berukuran 26 cm , lebar 9.5 cm dan tinggi 2 cm dengan lubang berbentuk lingkaran berdiameter 2 cm dan segi empat berukuran panjang 9 cm dan lebar 6 cm.

Gambar 4.24. Wadah dari Sampel dalam Wadah Utama

Gambar 4.25. Bentuk Fisik dari Sistem Secara Keseluruhan

Terdapat beberapa kendala diantaranya dalam beberapa uji warna bening direpresentasikan dengan warna putih pada warna referensi dan juga cahaya langsung dari lampu dapat menghilangkan warna natural yang ingin diperoleh. Oleh karenanya dibutuhkan wadah untuk tabung reaksi yang dapat memberikan warna latar putih dan menyebarkan cahaya untuk mendapatkan warna natural. Pada implementasi ini digunakan plastik putih sebagai wadah untuk tabung reaksi sesuai padaGambar 4.24. Gambar 4.25 menunjukkan keseluruhan dari arsitektur perangkat keras sistem yang telah dibangun. Terdapat tutup berbahan akrilik yang menutup lubang tabung reaksi. Hal ini untuk mengurangi jumlah cahaya yang masuk.

4.2 Implementasi Rancangan Modul Aplikasi Sistem



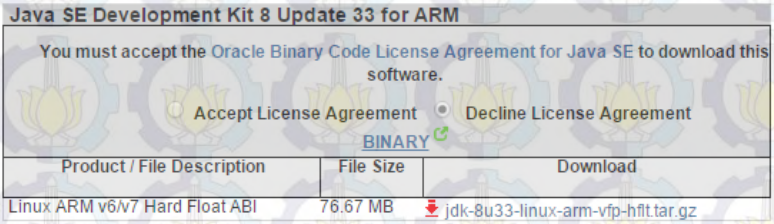
Gambar 4.26. Dependensi Antar Modul pada Implementasi Sistem

Pada subbab ini akan deijelaskan pembuatan rancangan aplikasi sistem. Aplikasi yang dibangun berbasis *desktop* yang akan berjalan ketika sistem dinyalakan. Daftar implementasi yang dilakukan meliputi instalasi lingukan kerja perangkat lunak, modul antarmuka, manajemen data, pemrosesan gambar dan modul GPIO. Hubungan dan dependensi antar modul setelah diimplementasikan sesuai dengan Gambar 4.26.

4.2.1 Instalasi Lingkungan Perangkat Lunak

Bahasa pemrograman yang digunakan untuk aplikasi sistem adalah java. Oleh karenanya dibutuhkan JRE(*Java Runtime Environment*) dan JDK(*Java Developer Kit*) terpasang pada Raspberry Pi. Pada implementasi sistem ini JDK 8 digunakan untuk mengembangkan aplikasi sistem. Raspberry Pi berjalan pada CPU(*Central Processing Unit*) ARM(*Advanced RISC Machines*)

dan menggunakan sistem operasi linux *Raspbian*. Sehingga jenis jdk yang digunakan adalah JDK 8 untuk linux ARM. Untuk menggunakan JDK 8 hal yang pertama dilakukan adalah mengunduh JDK tersebut di laman oracle yang menyediakan JDK pada link http://www.oracle.com/technetwork/java/javase/downloads/jdk8-arm-downloads-2187472.html. Pilih untuk mengunduh jdk-8u33-linux-arm-vfp-hflt.tar.gz sesuai dengan Gambar 4.27.



Gambar 4.27. Lokasi Pengunduhan *File* JDK 8 untuk ARM

```
sudo tar zxvf jdk-8-linux-arm-vfp-hflt.tar.gz -C /opt
```

Gambar 4.28. Perintah untuk Mengekstrak *File*

```
sudo update-alternatives --install /usr/bin/java java
/opt/jdk1.8.0/bin/java 1 dan sudo update-alternatives --
install /usr/bin/javac javac /opt/jdk1.8.0/bin/javac 1.
/opt
```

Gambar 4.29. Perintah untuk Mengatur Variabel Lingkungan PATH Java dan Javac

```
sudo update-alternatives --config java
sudo update-alternatives --config javac
```

Gambar 4.30. Perintah untuk Konfigurasi Java dan Javac

Setelah *file* diunduh *login* ke Raspberry Pi dan pindahkan *file* tersebut lalu ekstrak ke direktori ***/opt***. perintah untuk mengekstrak *file* tersebut sesuai dengan Gambar 4.28. Lalu set variabel untuk PATH dari Java dan Javac dengan menjalankan perintah pada

Gambar 4.29. Perbarui juga konfigurasi untuk java dan javac dengan menjalankan perintah pada Gambar 4.30.

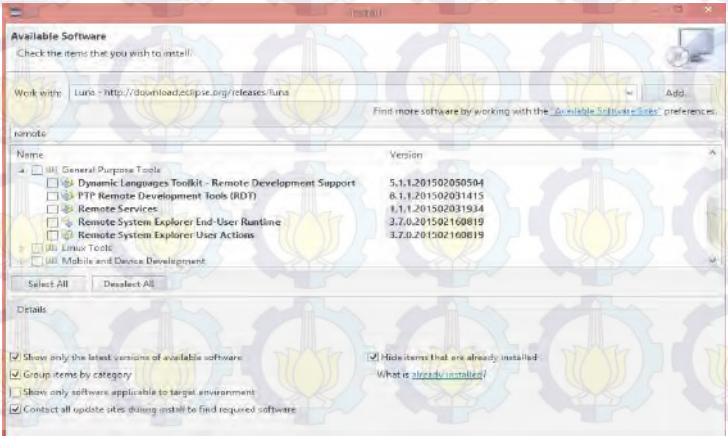
```
java -version
javac -version
```

Gambar 4.31. Perintah untuk Menampilkan Versi Java dan Javac

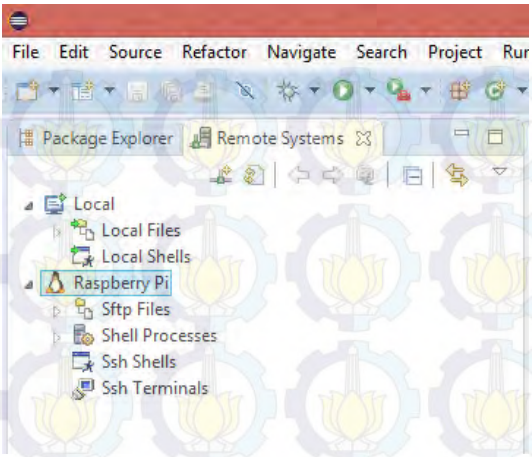
```
pi@raspberrypi ~ $ java -version
java version "1.8.0_33"
Java(TM) SE Runtime Environment (build 1.8.0_33-b05)
Java HotSpot(TM) Client VM (build 25.33-b05, mixed mode)
pi@raspberrypi ~ $ javac -version
javac 1.8.0_33
pi@raspberrypi ~ $
```

Gambar 4.32. Versi dari Java dan Javac

Java dan Javac adalah perintah dalam proses pembangunan(*build*) dan kompilasi(*compile*) dari kode sumber yand dibuat. untuk memverifikasi Java dan Javac yang digunakan perintah pada Gambar 4.31 akan menampilkan versi yang digunakan seperti pada Gambar 4.32.

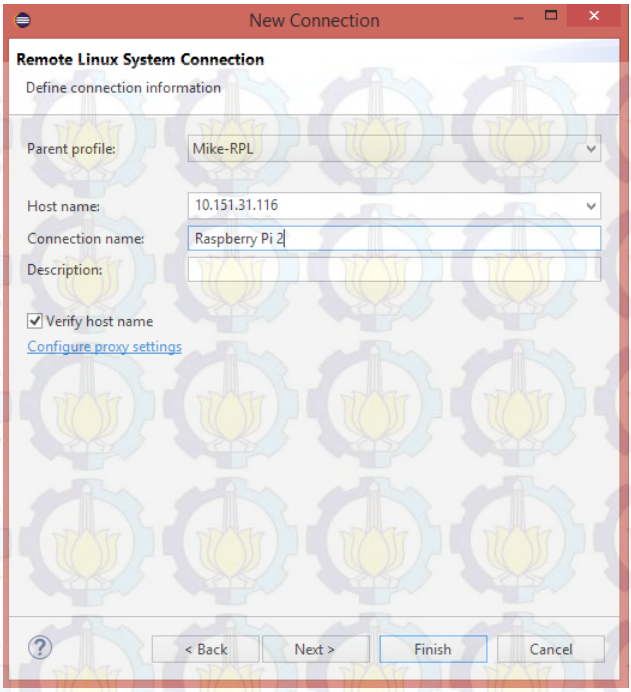


Gambar 4.33. Instalasi RSE pada Eclipse



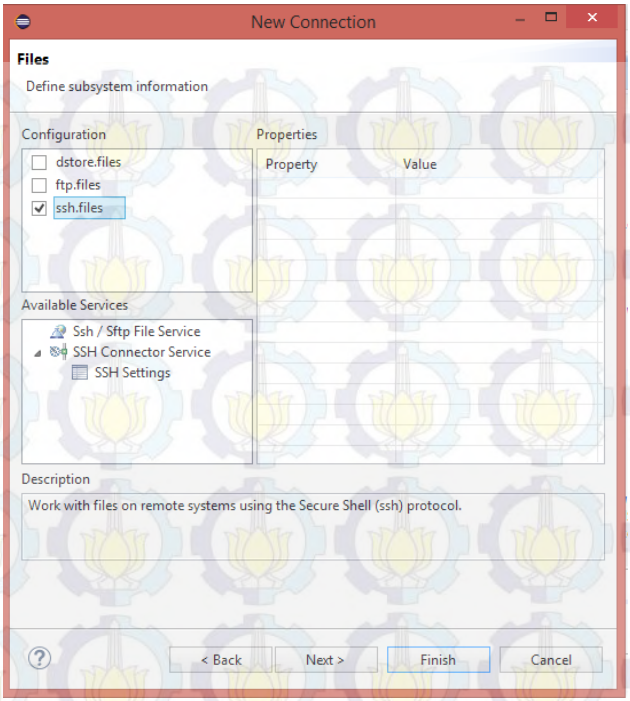
Gambar 4.34. Tampilan RSE pada Eclipse

Sistem dikembangkan menggunakan Eclipse IDE(*Integrated Development Environment*) pada komputer lain yang dihubungkan dengan Raspberry Pi sistem melalui RSE(*Remote System Explorer*). RSE memungkinkan pengiriman *file* antara 2 komputer yang terhubung melalui protokol SFTP(*SSH File Transfer Protocol*). Eclipse menyediakan RSE sebagai plugin yang dapat diunduh melalui dengan membuka *menu bar* Help lalu pilih Install New Software. Pilih *repository plugin* dan cari Remote System Explorer sesuai dengan Gambar 4.33 lalu *restart* Eclipse. Untuk menggunakan RSE pilih *menu bar* Window lalu pilih menu Show View lalu pilih Other lalu pilih Remote System. Gambar 4.34 menunjukkan RSE plugin telah dapat digunakan.



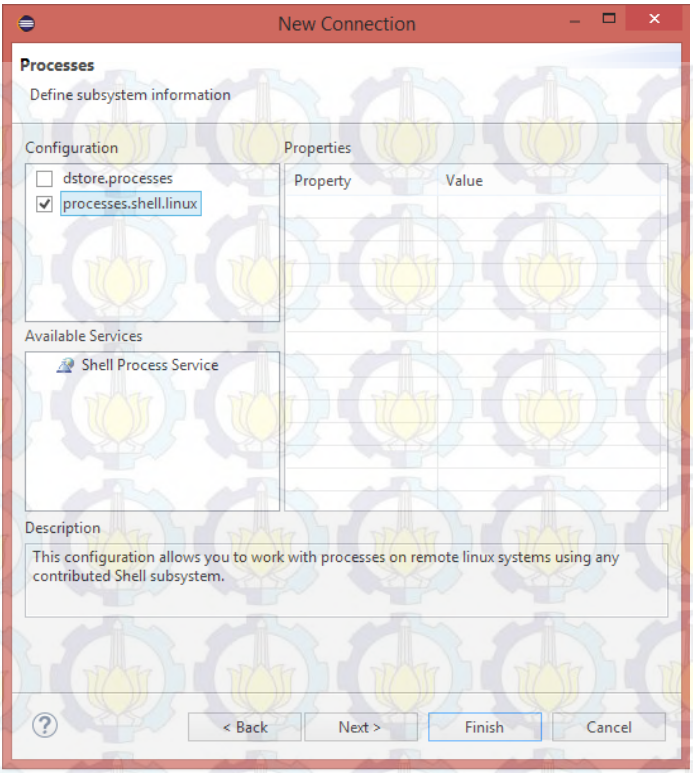
Gambar 4.35. Menambah Koneksi dengan RSE

Buat koneksi baru dengan Raspberry Pi. Pada *tab* Remote Systems klik kanan pada label Local lalu pilih menu New lalu pilih Connection. Jendela yang tampil akan menunjukkan sejumlah text box untuk mendeskripsikan alamat IP target, nama koneksi dan deskripsi dari koneksi. Isi alamat IP dari Raspberry Pi dan nama koneksi seperti pada Gambar 4.35 lalu pilih tombol Next. Alamat IP dapat diketahui dengan perintah ifconfig pada *terminal* Raspberry Pi.



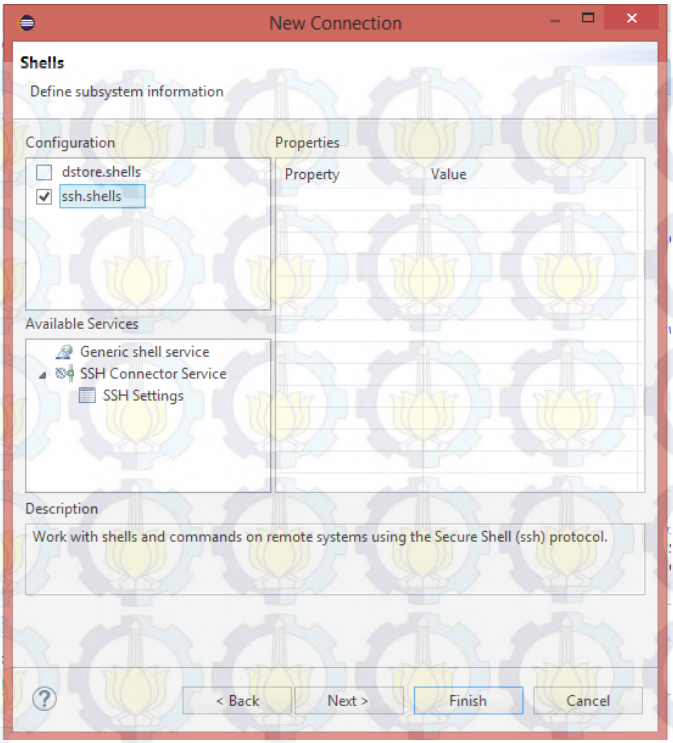
Gambar 4.36. Konfigurasi Jenis Protokol RSE yang Digunakan

Jendela yang tampil memiliki 3 panel diantaranya panel Configuration untuk memilih jenis protokol yang digunakan, Available Services untuk memilih *service* dari jenis protokol dan Properties menunjukkan properti dari jenis protokol yang dipilih. Pada opsi Configuration centang ssh.files lalu pilih tombol Next sesuai pada Gambar 4.36.



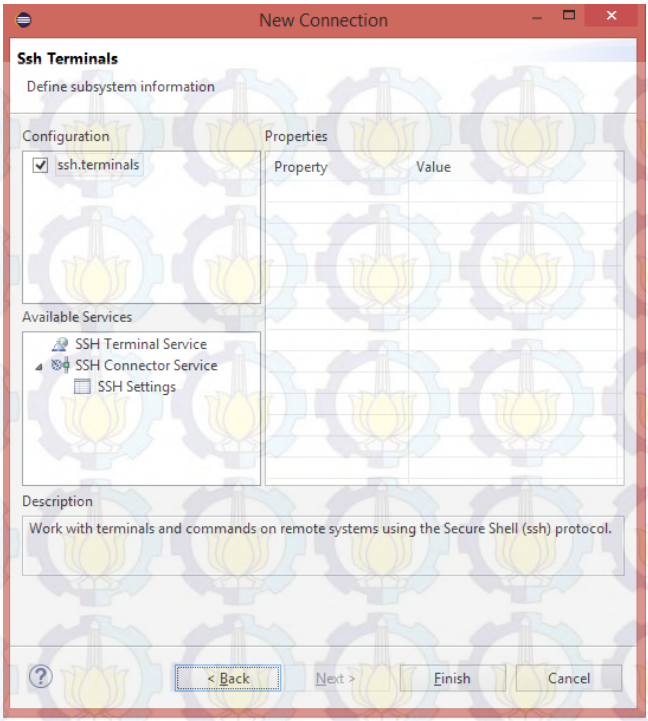
Gambar 4.37. Konfigurasi Sistem *Remote* Linux

Jendela yang tampil memiliki 3 panel diantaranya panel Configuration untuk memilih jenis proses yang akan digunakan terhadap linux yang di *remote*, panel Available Service untuk memilih jenis service dari proses dan panel Properties untuk menampilkan properti dari proese yang dipilih. Pada panel Configuration centang *processes.terminal.linux* lalu pilih tombol *Next* sesuai pada Gambar 4.37. Berikutnya adalah konfigurasi dari *shell command* yang digunakan pada Linux *remote*.



Gambar 4.38. Konfigurasi untuk Menggunakan *Shell* dan *Command* dengan SSH

Jendela yang tampil memiliki 3 panel diantaranya panel Configuration untuk memilih jenis dari shell command yang akan digunakan, panel Available Services menunjukkan daftar *service* yang tersedia, panel Properties menunjukkan properti dari *shell* yang dipilih. Pada panel Configuration centang ssh.*terminal*s. Lalu pilih tombol *Next* sesuai dengan Gambar 4.38. Berikutnya adalah pengaturan untuk *terminal* dengan menggunakan RSE. *Plugin* RSE juga menyediakan fasilitas untuk dapat membuka *terminal* dari Linux yang di *remote*.

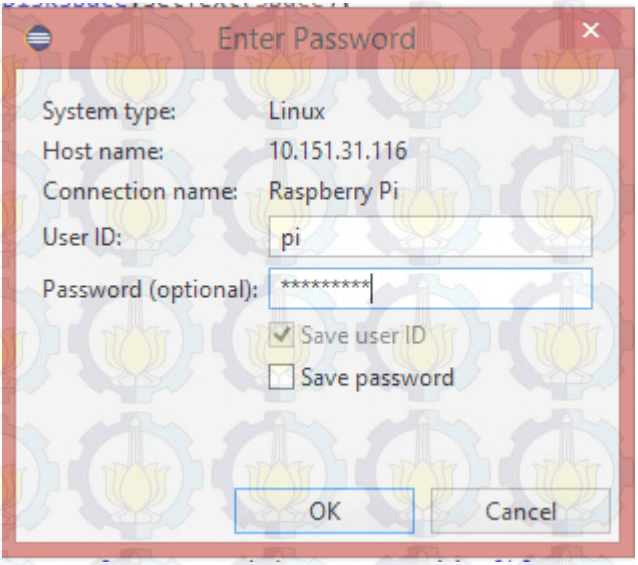


Gambar 4.39. Konfigurasi untuk Terminal dan *Command* dengan SSH

Jendela yang tampil akan memiliki 3 panel diantaranya panel Configuration untuk memilih *terminal*, panel Available Services untuk menampilkan daftar service dan panel Properties untuk menampilkan daftar properti dari *terminal*. Pada panel Configuration centang ssh.terminals. Lalu tekan tombol *finish* untuk menyelesaikan konfigurasi RSE sesuai dengan Gambar 4.39.

Koneksi baru berhasil dibuat dan dapat dilihat pada tab *Remote Systems* dengan nama koneksi yang telah diisi sebelumnya. Saat ini direktori dari Raspberry Pi yang menjadi target akan dapat diakses. Ketika *tab Home* dari Raspberry Pi tampilan *log*in akan muncul

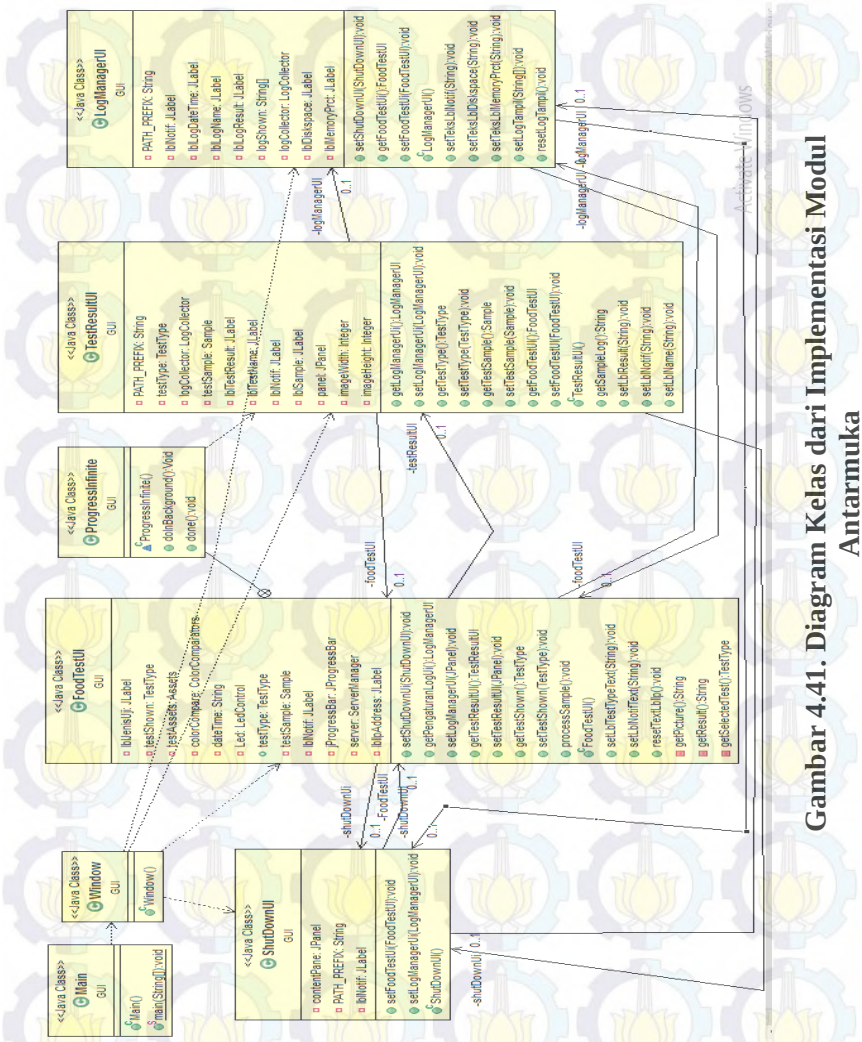
untuk masuk ke dalam *Rasberry Pi*. Jendela yang tampil berupa sejumlah label dan *text box* untuk mengisi *user ID* dan *password*. Gunakan akun sesuai yang digunakan sebelumnya, secara default *user ID* adalah pi dan *password* adalah "raspberry" sesuai dengan Gambar 4.40.



Gambar 4.40. Autentikasi untuk Terhubung dengan Raspberry Pi

Sistem dikembangakan dengan membangun dan mengkompilasi kode sumber pada *eclipse* lalu membuat *jar* dari dari project tersebut. *Jar* dan seluruh aset yang dibutuhkan dari program dipindahkan ke Raspberry Pi menggunakan RSE. Perintah pada *terminal* untuk menjalankan *file* jar adalah sudo java -jar [nama *file*].*jar* dengan dijalankan pada direktori tempat *file jar* berada.

4.2.2 Implementasi Modul Antarmuka

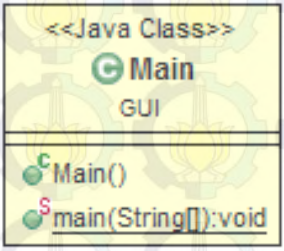


Gambar 4.41. Diagram Kelas dari Implementasi Modul Antarmuka

Setiap kelas antarmuka dari sistem dibuat dengan menggunakan Swing GUI(*Graphical User Interface*). Kelas-kelas dari antarmuka ini diantaranya adalah kelas Main, Window, FoodTestUI, LogManagerUI, TestResultUI dan ShutDownUI. Hubungan antar kelas ini pada satu modul ditunjukkan denagn Gambar 4.41.

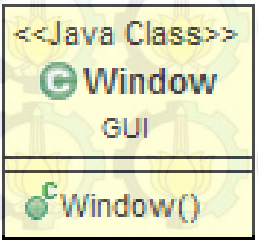
4.2.2.1 Kelas Main

Adalah kelas yang akan menginisiasi jalannya aplikasi. Pada kelas ini akan menginstansiasi kelas Window yang merupakan kerangka utama dari antarmuka aplikasi. Implementasi dari kelas ini sesuai dengan Gambar 4.42.



Gambar 4.42. Implementasi Kelas Main

4.2.2.2 Kelas Window



Gambar 4.43. Implementasi kelas Window

Kelas ini menurunkan kelas JFrame yang dimiliki Swing. JFrame adalah kelas yang menghasilkan antarmuka berupa *window* atau kerangka. Kelas ini menampung banyak JPanel yang berupa kelas FoodTestUI, LogManagerUI, TestResultUI dan ShutDownUI. Kelas-kelas JPanel tersebut akan diinstansiasi dalam konstruktor Window. Hubungan dan transisi setiap kelas JPanel tersebut akan disatur pada konstruktor Window. Implementasi pada kelas ini sesuai dengan Gambar 4.43.

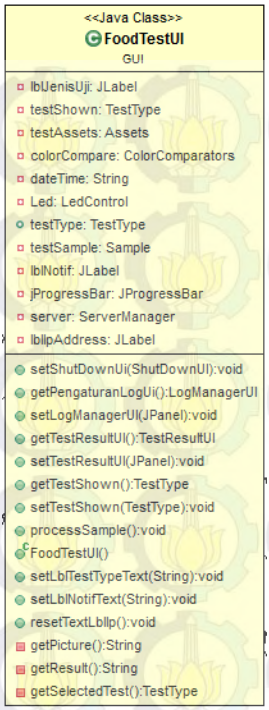
```
public Window(String ipAddress, String portNumber) {
        getContentPane().setLayout(new CardLayout(0, 0));

        FoodTestUI foodTestUI = new FoodTestUI(ipAddress,portNumber);
        LogManagerUI logManagementUI = new LogManagerUI();
        TestResultUI testResultUI = new TestResultUI();
        ShutDownUI shutDownUI = new ShutDownUI();
```

Kode Sumber 4.1. Konstruktor Kelas Window

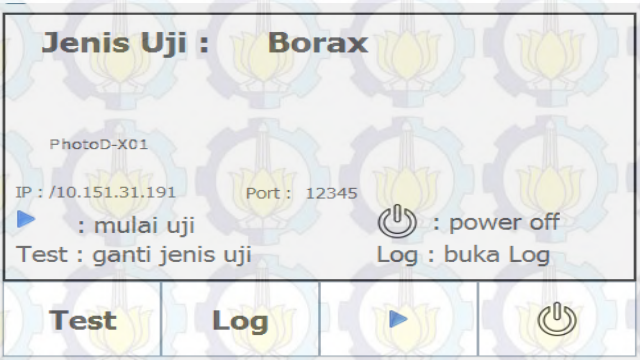
Kode Sumber 4.1 menunjukkan potongan kode sumber *constructor* dari kelas Window. Terdapat kelas FoodTestUI, LogManagerUI, TestResultUI dan ShutDownUI yang diinstansiasi di dalam kelas ini. Pengaturan transisi dari setiap kelas tersebut juga diatur dengan menyimpan objek dari kelas lain ke dalam kelas UI yang memiliki hubungan.

4.2.2.3 Kelas FoodTestUI

Kelas ini menurunkan kelas JPanel yang dimiliki *Swing*. JPanel adalah wadah tanpa kerangka yang dapat menyimpan berbagai objek antarmuka seperti label, dan tombol. Antarmuka ini merupakan antarmuka yang ditampilkan pertama kali oleh sistem ke pengguna. Tampilan ini digunakan untuk mengganti jenis uji dan menginisiasi pengujian yang ingin dilakukan.



Gambar 4.44. Implementasi dari Kelas FoodTestUI



Gambar 4.45. Implementasi Desain Antarmuka Kelas FoodTestUI

Implementasi dari kelas ini sesuai dengan Gambar 4.44. Instansiasi dari konstruktor kelas FoodTestUI akan menginstansiasi kelas Assets dan memuat semua aset dari jenis uji.
Transisi dari antarmuka ini diatur dengan menyimpan kelas yang telah diinstansiasi pada kelas Window. Antarmuka lain yang dapat di transisi dari kelas FoodTestUI adalah LogManagerUI, TestResultUI dan ShutDownUI. Desain antarmuka dari kelas ini ditunjukkan pada Gambar 4.45.

Gambar 4.46. Implementasi Progress Bar

Alamat IP dan nomor port yang digunakan oleh sistem tertera pada tampilan. Penguna dapat mengganti jenis uji yang dilakukan dengan menekan tombol Test. Tombol ini akan mengganti label teks lblJenisUji dengan jenis tes lain yang dimiliki kelas Assets dan juga menginisiasi kelas TestType dengan aset dari jenis uji yang dipilih. Tombol Log akan memindahkan panel ke panel manajemen *log*. Tombol ketiga atau tombol play akan memulai uji makanan sesuai jenis uji yang tampil. Ketika tombol play ditekan ProgressBar akan muncul ke layar seperti pada Gambar 4.46.

Hal ini disebabkan oleh kelas ProgressInfinite yang merupakan turunan dari kelas SwingWorker. Kelas ini digunakan agar kelas objek dari Swing yaitu ProgressBar dan JPanel TestResultUI yang dipanggil tetap bekerja secara parallel.

```
class ProgressInfinite extends SwingWorker<Void, Void> {
            @Override
        public Void doInBackground() {
            jProgressBar.setIndeterminate(true);
            processSample();

            return null;
        }
        @Override
        public void done()
        {
            jProgressBar.setIndeterminate(false);
            jProgressBar.setVisible(false);
            setVisible(false);
            testResultUI.setVisible(true);
        }
}
```

Kode Sumber 4.2. Kelas SwingWorker

Kode Sumber 4.2 menunjukkan potongan dari kode sumber pada kelas ProgressInfinite. Fungsi doInBackground() akan menjalankan isi dari fungsi ini di belakang layar dan fungsionalitas antarmuka Swing yang menampilkan ProgressBar. Fungsi yang dikerjakan di belakang ketika ProgressInfinite diinstansiasi adalah fungsi processSample(). Fungsi ini akan menginstansiasi kelas Sample dan memanggil fungsi lain untuk melengkapi properti dari kelas sampel tersebut. Setelah isi dari fungsi selesai kelas ini akan langsung memanggil fungsi done() dan menjalankan perintah didalamnya.

```
url ="./TestSample/"+filename;
Led.LedOn();
String command = "raspistill -vf -hf -o "+url+" -w 480 -h 360";
p =  Runtime.getRuntime().exec(command);
p.waitFor();
Led.LedOff();
command = "sudo chmod 777 "+url;
p = Runtime.getRuntime().exec(command);
p.waitFor();
```

Kode Sumber 4.3. Fungsi getPicture()

Fungsi pertama yang dipanggil adalah getPicture() dimana fungsi ini akan menjalankan perintah mengambil gambar menggunakan perintah raspistill dan mengembalikan path ke *file* dari gambar yang disimpan. Kode Sumber 4.3. **Fungsi getPicture()** merupakan potongan dari fungsi getPicture(). Variabel *url* adalah nilai yang akan dikembalikan setelah gambar diambil. Fungsi ini juga menyalakan lampu Led sebelum mengambil gambar dan juga mematikan lampu setelah gambar diambil. Perintah yang dijalankan ke *runtime* berikutnya adalah untuk merubah *permission* dari gambar yang diambil.

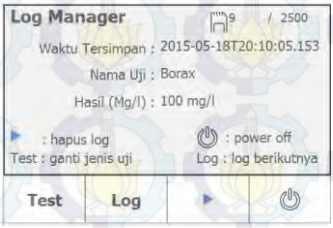
Lalu path dari *file* gambar akan digunakan pada properti dari kelas Sample. Fungsi berikutnya adalah getResult() yang mengembalikan nilai hasil perbandingan dari gambar sampel dengan gambar dari warna referensi. Kelas getResult() akan menginstansiasi kelas ColorComparators dengan parameter kelas Sample dan kelas TestType yang dipilih. Lalu dipanggil fungsi getCompareResult() dari kelas ColorComparators yang mengembalikan hasil perbandingan. Hasil perbandignan lalu dinisiasi ke property kelas Sample. Setelah itu kelas ProgressInfinite akan memanggil fungsi done() yang didalamnya dipanggil GUI TestResultUI dengan menginisiasi properti dari kelas Sample. Tombol keempat atau *Off* akan mengganti tampilan ke antarmuka ShutDownUI.

4.2.2.4 Kelas *LogManagerUI*

Kelas ini merupakan turunan dari kelas JPanel. Kelas ini berfungsi untuk mengatur tampilan yang dikeluarkan dalam mengatur *log* yang disimpan.

Gambar 4.47. Implementasi LogManagerUI

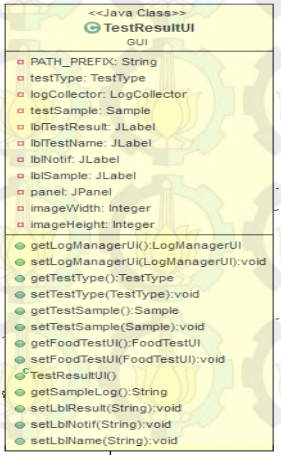
Kelas ini diimplementasikan dengan sesuai dengan Gambar 4.47. Transisi dari kelas ini meliputi perpindahan ke antarmuka FoodTestUI dan ShutDownUI. Setiap transisi diatur dengan menyimpan kelas antarmuka lain yang diinstansiasi di kelas Window.



Gambar 4.48. Implementasi Desain Antarmuka Manajemen Log

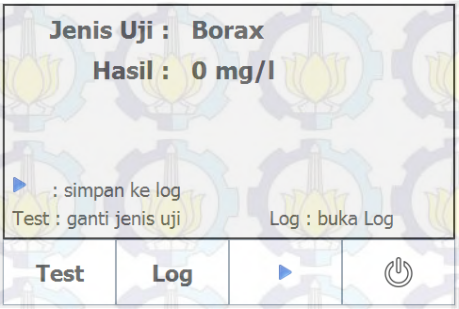
Desain antarmuka dari kelas ini ditunjukkan pada Gambar 4.48. Konstruktor dari kelas ini akan menginisiasi kelas LogCollector dan menyimpan *log* terakhir. *Log* yang pertama kali ditampilkan adalah *log* dari uji makanan yang terakhir dilakukan. *Log* yang disimpan berupa nama uji, tanggal waktu sistem menguji dan hasil dari pengujian. Ketika tombol Test ditekan maka tampilan akan berpindah ke antarmuka FoodTestUI, ketika tombol *Log* ditekan, *log* yang tampil akan berganti ke *log* yag disimpan setelah *log* yang tampil sebelumnya. Tombol Play digunakan untuk menghapus *log* yang tampil. Ketika tombol ini ditekan fungsi deleteLogByFileName() dan deleteImageSample() dari kelas *Log*Collector akan dipanggil. Kedua fungsi ini akan menghapus *log* dan menghapus gambar yang disimpan dari sampel. Tombol keempat atau Off akan mengganti tampilan ke antarmuka ShutDownUI.

4.2.2.5 Kelas TestResultUI



Gambar 4.49. Implementasi Kelas TestResultUI

Kelas ini merupakan turunan dari kelas JPanel. Kelas ini tampil setelah kejadian uji makanan dilakukan di antarmuka FoodTestUI. Kelas ini menampilkan hasil dari uji sampel yang dilakukan terhadap jenis uji yang dipilih. Implementasi dari kelas ini ditunjukkan pada Gambar 4.49. Transisi dari kelas ini meliputi perpindahan ke antarmuka FoodTestUI, LogManagerUI dan ShutDownUI. Setiap transisi diatur dengan menyimpan kelas antarmuka lain yang diinstansiasi di kelas Window. Desain antarmuka dari kelas ini ditunjukkan pada Gambar 4.50.



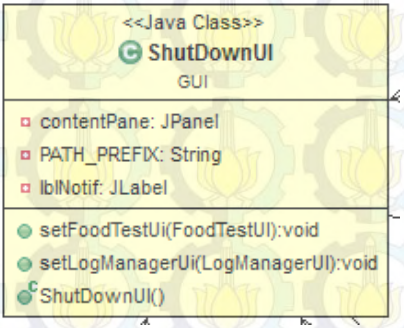
Gambar 4.50. Implementasi Desain Antarmuka Hasil Uji

Konstruktor dari kelas ini akan menginstansiasi kelas LogCollector. Tombol Test digunakan untuk berpindah ke tampilan kelas FoodTestUI. Tombol Log digunakan untuk berpindah ke tampilan kelas LogManagementUI. Tombol ketiga atau tombol play digunakan untuk menyimpan hasil uji dari sampel ke log. Ketika tombol play ditekan, fungsi appendLogFiles() dari kelas LogCollector akan dipanggil dengan parameter kelas Sample yang diinisiasi di kelas FoodTestUI. Tombol keempat atau Off akan mengganti tampilan ke antarmuka ShutDownUI.

4.2.2.6 Kelas ShutDownUI

Kelas ini adalah turunan dari kelas JPanel. Kelas ini menampilkan notifikasi dan memverifikasi perintah untuk

mematikan sistem. Implementasi dari kelas ini ditunjukkan pada Gambar 4.51.



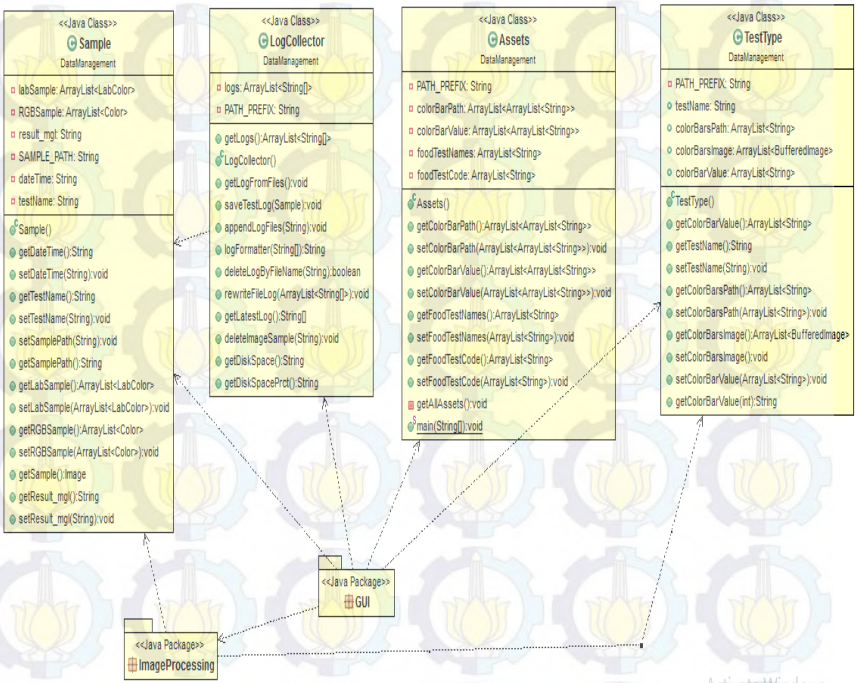
Gambar 4.51. Implementasi Kelas ShutDownUI

Gambar 4.52. Implementasi Desain Antarmuka Matikan Sistem

Transisi dari kelas ini meliputi perpindahan ke antarmuka FoodTestUI dan LogManagerUI . Setiap transisi diatur dengan menyimpan kelas antarmuka lain yang diinstansiasi di kelas Window. Desain antarmuka dari kelas ini ditunjukkan pada Gambar 4.52. Tombol Test digunakan untuk berpindah ke tampilan kelas FoodTestUI. Tombol Log digunakan untuk berpindah ke tampilan kelas LogManagementUI. Tombol ketiga atau tombol Play tidak

akan akan memberikan respon apapun. Tombol keempat atau Off akan mematikan sistem dengan menjalankan perintah shutdown ke *terminal*. Perintah ini adalah sudo shutdown –h 0.

4.2.3 Implementasi Modul Manajemen Data

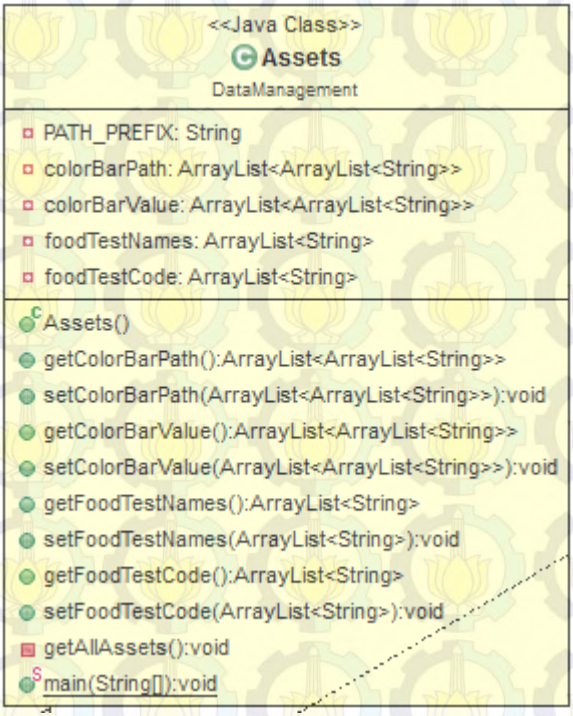


Gambar 4.53. Implementas Diagram Kelas Modul Manajemen Data

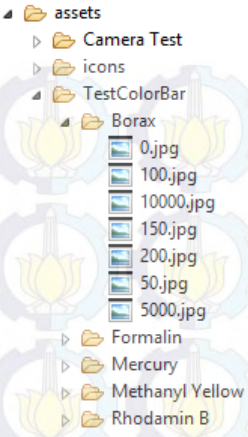
Modul ini berisi kelas-kelas yang digunakan untuk memuat aset dan *log*, juga kelas untuk merepresentasikan jenis uji dan sampel sebagai kelas dalam program. Kelas-kelas dalam modul ini diantaranya kelas Assets, *Log*Collector, Sample dan TestType. Hubungan antar kelas ini ditunjukkan pada Gambar 4.53

4.2.3.1 Kelas Assets

Kelas ini digunakan untuk menyimpan semua data dari aset sistem ke dalam aplikasi. Aset yang disimpan berupa daftar jenis uji dan gambar dari warna referensi setiap jenis uji. Gambar 4.54 menunjukkan implementasi dari kelas Assets.



Gambar 4.54. Implementasi Kelas Assets



Gambar 4.55. Hirarki Direktori Assets

Gambar 4.56. Contoh Gambar dari Warna Referensi

Semua gambar dari jenis uji disimpan dalam direktori /assets/TestColorBar/. Gambar 4.55 menunjukkan hirarki dari direktori assets. Setiap jenis uji dipilih maka gambar dari warna referensi jenis uji dan nilainya harus dimuat ke dalam program. *File config_path.txt* menyimpan semua path dan nilai dari warna referensi pada direktori */assets/TestColorBar/*.

```
Borax,Borax1,./assets/TestColorBar/Borax/0.jpg,0 mg/l
```

Gambar 4.57. Format Penulisan untuk Warna Referensi

Format untuk setiap baris dari *config_path.txt* adalah [nama jenis uji],[kode jenis uji],[path ke *file* gambar],[nilai]. Setiap baris merepresentasikan satu batang warna dan nilai untuk satu jenis uji. Untuk [nama jenis uji] diisikan nama dari jenis uji. [kode jenis uji] merepresentasikan nama jenis uji ditambah urutan dari gambar berdasarkan nilai paling kecil ke besar. [path ke *file* gambar] adalah teks path dari aplikasi ke gambar batang warna referensi tersebut. [nilai] adalah nilai dari satu batang warna dalam mg/l. Contoh untuk satu batang warna pada *Borax* pada Gambar 4.56 ditulis sesuai dengan Gambar 4.57 di *file* config_path.txt. Gambar yang disimpan adalah gambar dari kertas warna referensi yang diambil dengan sistem.

```
br = new BufferedReader(new FileReader("./config_path.txt"));
while ((line = br.readLine()) != null) {
        String[] properties = line.split(splitBy)
        if(!properties[0].equals(testName)) {
                testName = properties[0];
                foodTestNames.add(properties[0]);
        }
        this.foodTestCode.add(properties[1]);
        ArrayList<String> property = new ArrayList<String>();
        property.add(properties[1]);
        property.add(properties[2]);
        this.colorBarPath.add(property);

        property = new ArrayList<String>();
        property.add(properties[1]);
        property.add(properties[3]);

        this.colorBarValue.add(property);
```

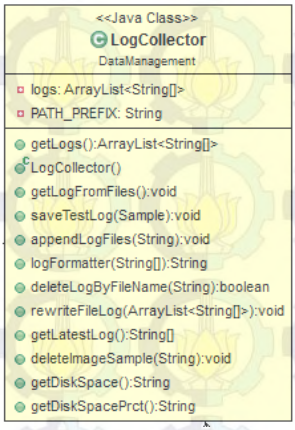
Kode Sumber 4.4. Fungsi getAllAssets()

Kode Sumber 4.4 menunjukkan potongan fungsi getAllAssets() dari kelas ini. Ketika diinstansiasi konstruktor dari kelas Assets akan memuat *file* config_path.txt dan menyimpan tiap baris teks sebagai *String*. Informasi dari teks ini akan dipisah dengan tanda koma dan disimpan pada sebuah *Array String*. Informasi dari *Array String* ini akan dipisah lagi dan disimpan ke

dalam *ArrayList of String* untuk nama jenis uji, path ke gambar dan nilai dari warna referensi.

4.2.3.2 Kelas *Log*Collector

Kelas ini digunakan untk mengambil *log* yang disimpan, menghapus *log* dan menambahkan *log* dalam *file* testLogs.txt. setiap *log* yang disimpan akan ditambahkan ke dalam *file* testLogs.txt dengan format [jenis uji],[tanggal dan waktu sistem],[nilai hasil uji]. [jenis uji] adalah nama jenis uji yang disimpan. [tanggal dan waktu sistem] adalah tanggal dan waktu sistem ketika hasil uji disimpan. [nilai hasil uji] adalah nilai dari hasil uji yang telah dilakukan.



Gambar 4.58. Implementasi Kelas LogCollector

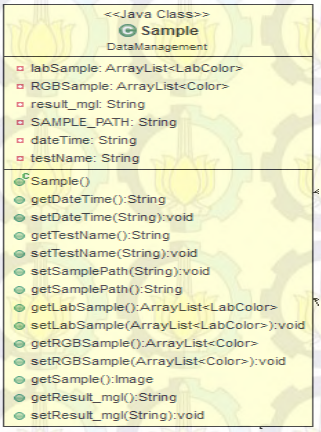
```
Formalin,2015-05-24T13:18:10.121,4 mg/l
```

Gambar 4.59. Contoh Penulisan *Log* pada Sistem

Contoh untuk satu baris penyimpanan hasil uji formalin dengan hasil 4 mg/l sesuai format sesuai pada Gambar 4.59. Implementasi dari kelas ini ditunjukkan pada Gambar 4.58. Konstruktor dari kelas ini akan memanggil fungsi

getLogFromFiles() dan menyimpan semua baris *log* dari *file* testLogs.txt ke dalam *Array of String*. Fungsi savetTestLog() dengan parameter kelas Sampel akan menambahkan *log* dari sampel hasil uji yang diberikan. Fungsi appendLogFile() menambahkan baris baru berupa *String log* ke dalam *file* test*Log*s.txt. fungsi deleteLogByFileName() dengan parameter nama *file* digunakan untuk menghapus baris *log* dengan nama *file* yang dimasukan pada *file* testLogs.txt. fungsi getLatestLog() mengembalikan *Array String*, digunakan untuk mengambil *log* yang paling terakhir disimpan. Fungsi rewriteFileLog() digunakan untuk memuat kembali teks dalam *file* testLogs.txt dengan sejumlah daftar *log* baru jika terjadi perubahan pada *log*.

4.2.3.3 Kelas Sample



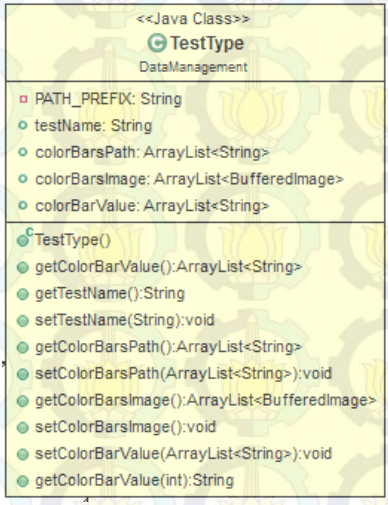
Gambar 4.60. Implementasi Kelas Sample

Kelas ini digunakan untuk merepresentasikan sampel yang akan diuji ke dalam sistem sebagai sebuah kelas. Properti dari kelas ini berupa nama jenis uji, tanggal dan waktu, hasil uji dalam mg/l dan path ke gambar sampel yang diambil. Setiap sampel yang diuji akan diambil gamabrnya dan disimpan dalam direktori

/TestSample/. Implementasi dari kelas ini ditunjukkan pada Gambar 4.60.

4.2.3.4 Kelas TestType

Kelas ini digunakan untuk merepresentasikan setiap jenis uji yang dipilih untuk dimuat ke dalam sistem. Ketika pengguna memulai uji, jenis uji yang dipilih akan dimuat ke dalam kelas ini TestType dan direpresentasikan sebagai sebuah kelas. Properti kelas ini terdiri dari nama jenis uji, daftar warna referensi dalam bentuk kelas BufferedImage dan daftar nilai untuk setiap warna referensi. Implementasi dari kelas ini ditunjukkan Gambar 4.61.

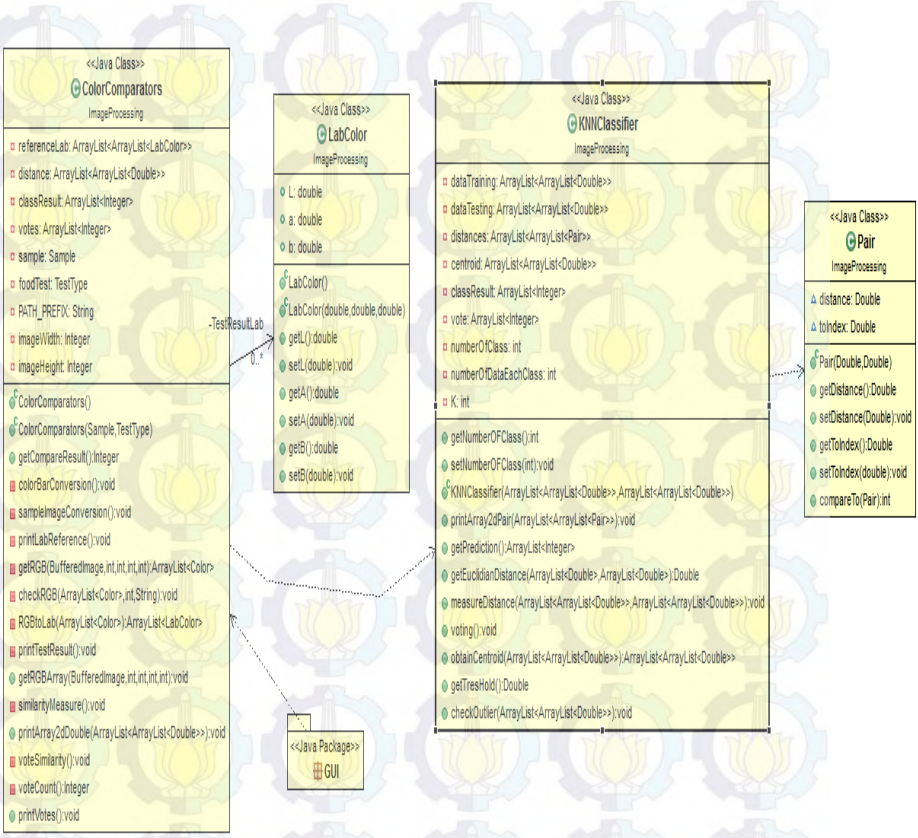


Gambar 4.61. Implementasi Kelas TestType

4.2.4 Implementasi Modul Pemrosesan Gambar

Modul ini berisi kelas-kelas yang mengatur pemrosesan dari gambar sampel untuk dibandingkan dengan gambar dari warna referensi yang diambil. Kelas dari modul ini meliputi kelas ColorComparators, KNNClassifier, LabColor dan Pair.

Implementasi dan hubundan antar kelas ini ditunjukkan pada Gambar 4.62.

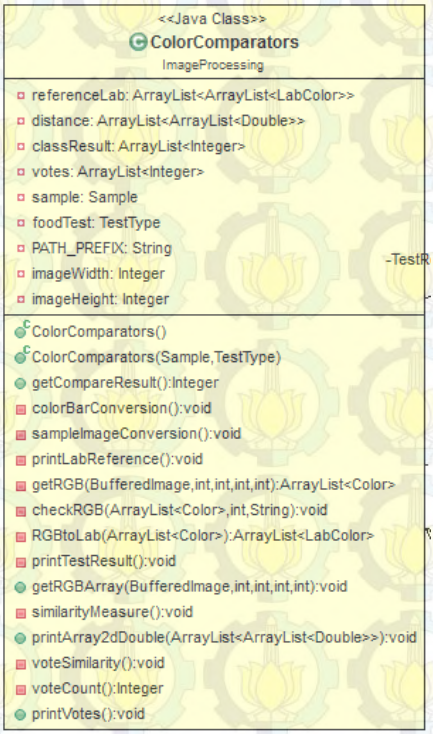


Gambar 4.62. Implementasi Modul Pemrosesan Gambar

4.2.4.1 Kelas ColorComparators

Kelas ini digunakan untuk mengekstrak setiap warna piksel dari gambar sampel dan gambar warna referensi yang disimpan ,merubah daftar warna piksel tersebut ke dalam ruang warna Lab dan mengambil hasil dari perbandingan dengan nilai

modus untuk setiap hasil klasifikasi dari setiap *pixel* sampel dengan piksel warna referensi. Implementasi kelas ini ditunjukkan pada Gambar 4.63.



Gambar 4.63. Implementasi Kelas ColorComparators

Konstruktor dari kelas ini akan menyimpan kelas *Sample* dan *TestType* yang akan dibandingkan. Fungsi *getCompareResult()* akan mengembalikan nilai perbandingan berupa *Integer* urutan dari nilai warna referensi.

```
private void colorBarConversion() {
ArrayList <BufferedImage> colorBars = new
ArrayList<BufferedImage>();
colorBars = foodTest.getColorBarsImage();
ArrayList<Color>colors = new ArrayList<Color>();
ArrayList<BufferedImage> referenceSamples = new
ArrayList<BufferedImage>();
int count = 1;
int classT = 1;
for (BufferedImage image : colorBars) {
int width  = image.getWidth();
        int height = image.getHeight();
        int x = (width/2);
        int y = (height/2);
        BufferedImage croppedImage = image.getSubimage(x, y,
this.imageWidth, this.imageHeight);
        colors = getRGB(croppedImage, x, y, this.imageWidth,
this.imageHeight);
        referenceLab.add(RGBtoLab(colors));
        this.checkRGB(colors, classT,"./RGBTrain.txt");
                classT++;
            }
   }
```

Kode Sumber 4.5. Fungsi colorBarConversion

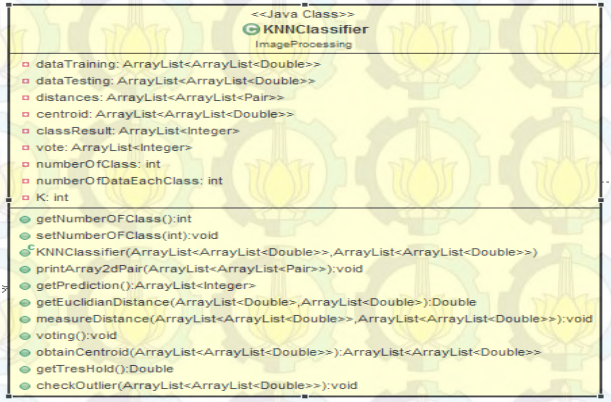
Kode Sumber 4.5 menunjukkan potongan kode sumber dari fungsi colorBarConversion. Fungsi colorBarConversion() akan mengekstrak nilai warna RGB(*Red Green Blue*) pada daftar warna referensi pada kelas TestType lalu merubahnya menjadi ruang warna Lab. Fungsi sampleImageConversion() akan mengekstrak nilai warna RGB pada gambar sampel yang di potong segi empat berukuran 10x30 piksel lalu merubahnya menjadi ruang warna Lab. Metode yang sama juga digunakan untuk mengekstrak warna dari sampel.

Fungsi similarityMeasure() merubah daftar nlai warna Lab dari piksel gambar sampel dan gambar warna referensi menjadi nilai *Double* data uji dan data latih. Nilai dari data uji dan data latih ini akan diklasifikasi dengan algoritma KNN(*K Nearest Neighbor*) menggunakan kelas KNNClassifier. Hasil dari klasifikasi kelas KNNClassifier berupa array kelas prediksi nilai jenis uji makanan dari setiap data uji. Fungsi voteSimilarity() digunakan untuk

menghitung jumlah kelas dari hasil kelas prediksi nilai jenis uji untuk setiap data uji. Setiap kelas merepresentasikan urutan dari nilai warna referensi. Fungsi voteCount() digunakan untuk mendapatkan modus dari penghitungan kelas pada setiap kelas warna referensi. Kelas modus dari kelas warna referensi adalah indeks dari urutan warna referensi yang menunjuk ke hasil uji yang dikeluarkan.

4.2.4.2 Kelas KNNClassifier

Kelas ini digunakan untuk melakukan klasifikasi menggunakan algoritma KNN terhadap data uji dari sampel dan data latih dari daftar warna referensi. Implementasi dari kelas ini ditunjukkan pada Gambar 4.64.



Gambar 4.64. Implementasi Kelas KNNClassifier

Konstruktor dari kelas ini akan menyimpan daftar data uji dan data latih yang diberkan. Fungsi gePrediction() mengembalikan *ArrayList* of *Integer* bernilai kelas dari hasil prediksi dengan KNN. Fungsi obainCentroid() dengan parameter seluruh data latih akan mengembalikan nilai *centroid* dari setiap data dalam satu kelas warna referensi yang sama. Data *centroid* adalah nilai rata-rata dari setiap fitur data.

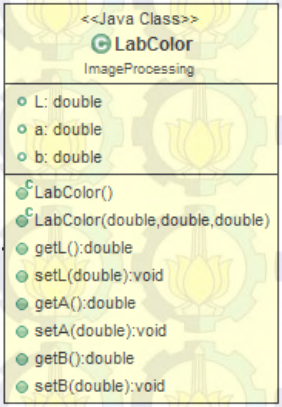
```
public void checkOutlier(ArrayList<ArrayList<Double>> dataTest)
{
        boolean isOutlier = true;
        double treshold = this.getTresHold();
        for (ArrayList<Double> list : dataTest) {
                for (ArrayList<Double> listCentroid :
this.centroid) {
                    double distance =
this.getEuclidianDistance(list, listCentroid);
                        if(distance < (double) treshold) {
                                        isOutlier = false;
                                        break;
                                }
                        }
                        if(isOutlier == true) {
                                list.set(list.size()-1,1.0);
                        }
                        else {
                                list.set(list.size()-1,0.0);
                        }
                }
        }
```

Kode Sumber 4.6. Fungsi checkOutlier

Kode Sumber 4.6 menunjukkan kode sumber dari fungsi checkOutlier(). Fungsi checkOutlier() akan membandingkan jarak setiap data uji dengan nilai *centroid* setiap kelas. Jika nilai jarak terlalu jauh dari batas yang telah ditentukan makan satu data uji tersebut adalah *outlier* dan masuk ke kelas 0 yang nilai warna referensinya setara dengan 0 mg/l. Fungsi measureDistance() dengan parameter data latih dan data uji akan menghitung jarak dari setiap data dari data latih terhadap data uji. Jarak dihitung dengan rumus Euclidian *distance*. Fungsi voting() akan mengambil sejumlah K data terdekat dari setiap data latih yang telah dihitung jaraknya dengan data uji, lalu mencari modus kelas dari sejumlah K data tersebut. Modus kelas ini adalah kelas prediksi dari sebuah data uji. Setiap data uji akan memiliki kelas prediksi yang dikembalikan dalam bentuk *ArrayList of Integer*.

4.2.4.3 Kelas LabColor

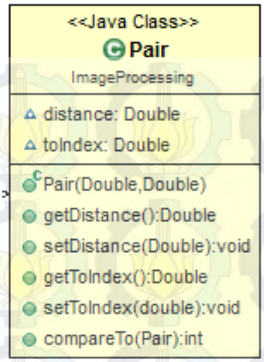
Kelas ini digunakan untuk merepresentasikan ruang warna Lab menjadi sebuah kelas. Hal ini dilakukan untuk mempermuda implementasi penyimpanan nilai dari warna piksel setiap gambar ke dalam nilai Lab. Properti dari kelas ini berupa nilai *double* L, a, dan b. Implementasi dari kelas ini ditunjukkan Gambar 4.65.



Gambar 4.65. Implementasi Kelas LabColor

4.2.4.4 Kelas Pair

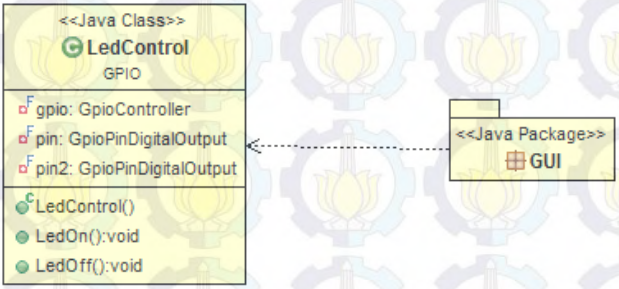
Kelas ini digunakan untuk menyimpan 2 buah nilai yang berpasangan menjadi sebuah objek. Kelas ini digunakan untuk menyimpan nilai jarak dan indeks dari data uji ke dalam sebuah array. Implementasi kelas ini ditunjukkan pada Gambar 4.66.



Gambar 4.66 Implementasi Kelas Pair

4.2.5 Implementasi Modul GPIO

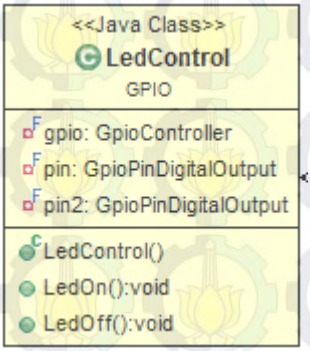
Modul ini berisi kelas untuk mengatur penggunaan pin dan GPIO pada Raspberry Pi. Implementasi kelas ini menggunakan library Pi4J untuk menginisiasi pin dan GPIO yang akan digunakan. Library ini mendukung penggunaan java dalam pengembangan sistem Raspberry Pi. Implementasi dari modul ini ditunjukkan Gambar 4.67.



Gambar 4.67. Implementasi Modul GPIO

4.2.5.1 Kelas *LedControl*

Kelas ini digunakan untuk mengontrol GPIO yang terhubung dengan lampu LED. Nomor pin dari GPIO yang digunakan untuk lampu adalah pin nomor 17 dan 18 pada P5 Header. Konstruktor dari kelas ini akan menginstansiasi pin yang akan digunakan.



Gambar 4.68. Implementasi Kelas LedControl

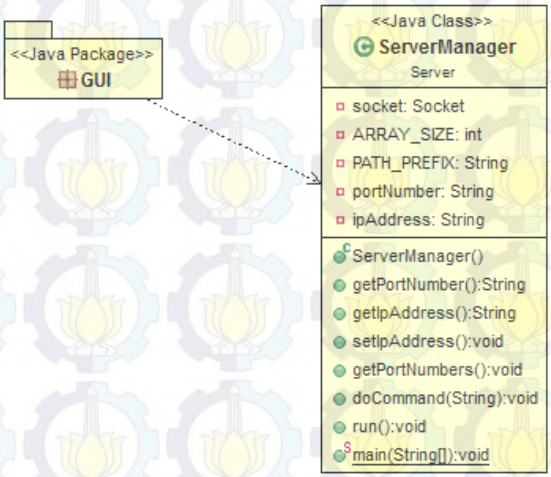
```
public LedControl() {
    gpio = GpioFactory.getInstance();
    pin = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_18,
"PinLED", PinState.LOW);
    pin2 =gpio.provisionDigitalOutputPin(RaspiPin.GPIO_17,
"PinLED", PinState.LOW);
     }
     public void LedOn() {
          pin.high();
          pin2.high();
     }
     public void LedOff() {
          pin.low();
          pin2.low();
          gpio.shutdown();
          gpio.unprovisionPin(pin);
          gpio.unprovisionPin(pin2);
     }
```

Kode Sumber 4.7. Kelas LedControl

Kode Sumber 4.7. Kelas LedControl menunjukkan potongan kode sumber dari kelas LedControl. Konstruktor LedControl() menginisiasi pin yang akan digunakan. Fungsi LedOn() akan membuat status kedua pin memiliki status *High* sehingga listrik mengalir dan menyalakan lampu. Fungsi LedOff() akan membuat status kedua pin memiliki status *Low* sehingga lampu mati. Implementasi dari kelas ini ditunjukkan Gambar 4.68.

4.3 Implementasi *Socket Server*

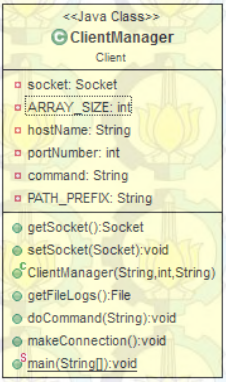
Socket Server pada sistem ini digunakan untuk mendukung transfer *file* testLogs.txt ke aplikasi *Client*. Manajemen *log* dapat dilakukan pada aplikasi web yang dibuka pada satu jaringan yang sama. *Socket Server* ini diinstansiasi pada kelas Main sebagai sebuah *thread* yang menunggu perintah dari aplikasi *Client*. Implementasi dari kelas ServerManager ditunjukkan pada Gambar 4.69.



Gambar 4.69. Implementasi Kelas ServerManager

```
Enumeration<NetworkInterface> n =
NetworkInterface.getNetworkInterfaces();
        for (; n.hasMoreElements();) {
          NetworkInterface e = n.nextElement();
          Enumeration<InetAddress> a = e.getInetAddresses();
          for (; a.hasMoreElements();){
                InetAddress addr = a.nextElement();
                if(!addr.toString().equals("/127.0.0.1") &&
addr.toString().length()<=16) {
        this.ipAddress =addr.toString();
                                        }
                                    }
                                }
```

Kode Sumber 4.8. Fungsi setIpAddress()

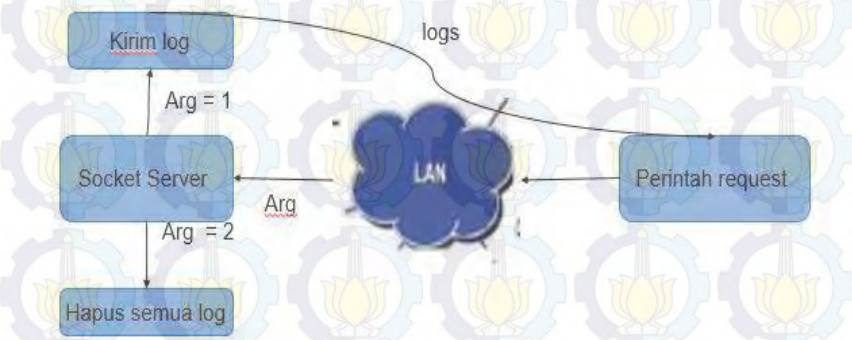


Gambar 4.70. Implementasi Kelas ClientManager

Instansiasi dari konstruktor kelas ini akan menyimpan nilai alamat IP yang digunakan oleh Raspberry Pi. Kode Sumber 4.8. Fungsi setIpAddress()merupakan potongan kode sumber dari fungsi setIpAddress() yang menjelaskan cara mendapatkan IP dari sistem. untuk setiap IP yang terdapat pada *Network Interface* sistem akan dicari IP yang bukan *localhost*. Nomor *port* yang digunakan 12345 secara statis. Alamat IP dan nomor *port* ini akan ditunjukkan pada antarmuka *FoodTestUI*.

Aplikasi dari *Client* akan menjalankan program java ClientManager yang membawa argumen berupa perintah untuk

menghapus atau mengirim data dari *file* testLog.txt. Implementasi dari kelas ClientManager ditunjukkan Gambar 4.70.



Gambar 4.71. Alur Proses pertukaran *Log*

Program ClientManager akan dijalankan melalui perintah *shell_exec* pada halaman php aplikasi *Client*. Perintah menjalankan program yaitu java ClientManager [IP] [port] [argumen]. [IP] adalah alamat IP dari sistem yang terhubung pada satu jaringan. [*port*] adalah nomor *port* yang digunakan aplikasi *Server*Manager. [argumen] adalah nilai yang menjadi perintah jika 1 makan data dari testLogs.txt akan dikirim dan jika 2 data dari testLogs.txt akan dihapus. Alur dari proses ini digambarkan pada Gambar 4.71.

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas tentang pengujian dan evaluasi dari perangkat lunak pada Tugas Akhir ini. Pengujian dilakukan pada fungsionalitas, akurasi, skalabilitas dan skenario. Pembahasan tentang lingkungan pengujian, dasar pengujian, pengujian fungsionalitas, pengujian akurasi, pengujian skalabilitas, pengujian dengan skenario, pengujian oleh pengguna dan evaluasi pengujian.

5.1 Lingkungan Pelaksanaan Pengujian

Lingkungan pengujian merupakan perangkat-perangkat yang dilibatkan dalam proses pengujian. Lingkungan pengujian ini menggunakan perangkat keras berupa komputer kecil Raspberry Pi. Spesifikasi lingkungan pengujian dijelaskan pada Tabel 5.1.

Tabel 5.1 Spesifikasi Lingkungan Pengujian Perangkat Lunak

Raspberry Pi Model B Rev 2	Processor	700 MHz single-core ARM1176JZF-S
	RAM	512 MB
	Sistem Operasi	*Raspbian* , versi kernel 3.18

5.2 Dasar Pengujian

Pengujian yang dilakukan berupa pengujian fungsionalitas , akurasi, skalabilitas dan skenario sistem. Pengujian fungsionalitas dilakukan dengan model *black box* untuk masing-masing fungsionalitas dari aplikasi ini. Pengujian ini dilakukan untuk menguji apakah fungsionalitas yang diidentifikasi pada tahap kebutuhan benar-benar diimplementasikan dan bekerja seperti yang diharapkan. Pengujian akurasi dilakukan dengan melakukan uji makanan untuk setiap jenis uji dengan sampel yang telah diketahui nilai kandungannya. Pengujian skalabilitas dilakukan dengan

menambahkan jenis uji baru ke dalam sistem dan pengujian kegunaan dengan skenario dimana menggunakan salah satu reagen dalam Food Security Kit.

5.3 Pengujian Fungsionalitas

Subbab ini menjelaskan tentang skenario pengujian fungsionalitas sistem pada Tugas Akhir ini. Pengujian dilakukan berdasarkan setiap spesifikasi kebutuhan. Pengujian didokumentasikan secara sistematis sebagai tolok ukur keberhasilan sistem.

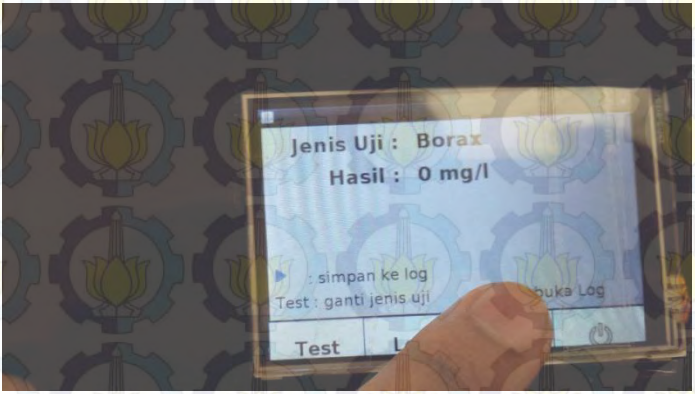
5.3.1 Pengujian Mengukur Kadar Senyawa Pada Sampel

Pengujian dilakukan dengan skenario pengguna melakukan uji makanan yang telah dipilih. Skenario terbagi sesuai alur yang terjadi diantaranya proses mengukur kadar senyawa pada sampel, memilih jenis uji sampel dan menyimpan hasil uji sampel. Skenario pengujian mengukur senyawa pada sampel dijelaskan pada Tabel 5.2. Skenario pengujian memilih jenis uji sampel dijelaskan pada Tabel 5.3. Skenario pengujian menyimpan hasil uji sampel dijelaskan pada Tabel 5.4.

Gambar 5.1. Pengujian Mengukur Kadar Senyawa Sampel

Tabel 5.2. Skenario Pengujian Mengukur Kadar Senyawa Pada Sampel

Nomor	P-01
Nama	Mengukur Kadar Senyawa Pada Sampel
Use Case	UC-01
Tujuan	Mendapatkan nilai kadar dari zat kimia jenis uji pada sampel
Kondisi awal	Sampel dalam tabung reaksi dari uji makanan yang dipilih
Skenario	1. Pengguna memasukan sampel ke dalam wadah sistem. 2. Pengguna menekan tombol mulai untuk melakukan pengujian.
Masukan	Sampel
Keluaran yang diharapkan	Nilai dari kadar zat kimia jenis uji dalam sampel
Hasil pengujian	Berhasil



Gambar 5.2. Pengujian Menampilkan Hasil Uji

Dalam pengujian mengukur kada senyawa dalam sampel, sampel dimasukan ke dalam container untuk diuji. Lalu pengguna menekan tombol mulai uji untuk melakukan pengujian sesuai dengan Gambar 5.1. Antarmuka pengujian akan menampilkan potifikasi dari proses yang berlangsung. Jenis zat kimia yang diuji berdasarkan zat kimia yang tampil pada antarmuka pengujian sebelum pengguna menekan tombol mulai. Hasil dari pengujian akan ditampilkan ke antarmuka pengujian seperti pada Gambar 5.2.

Tabel 5.3. Skenario Pengujian Memilih Jenis Uji

Nomor	P-02
Nama	Memilih jenis uji
Use Case	UC-01
Tujuan	Mengganti jenis uji yang tampil
Kondisi awal	Jenis uji pertama yang tampil
Skenario	1. Pengguna menekan tombol ganti jenis uji.
Masukan	-
Keluaran yang diharapkan	Jenis uji berikutnya yang tampil
Hasil pengujian	Berhasil

Gambar 5.3. Pengujian Memilih Jenis Uji

Pengujian ini dilakukan berdasarkan alur alternatiif dari kasus penggunaan UC-01 dimana jika pengguna ingin mengganti jenis uji yang dipilih sebelum memulai pengujian. Antarmuka pengujian akan menampilkan jenis uji pertama. Pengguna lalu menekan tombol ganti jenis uji untuk menganti jenis uji yang berikutnya. Sesuai pada Gambar 5.3.

Tabel 5.4. Skenario Pengujian Menyimpan Hasil Uji

Nomor	P-03
Nama	Menyimpan Hasil Uji
Use Case	UC-01
Tujuan	Menyimpan hasil uji sampel ke dalam *log*
Kondisi awal	Hasil uji yang ditampilkan
Skenario	1. Pengguna menekan tombol simpan pengujian.
Masukan	Hasil uji sampel
Keluaran yang diharapkan	*Log* dari hasil uji sampel tersimpan
Hasil pengujian	Berhasil

Gambar 5.4. Pengujian Menyimpan Hasil Uji

Pengujian ini dilakukan berdasarkan alur alternatiif dari kasus penggunaan UC-01 dimana jika pengguna ingin hasil dari uji sampel yang dilakukan. Antarmuka hasil uji akan menampilkan hasil dari uji sampel. Pengguna lalu menekan tombol simpan hasil uji. Sehingga antarmuka berpindah ke antarmuka pengujian dan muncul notifikasi *log* tersimpan sesuai pada Gambar 5.4.

5.3.2 Pengujian Menampilkan Log

Pengujian ilakukan dengan skenario pengguna memilih untuk membuka antarmuka manajemen *log*. Skenario terbagi sesuai alur yang terjadi diantaranya proses menampilkan *log*, menampilkan *log* berikutnya dan menghapus *log* yang tampil. Skenario pengujian menampilkan *log* dijelaskan pada Tabel 5.5. Skenario pengujian menampilkan *log* berikutnya dijelaskan pada Tabel 5.6. Skenario pengujian menghapus *log* dijelaskan pada Tabel 5.7.

Tabel 5.5. Skenario Pengujian Menampilkan Log

Nomor	P-04
Nama	Menampilkan *Log*
Use Case	UC-02
Tujuan	Menampilkan *log* yang tersimpan
Kondisi awal	Antarmuka tampilan pengujian
Skenario	1. Pengguna menekan tombol Log untuk membuka antarmuka manajemen *log*
Masukan	-
Keluaran yang diharapkan	*Log* yang terakhir kali disimpan tampil
Hasil pengujian	Berhasil

Gambar 5.5. Pengujian Menampilkan Log

Pengguna menekan tombol Log untuk melihat *log* yang disimpan. Antarmuka *log* akan menampilkan *log* terakhir yang disimpan. Log uji sampel terakhir akan ditampilkan ke antarmuka Log Manager seperti pada Gambar 5.5.

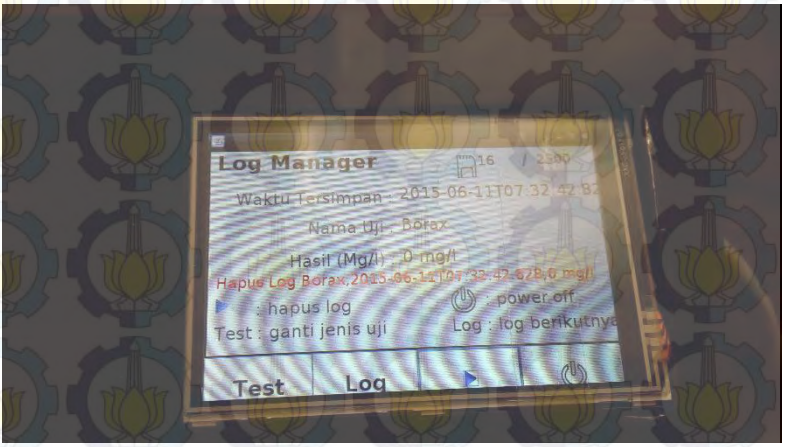
Tabel 5.6. Skenario Pengujian Menampilkan Log Berikutnya

Nomor	P-05
Nama	Menampilkan *Log* berikutnya
Use Case	UC-01
Tujuan	Menampilkan *log* yang disimpan sebelum *log* yang tampil
Kondisi awal	*Log* tampil di antarmuka manajemen *log*
Skenario	1. Pengguna menekan tombol Log berikutnya
Masukan	-
Keluaran yang diharapkan	*Log* yang tampil adalah *log* yang disimpan sebelumnya
Hasil pengujian	Berhasil

Pengujian ini dilakukan berdasarkan salah satu alur alternatiif dari kasus penggunaan UC-02 dimana jika melihat *log* lain yang telah disimpan sebelumnya. Antarmuka manajemen *log* akan menampilkan *log* yang paling terakhir disimpan. Lalu pengguna menekan tombol ganti *log* untuk mengganti *log* yang ditampilkan.

Tabel 5.7. Skenario Pengujian Menghapus Log Tampil

Nomor	P-06
Nama	Menghapus *Log* Tampil
Use Case	UC-02
Tujuan	Menghapus *log* yang sedang ditampilkan di antarmuka manajemen *log*
Kondisi awal	Antarmuka manajemen *log* menampilkan *log*
Skenario	1. Pengguna menekan tombol hapus *log*.
Masukan	-
Keluaran yang diharapkan	*Log* terhapus dan sistem memberikan respon berupa notifikasi
Hasil pengujian	Berhasil



Gambar 5.6. Pengujian Menghapus *Log* Tampil

Pengujian ini dilakukan berdasarkan alur alternatiif dari kasus penggunaan UC-02 dimana jika pengguna ingin menhapus *log* yang ditampilkan. Antarmuka manajemen *log* akan menampilkan *log*. Pengguna lalu menekan tombol hapus *log*. Sehingga *log* yang tampil terhapus, sistem memberikan notifikasi dan menampilkan *log* lain yang ada sesuai pada Gambar 5.6.

5.3.3 Pengujian Mengirim File Log Ke Aplikasi Web Client

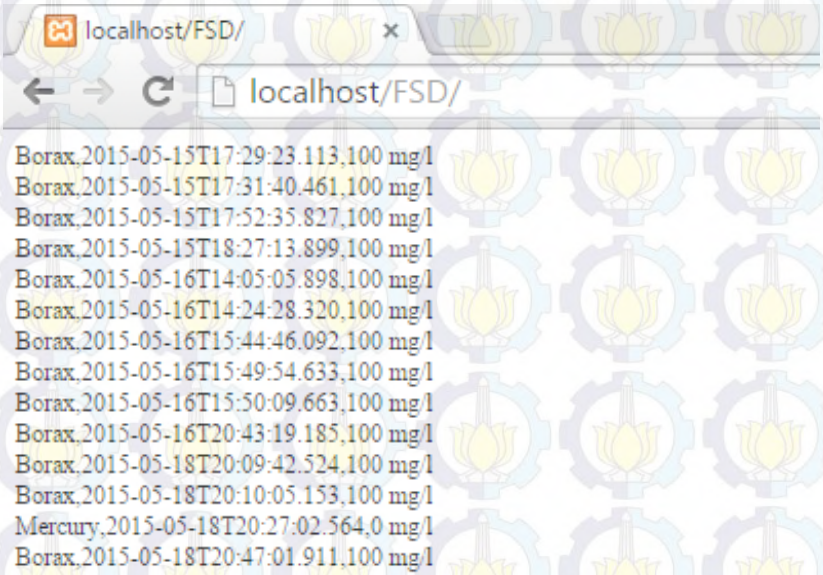
Pengujian ini dilakukan dengan skenario terdapat aplikasi web sederhana dengan bahasa php yang akan menjalankan program java ClientManager yang telah tertanam didalamnya. Aplikasi ini akan mengirimkan pesan permintaan sesuai dengan perintah yang diberikan yaitu menghapus semua isi *log* atau mengirim *log* ke aplikasi web. Terdapat juga kelas ServerManager yang merupakan *Socket Server* yang berjalan dibelaang sistem menunggu perintah berupa argumen. Skenario pengujian mengirim *log* dijelaskan pada Tabel 5.8. Skenario Pengujian Mengirim Log ke Aplikasi . Skenario pengujian menghapus semua *log* sistem dijelaskan pada Tabel 5.9. Skenario Pengujian Menghapus Semua Log Sistem.

Tabel 5.8. Skenario Pengujian Mengirim Log ke Aplikasi Manajemen Log

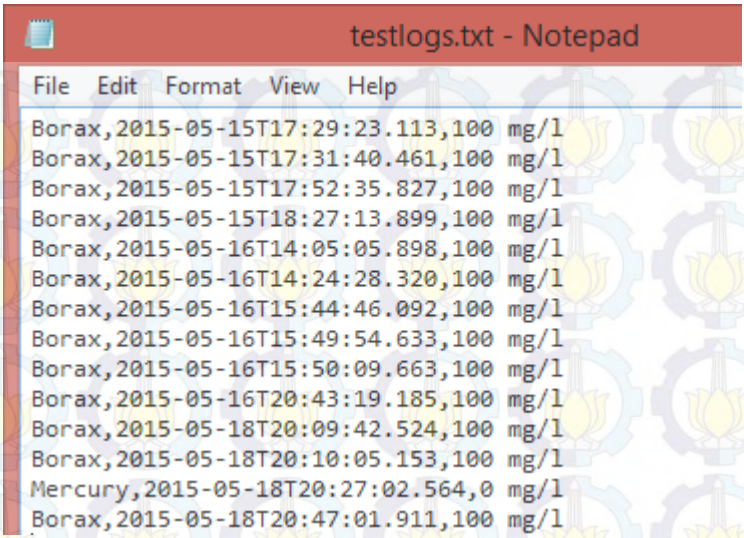
Nomor	P-07
Nama	Mengirim *Log* ke Aplikasi Manajemen Log
Use Case	-
Tujuan	Mengirimkan *file log* ke aplikasi Manajemen Log
Kondisi awal	*File log* tersimpan di sistem
Skenario	1. Administrator mengatur argument bernilai 1 untuk menjalankan aplikasi *Client*Manager 2. Administrator membuka halaman php yang menjalankan aplikasi *Client* Manager

Masukan	Argumen dari aplikasi *Client*Manager
Keluaran yang diharapkan	*File log* dari sistem yang tampil di halaman dan tersimpan di aplikasi web
Hasil pengujian	Berhasil

Pengujian ini dilakukan berdasarkan salah satu alur kebutuhan untuk memfasilitasi permintaan dari aplikasi web *Client*. Administrator dari aplikasi web mengatur argumen dari aplikasi *Client*Manager bernilai 1.



Gambar 5.7. Pengujian mengirim log



Gambar 5.8. File log tersimpan

Administrator membuka halaman aplikasi web yang menjalankan program *Client*Manager sesuai dengan Gambar 5.7. Halaman web menampilkan daftar *log* dan *file log* tersimpan dalam direktori di web sesuai dengan Gambar 5.8.

Tabel 5.9. Skenario Pengujian Menghapus Semua Log Sistem

Nomor	P-08
Nama	Mengirim *Log* ke Aplikasi Manajemen Log
Use Case	-
Tujuan	Menghapus isi dari *file log* sistem
Kondisi awal	*File log*
Skenario	1. Administrator mengatur argument bernilai 2 untuk menjalankan aplikasi *Client*Manager 2. Administrator membuka halaman php yang menjalankan aplikasi *Client* Manager

Masukan	Argumen dari aplikasi *Client*Manager
Keluaran yang diharapkan	Isi dari *File log* sistem akan dihapus
Hasil pengujian	Berhasil

Pengujian ini dilakukan berdasarkan salah satu alur kebutuhan untuk memfasilitasi permintaan dari aplikasi *Client*. Administrator dari aplikasi web mengatur argumen dari aplikasi ClientManager bernilai 2. Argumen ini adalah untuk perintah untuk kelas ServerManager menghapus semua *log* dalam sistem. Administrator membuka halaman aplikasi web yang menjalankan program ClientManager sesuai dengan Gambar 5.9.

Gambar 5.9. Pengujian menghapus semua log sistem

5.4 Pengujian Akurasi

Pengujian dilakukan untuk mengetahui keakuratan dari sistem dalam menentukan kadar senyawa dari sampel. Skenario uji coba dilakukan dengan melakukan percobaan terhadap jenis uji yang disimpan yaitu Borax, *Rhodamin B*, Formalin, *Methanyl Yellow* dan Merkuri. Percobaan dilakukan dengan sampel yang telah diketahui

kadar kandungannya sebanyak 5 kali percobaan untuk setiap sampel. Lalu diambil rata-rata akurasi dari setiap kelas dalam satu jenis uji.

Tabel 5.10. Hasil Uji Boraks untuk 5 Kali Uji Coba Pada Setiap Kelas

Boraks					
kelas (mg/l)	prediksi tiap percobaan (mg/l)				
sampel	1	2	3	4	5
0	0	0	0	0	0
50	50	50	50	50	50
100	100	100	100	100	100
150	150	150	150	150	150
200	200	200	200	200	200
5000	5000	5000	5000	5000	5000
10000	10000	10000	10000	10000	10000

Pada Tabel 5.10 menunjukkan hasil pengujian untuk uji zat boraks. Terdapat 7 kelas pada uji boraks yang merupakan warna referensi kandungan pada reagen boraks. Pengujian pada semua kelas reagen boraks menghasilkan nilai kandungan yang sesuai dengan kelasnya. Dalam pengujian kali ini setiap kelas mendapatkan akurasi 100%. Rata-rata akumulasi setiap kelas untuk uji boraks pada pengujian ini menghasilkan akurasi 100%.

Tabel 5.11. Hasil Uji *Rhodamin B* untuk 5 Kali Uji Coba Pada Setiap Kelas.

Rhodamin B					
kelas (mg/l)	prediksi (mg/l)				
	1	2	3	4	5
0	0	0	0	0	0
2	2	2	2	2	2
10	10	10	10	10	10
20	20	20	20	20	20

30	30	30	30	30	30
40	30	30	40	30	30

Pada Tabel 5.11 menunjukkan hasil pengujian untuk uji zat *Rhodamin B*. Terdapat 5 kelas pada uji *Rhodamin B* yang merupakan warna referensi kandungan pada reagen *Rhodamin B*. Pengujian pada kelas 0 mg/l, 2 mg/l, 10 mg/l, 20 mg/l, 30 mg/l reagen *Rhodamin B* menghasilkan nilai kandungan yang sesuai dengan kelasnya. Namun pada kelas 40 mg/l uji *Rhodamin B* hanya sesuai pada percobaan ke 3. Dalam pengujian kali ini kelas 0 mg/l, 2 mg/l, 10 mg/l, 20 mg/l ,30 mg/l mendapatkan akurasi 100% dan kelas 40 mg/l mendapatkan akurasi 20%. Rata-rata akumulasi untuk setiap kelas dalam uji *Rhodamin B* adalah 86.667%.

Tabel 5.12. Hasil Uji Merkuri untuk 5 Kali Uji Coba Pada Setiap Kelas.

Merkuri					
kelas (mg/l) sampel	prediksi tiap percobaan (mg/l)				
	1	2	3	4	5
0	0	0	0	0	0
50	50	50	50	50	50
100	100	100	100	100	100
150	150	150	150	150	150
200	200	200	200	200	200
1000	1000	1000	200	1000	1000
2000	2000	2000	2000	3000	2000
3000	2000	3000	2000	3000	3000

Pada Tabel 5.12 menunjukkan hasil pengujian untuk uji zat Merkuri. Terdapat 8 kelas pada uji Merkuri yang merupakan warna referensi kandungan pada reagen Merkuri Pengujian pada kelas 0 mg/l, 50 mg/l, 100 mg/l, 150 mg/l dan 200 mg/l reagen Merkuri menghasilkan nilai kandungan yang sesuai dengan kelasnya. Namun pada kelas 1000 mg/l uji Merkuri hanya sesuai pada percobaan ke 1,2,4 dan 5. Pada kelas 2000 mg/l uji Merkuri hanya

sesuai pada percobaan ke 1,2,3 dan 5. Pada kelas 3000 mg/l uji Merkuri hanya sesuai pada percobaan ke 2,4 dan 5. Dalam pengujian kali ini kelas 0 mg/l, 50 mg/l, 100 mg/l, 150 mg/l dan 200 mg/l mendapatkan akurasi 100% , kelas 1000 mg/l mendapatkan akurasi 80%, kelas 2000 mg/l mendapatkan akurasi 80% dan kelas 3000 mg/l mendapatkan akurasi 60%. Rata-rata akumulasi untuk setiap kelas dalam uji Merkuri adalah 90%.

Tabel 5.13. Hasil Uji Formalin untuk 5 Kali Uji Coba Pada Setiap Kelas.

Formalin					
kelas (mg/l) sampel	prediksi tiap percobaan (mg/l)				
	1	2	3	4	5
0	0	0	0	0	0
2	2	2	2	2	2
4	4	4	4	4	4
6	6	4	6	6	6
8	10	6	6	10	6
10	10	10	10	10	10

Pada Tabel 5.13 menunjukkan hasil pengujian untuk uji zat Formalin. Terdapat 6 kelas pada uji Formalin yang merupakan warna referensi kandungan pada reagen Formalin Pengujian pada kelas 0 mg/l, 2 mg/l, 4 mg/l dan 10 mg/l reagen Formalin menghasilkan nilai kandungan yang sesuai dengan kelasnya. Namun pada kelas 6 mg/l uji Formalin hanya sesuai pada percobaan ke 1,3,4 dan 5. Pada kelas 8 mg/l uji Formalin tidak mendapatkan hasil yang sesuai. Dalam pengujian kali ini kelas 0 mg/l, 2 mg/l, 4 mg/l dan 10 mg/l mendapatkan akurasi 100% , kelas 6 mg/l mendapatkan akurasi 80% dan kelas 8 mg/l mendapatkan akurasi 0%. Rata-rata akumulasi untuk setiap kelas dalam uji Formalin adalah 80%.

Sistem tidak memprediksi hasil yang tepat untuk kelas 8 mg/l. Namun jika dilihat dari rentang selisih kelasnya, hasil prediksi sistem hanya meleset satu kelas dari kelas target. Jika prediksi tepat

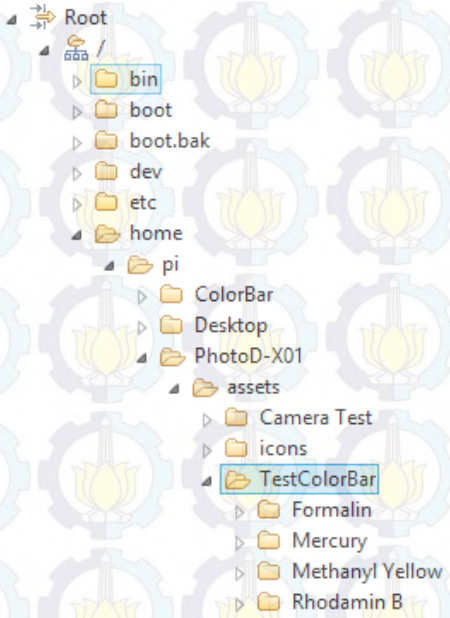
di representasikan menjadi nilai 100% dan meleset satu kelas di representasikan menjadi 1/n dimana n adalah jumlah kelas. Maka rata-rata akumulasi setiap kali percobaan untuk prediksi kelas 8 mg/l nilainya adalah 83.33% ,untuk kelas 0 mg/l, 2 mg/l, 4 mg/l dan 10 mg/l bernilai 100% dan untuk kelas 6 mg/l bernilai 96%. Sehingga didapat rata-rata akumulasi dari perobaan setiap kelas ini merepresentasikan uji Formalin bernilai 96.5% jika berdasarkan akurasi dari rentang kelasnya.

Tabel 5.14. Hasil Uji *Methanyl Yellow* untuk 5 Kali Uji Coba Pada Setiap Kelas.

Methanyl Yellow					
kelas (mg/l) sampel	prediksi tiap percobaan (mg/l)				
	1	2	3	4	5
0	0	0	0	0	0
50	50	50	50	50	50
100	100	100	100	100	100
150	150	150	150	150	150
200	200	200	200	200	200
5000	10000	10000	10000	5000	5000
10000	10000	5000	10000	10000	10000

Pada Tabel 5.14 menunjukkan hasil pengujian untuk uji zat *Methanyl Yellow*. Terdapat 7 kelas pada uji *Methanyl Yellow* yang merupakan warna referensi kandungan pada reagen *Methanyl Yellow* Pengujian pada kelas 0 mg/l, 50 mg/l, 100 mg/l, 150 mg/l dan 200 mg/l reagen *Methanyl Yellow* menghasilkan nilai kandungan yang sesuai dengan kelasnya. Namun pada kelas 5000 mg/l uji *Methanyl Yellow* hanya sesuai pada percobaan ke 4 dan 5. Pada kelas 10000 mg/l uji *Methanyl Yellow* hanya sesuai pada percobaan ke 1,3,4 dan 5. Dalam pengujian kali ini kelas 0 mg/l, 50 mg/l, 100 mg/l, 150 mg/l dan 200 mg/l mendapatkan akurasi 100% , kelas 5000 mg/l mendapatkan akurasi 40% dan kelas 10000 mg/l mendapatkan akurasi 80%. Rata-rata akumulasi untuk setiap kelas dalam uji *Methanyl Yellow* adalah 88.571%.

5.5 Pengujian Skalabilitas



Gambar 5.10. Hirarki Direktori Assets

Pengujian dilakukan untuk mendemonstrasikan skalabilitas dari sistem dalam kasus menambah jenis uji. Skenario yang dilakukan adalah dengan menghilangkan salah satu jenis uji yang ada untuk berperan sebagai jenis uji baru yang tidak terdapat dalam sistem. Jenis uji tersebut akan ditambahkan ke dalam sistem tanpa melakukan perubahan dengan kode sumber dari aplikasi sistem. Lalu uji sampel dilakukan dengan jenis uji yang baru.

config_path.txt

```
Rhodamin-B,Rhodamin-B1,./assets/TestColorBar/Rhodamin B/0.jpg,0 mg/l
Rhodamin-B,Rhodamin-B2,./assets/TestColorBar/Rhodamin B/2.jpg,2 mg/l
Rhodamin-B,Rhodamin-B3,./assets/TestColorBar/Rhodamin B/10.jpg,10 mg/l
Rhodamin-B,Rhodamin-B4,./assets/TestColorBar/Rhodamin B/20.jpg,20 mg/l
Rhodamin-B,Rhodamin-B5,./assets/TestColorBar/Rhodamin B/30.jpg,30 mg/l
Rhodamin-B,Rhodamin-B6,./assets/TestColorBar/Rhodamin B/40.jpg,40 mg/l
Mercury,Mercury1,./assets/TestColorBar/Mercury/0.jpg,0 mg/l
Mercury,Mercury2,./assets/TestColorBar/Mercury/50.jpg,50 mg/l
Mercury,Mercury3,./assets/TestColorBar/Mercury/100.jpg,100 mg/l
Mercury,Mercury4,./assets/TestColorBar/Mercury/150.jpg,150 mg/l
Mercury,Mercury5,./assets/TestColorBar/Mercury/200.jpg,200 mg/l
Mercury,Mercury6,./assets/TestColorBar/Mercury/1000.jpg,1000 mg/l
Mercury,Mercury7,./assets/TestColorBar/Mercury/2000.jpg,2000 mg/l
Mercury,Mercury8,./assets/TestColorBar/Mercury/3000.jpg,3000 mg/l
Methanyl-Yellow,Methanyl-Yellow1,./assets/TestColorBar/Methanyl Yellow/0.jpg,0 mg/l
Methanyl-Yellow,Methanyl-Yellow2,./assets/TestColorBar/Methanyl Yellow/50.jpg,50 mg/l
Methanyl-Yellow,Methanyl-Yellow3,./assets/TestColorBar/Methanyl Yellow/100.jpg,100 mg/l
Methanyl-Yellow,Methanyl-Yellow4,./assets/TestColorBar/Methanyl Yellow/150.jpg,150 mg/l
Methanyl-Yellow,Methanyl-Yellow5,./assets/TestColorBar/Methanyl Yellow/200.jpg,200 mg/l
Methanyl-Yellow,Methanyl-Yellow6,./assets/TestColorBar/Methanyl Yellow/5000.jpg,5000 mg/l
Methanyl-Yellow,Methanyl-Yellow7,./assets/TestColorBar/Methanyl Yellow/10000.jpg,10000 mg/l
Formalin,Formalin1,./assets/TestColorBar/Formalin/0.jpg,0 mg/l
Formalin,Formalin2,./assets/TestColorBar/Formalin/2.jpg,2 mg/l
Formalin,Formalin3,./assets/TestColorBar/Formalin/4.jpg,4 mg/l
Formalin,Formalin4,./assets/TestColorBar/Formalin/6.jpg,6 mg/l
Formalin,Formalin5,./assets/TestColorBar/Formalin/8.jpg,8 mg/l
Formalin,Formalin6,./assets/TestColorBar/Formalin/10.jpg,10 mg/l
```

Gambar 5.11. ***File config_path.txt***

Skenario diawali dengan memilih salah satu jenis uji yang akan dihilangkan. Misalkan jenis uji boraks belum terdapat dalam sistem. Hal ini dapat dibuktikan dengan tidak terdapatnya aset berupa daftar gambar warna referensi untuk jenis uji tersebut seperti pada Gambar 5.10 yang menampilkan hirarki direktori dengan RSE. Juga daftar uji boraks tidak ditambahkan ke dalam *file* config_path.txt yang mengatur penyimpanan aset jenis uji. Isi dari *file* config_path.txt ditunjukkan Gambar 5.11.

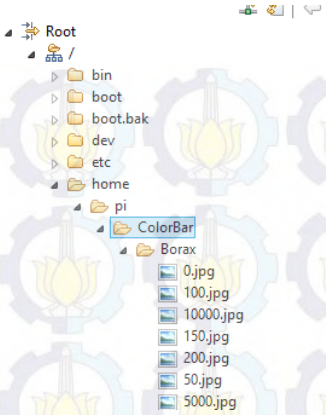
Berikutnya adalah pengambilan aset dari jenis uji untuk uji boraks. Hal ini dilakukan dengan mengambil gambar dari warna referensi satu per satu untuk setiap kelas menggunakan sistem. Terdapat cara untuk mengambil gambar ini diantaranya adalah

dengan perintah menggunakan terminal untuk mengambil gambar secara langsung. Hal ini dilakukan mengunakan protokol SSH (Secure Shell) untuk terhubung dengan sistem. Untuk mengambil gambar langsung digunakan daftar perintah pada yang merupakan daftar perintah yang digunakan untuk mengambil gambar sampel untuk jenis uji boraks.

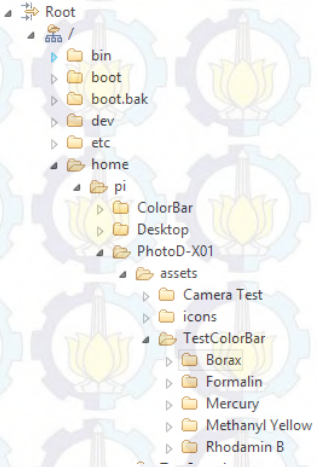
```
raspistill -vf -hf -o /home/pi/ColorBar/Borax/0.jpg -w 480
-h 360
raspistill -vf -hf -o /home/pi/ColorBar/Borax/50.jpg -w 480
-h 360
raspistill -vf -hf -o /home/pi/ColorBar/Borax/100.jpg -w 480
-h 360
raspistill -vf -hf -o /home/pi/ColorBar/Borax/150.jpg -w 480
-h 360
raspistill -vf -hf -o /home/pi/ColorBar/Borax/200.jpg -w 480
-h 360
raspistill -vf -hf -o /home/pi/ColorBar/Borax/5000.jpg -w
480 -h 360
raspistill -vf -hf -o /home/pi/ColorBar/Borax/10000.jpg -w
480 -h 360
```

Gambar 5.12. Daftar Pernitah untuk Mengambil Gambar Warna Referensi

Perintah raspistill adalah perintah untuk mengambil gambar. Opsi –vf dan –hf adalah perintah untuk memutar kamera secara vertical lalu horizontal untuk memutar gambar 360 derajat. Hal ini dilakukan karena posisi kamera yang terbalik. Opsi –o untuk mendefinisikan keluaran dari kamera. Berikutnya adalah path ke *file* yang akan disimpan. Nama *file* yang digunakan harus menggunakan salah satu dari format penyimpanan gambar yang kompatibel dengan kamera Raspberry Pi. Opsi –w 480 dan –h 360 digunakan untuk mendefinisikan resolusi dari gambar yang diambil. Semua gambar ini akan terdapat dalam direktori sesuai path yang dituliskan. Daftar gambar yang diambil dapat dilihat pada Gambar 5.13.



Gambar 5.13. Lokasi Direktori Penyimpanan Gambar



Gambar 5.14. Lokasi Direktori Assets

Lalu direktori dari gambar yang disimpan di pindahkan ke dalam folder asset dalam sistem. Gambar 5.14 menunjukkan folder boraks yang ditambahkan ke dalam folder asset. Setelah itu daftar asset yang digunakan pada uji boraks ditambahkan ke *file* config_path.txt. penambahan dilakukan dengan format yang dituliskan pada kode Gambar 5.15

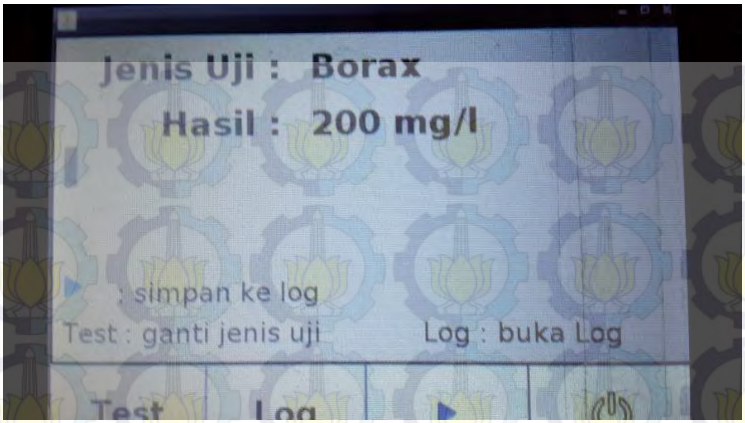
config_path.txt

```
1 Borax,Borax1,./assets/TestColorBar/Borax/0.jpg,0 mg/l
2 Borax,Borax2,./assets/TestColorBar/Borax/50.jpg,50 mg/l
3 Borax,Borax3,./assets/TestColorBar/Borax/100.jpg,100 mg/l
4 Borax,Borax4,./assets/TestColorBar/Borax/150.jpg,150 mg/l
5 Borax,Borax5,./assets/TestColorBar/Borax/200.jpg,200 mg/l
6 Borax,Borax6,./assets/TestColorBar/Borax/5000.jpg,5000 mg/l
7 Borax,Borax7,./assets/TestColorBar/Borax/10000.jpg,10000 mg/l
```

Gambar 5.15. Konfigurasi Menambahkan Aset Jenis Uji

Ketika sistem dinyalakan kembali *file config_path.txt* akan memuat semua aset baru yang ditambahkan. Uji makanan boraks terdapat pada daftar uji yang diberikan. Gambar 5.16 menunukkan jenis uji boraks yang bertambah.

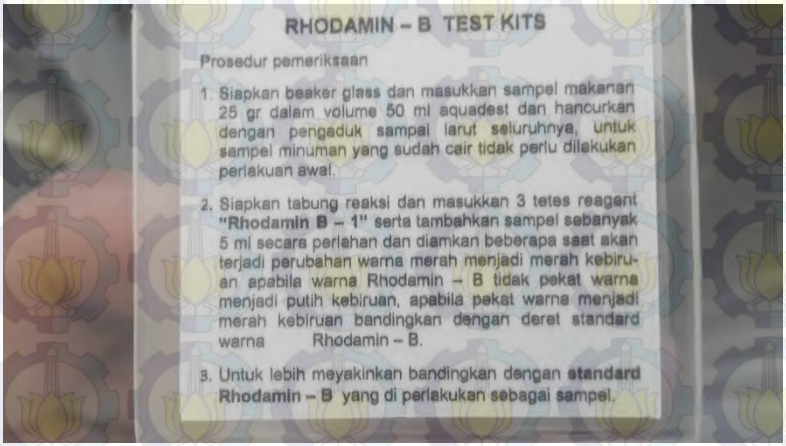
Gambar 5.16. Uji Baru Berhasil Ditambahkan



Gambar 5.17. Pengujian dengan Jenis Uji Baru

Dilakukan pengujian dengan jenis uji boraks menggunakan sampel dengan kelas 200 mg/l. Gambar 5.17 menunjukkan hasil uji dari jenis uji boraks yang baru ditambahkan.

5.6 Pengujian Kegunaan dengan Skenario



Gambar 5.18. Prosedur Uji *Rhodamin B*

Pengujian ini dilakukan untuk menunjukkan bahwa sistem mampu untuk melakukan proses uji makanan menggunakan reagen asli dari *Food Security Kit*. Uji coba ini dilakukan sesuai dengan prosedur yang terdapat pada salah satu uji. Reagen yang digunakan adalah *Rhodamin B*. Prosedur untuk uji coba ini ditunjukkan pada Gambar 5.18.

Sampel yang digunakan dalam uji coba ini adalah cairan dengan jumlah *Rhodamin B* yang tidak diketahui. Sampel dimasukan ke dalam tabung reaksi sesuai dengan prosedur dan juga beberapa tetes dari reagen *Rhodamin B*. Setelah beberapa saat reagen akan bereaksi dengan sampel dan menghasilkan sampel seperti pada Gambar 5.19.

Gambar 5.19. Sampel Mengandung *Rhodamin B*

Proses uji coba berikutnya untuk menentukan kadar reagen dalam sampel. Karena dipastikan terdapat zat *Rhodamin B* dalam sampel maka sistem seharusnya tidak menghasilkan nilai 0. Uji coba reagen *Rhodamin B* dilakukan menggunakan sistem seperti pada Gambar 5.20.

Gambar 5.20. Uji *Rhodamin B*

Setelah proses selesai hasil uji ditampilkan oleh sistem seperti pada Gambar 5.21. Hasil prediksi sistem menunjukkan jumlah senyawa Rhodamin B yang terdapat dalam sampel sebanyak 30 mg/l dan tidak menunjuk ke kelas 0 mg/l. Hal ini menunjukkan sistem dapat mendeteksi perubahan warna yang terjadi jika dilakukan uji coba dengan reagen sebenarnya.

Gambar 5.21. Hasil Uji *Rhodamin B*

5.7 Pengujian oleh Pengguna

Pengujian ini dilakukan dengan skenario yang dijalankan oleh pengguna dari sistem yang akan dibuat. Pengguna adalah beberapa orang yang ahli dalam bidang kimia dan farmasi terutama uji sampel dengan reagen. adalah daftar pengguna yang telah mencoba sistem yang dibuat.

Tabel 5.15. Daftar Profil Pengguna

No	Nama	Profesi	Instansi	Email
1	Dr. Titik Taufikurohmah, S.Si., M.Si.	Dosen & Praktisi	Jurusan Kimia UNESA(Universitas Negeri Surabaya)	ttaufikurohmah@yahoo.com
2	Farah Medina	Mahasiswa	Jurusan Farmasi Universitas Airlangga	farahmedina@icloud.com

Tabel 5.16. Prediksi Pengguna dengan Prediksi Sistem

No	Pengguna	Jenis Uji	Prediksi Pengguna(mg/l)	Prediksi Sistem
1	Dr. Titik Taufikurohmah, S.Si., M.Si.	Rhodamin B	7	2
		Rhodamin B	2	2
		Rhodamin B	10	10
		Rhodamin B	30	30
2	Farah Medina	Rhodamin B	10	10
		Rhodamin B	2	2
		Borax	0 / 50	50
		Borax	100	200

Skenario pertama adalah pengguna menentukan nilai kadar sampel dengan membandingkan warna standar dan warna dari sejumlah sampel sesuai dengan jenis uji yang dipilih. Lalu hasil dari

prediksi pengguna akan dibandingkan dengan hasil prediksi sistem. Tabel 5.16 menunjukkan hasil pengujian sampel untuk setiap pengguna. Pengguna 1 menggunakan sampel dengan jenis uji untuk *Rhodamin B* sebanyak 4 jenis sampel.

Pada sampel pertama pengguna menentukan kelas warna standar 7 mg/l dikarenakan pengguna menganggap warna sampel berada diantara kelas 2 mg/l dan 10 mg/l. Kecuali pada sampel pertama sistem berhasil memprediksi nilai sampel lainnya sesuai dengan prediksi pengguna.

Pengguna nomor 2 menggunakan 2 jenis uji yaitu untuk *Rhodamin B* dan *Borax* masing-masing sebanyak 2 sampel. Pada sampel *borax* pertama pengguna 2 menentukan nilai di antara 0 atau 50. Sistem berhasil memprediksi sampel sesuai dengan prediksi pengguna kecuali pada sampel *borax* yang ke-2.

5.8 Evaluasi

Pada subbab ini akan dijelaskan evaluasi dari pengujian fungsionalitas, pengujian akurasi, pengujian skalabilitas, pengujian kegunaan, evaluasi performas sistem dan perbandingan sistem dengan alat lain yang sejenis.

5.8.1 Evaluasi Pengujian Fungsionalitas

Pengujian fungsionalitas yang telah dilakukan memberikan hasil yang sesuai dengan skenario yang telah direncanakan. Evaluasi pengujian pada masing-masing fungsionalitas dijelaskan dengan sebagai berikut.

1. Pengujian mengukur kadar senyawa pada sampel sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PW-01 yang memberikan hasil dari uji makanan berupa prediksi kandungan senyawa dari sampel.

2. Pengujian memilih jenis uji sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PW-02 yang

memberikan informasi bahwa proses memilih jenis uji berjalan dengan benar.

3. Pengujian menyimpan hasil uji ke *log* sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PW-03 yang memberikan informasi bahwa proses menyimpan hasil uji berjalan dengan benar.
4. Pengujian menampilkan *log* sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PW-04 yang memberikan informasi bahwa proses menampilkan *log* yang terakhir disimpan berjalan dengan benar.
5. Pengujian mengganti *log* tampil ke *log* berikutnya sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PW-05 yang memberikan informasi bahwa proses mengubah *log* tampil dengan *log* berikutnya pada daftar *log* yang tersimpan berjalan dengan benar.
6. Pengujian menghapus *log* tampil sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PW-06 yang memberikan informasi bahwa proses menghapus *log* yang tampil berjalan dengan benar.
7. Pengujian mengirimkan *log* ke aplikasi Manajemen Log sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PW-07 yang memberikan informasi bahwa proses mengirimkan *log* yang tersimpan di sistem ke aplikasi web berjalan dengan benar.
8. Pengujian menghapus semua *log* sistem sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PW-08 yang memberikan informasi bahwa proses menghapus seluruh *log* sistem melalui aplikasi web berjalan dengan benar.

5.8.2 Evaluasi Pengujian Akurasi

Pengujian akurasi yang dilakukan menghasilkan akurasi pengujian untuk setiap jenis uji. Evaluasi dilakukan dengan merekapitulasi setiap akurasi dari jenis uji tersebut. Tabel 5.17 menunjukkan rekapitulasi dari setiap hasil uji akurasi pada tiap jenis uji.

Tabel 5.17. Rekapitulasi Akurasi Setiap Jenis Uji

No	Jenis Uji	Akurasi
1	Boraks	100.0%
2	Rhodamin B	86.667%
3	Merkuri	90.0%
4	Formalin	80.0%
5	*Methanyl Yellow*	88.571%

Kolom kedua menunjukkan jenis uji dan kolom ketiga menunjukkan hasil akurasi dari pengujian yang dilakukan. Pada pengujian kali ini rata-rata dari akumulasi semua jenis uji jika disatukan maka menghasilkan akurasi sebesar 89.047%.

5.8.3 Evaluasi Pengujian Skalabilitas

Prosedur pengujian skalabilitas telah berjalan dengan benar. Sesuai dengan skenario dimana salah satu jenis uji yaitu boraks ditambahkan ke dalam sistem melalui config_path.txt.

Uji makanan dengan boraks sudah dapat dilakukan setelah ditambahkan semua aset dan daftar kelas ditambahkan ke dalam sistem. Namun pengujian ini hanya untuk pengembang yang paham tentang protokol SSH untuk terhubung dengan sistem dan lingkungan sistem operasi Linux.

5.8.4 Evaluasi Pengujian Kegunaan dengan Skenario

Pengujian dilakukan menggunakan salah satu reagen sebenarnya dari Food Security Kit. Reagen yang digunakan adalah reagen untuk Rhodamin B. Kekurangan terdapat pada uji dilakukan oleh penulis tanpa campur tangan dari penguji Food Security Kit sebenarnya. Pengujian dilakukan dengan prosedur dari uji Rhodamin B yang terdapat pada reagen.

Sampel yang digunakan dipastikan mengandung senyawa Rhodamin B. Sistem mampu mendeteksi perubahan warna dan memprediksi nilai dari kandungan senyawa Rhodamin B pada sampel dengan benar.

5.8.5 Evaluasi Performa Sistem

Performa sistem diukur dengan menghitung waktu yang dibutuhkan untuk menentukan kandungan zat dari sampel. Setiap jenis uji yang terdapat dalam sampel dilakukan sebanyak 5 kali dan dicatat waktunya hingga menampilkan hasil uji.

Tabel 5.18. Pengukuran Waktu Pemrosesan Sistem

Jenis Uji	Waktu Proses (detik)					Rata - rata
	1	2	3	4	5	
Boraks	49.1	50.61	49.01	49.29	49.88	49.578
Formalin	41.19	41.47	41.84	40.81	42.79	41.62
Rhodamin B	43.94	42.96	42.56	41.32	42.3	42.616
Methanyl Yellow	49	46.44	46.16	46.26	48.84	47.34
Merkuri	53.91	53.25	53.8	54.45	54.81	54.044

Tabel 5.18 menunjukkan hasil dari pengukuran waktu yang dibutuhkan sistem untuk melakukan pengujian untuk setiap jenis uji. Terlihat waktu untuk jenis uji yang memiliki daftar warna referensi lebih banyak akan membutuhkan waktu lebih lama. Semua jenis uji membutuhkan waktu lebih dari 40 detik hingga dapat menentukan kadar kandungan dalam sampel.

5.8.6 Evaluasi Perbandingan Sistem dengan Alat Lain Sejenis

Tabel 5.19. Daftar Harga Komponen Sistem

Komponen	Harga
Raspberry Pi Model B Rev 2	Rp. 559.020 / $44 (USD) [16]
Kamera Raspberry Pi	Rp. 400.000
LCD TFT Waveshare 3.5”	Rp. 427.875
Lampu LED (2)	Rp. 5000
Wadah Akrilik	Rp. 150.000
Total	Rp. 1.392.045

Tabel 5.20. Perbandingan Harga

Nama	**Harga**	**Fitur**	**Sensor**
RVLM (Royal Vial Lab Multireader)	10.921 (USD) / 134.299.843,26 (IDR) [17]	• Membaca jumlah kadar mikroba dalam sampel • Dapat melakukan lebih dari satu pengujian sekaligus	• Photo Transistor
Primelab Photometer	833.33(USD)/ 11.091.622,30 (IDR) [18]	• Membaca parameter-parameter sampel air • Terhubung dengan aplikasi *desktop* untuk manajemen jenis uji dan parameter yang ingin dilakukan	• JenColor Sensor
PhotoD-X01	Rp. 1.392.045	• Menemukan kadar senyawa pada sampel sesuai dengan reagen Food Security Kit • Melakukan perbandingan warna antara sampel dengan warna standar	• Kamera Raspberry Pi

Sistem menghasilkan fungsionalitas dan akurasi yang cukup baik untuk melakukan pengujian makanan. Rincian biaya yang

dibutuhkan untuk membangun sistem terdapat pada Tabel 5.19 .Biaya hanya berupa daftar harga komponen belum termasuk biaya produksi dan pembangunan aplikasi sistem. Di pasar terdapat alat lain yang sejenis diantaranya RVLM(*Royal Vial Lab Multireader*) dan Primelab Photometer yang memiliki tujuan berbeda namun memiliki metode yang sama yaitu membandingkan warna dari sampel. Jika dibandingakan dengan alat lain yang sejenis pada Tabel 5.20, Sistem yang dibuat cenderung lebih murah. Sistem diberi nama PhotoD-X01.

5.8.7 Evaluasi Pengujian dengan Pengguna

Sistem menghasilkan akurasi yang signifikan dalam memprediksi kadar dari sampel sesuai deengan jenis uji dan warna standar. Terbukti pada percobaan dengan pengguna 1 sistem berhasil memprediksi dengan sesuai 3 dari 4 sampel yang diuji untuk jenis uji *Rhodamin B*. Pada percobaan pengguna 2 sistem memprediksi dengan sesuai 2 dari 2 sampel jenis uji *Rhodamin B* dan 1 dari 2 sampel *Borax*. Dari segi fungsionalitas pengguna merasa bahwa aplikasi dari sistem telah berfungsi dengan baik dan antarmuka dari sistem mudah untuk dipahami. Dari segi performa pengguna 1 merasa bahwa kecepatan proses cukup baik sedangkan keakuratan perlu ditingkatkan. Sedangkan pengguna 2 merasa kecepatan proses masih lambat dan perlu ditingkatkan. Dari segi proses bisnis untuk uji makanan pengguna 1 merasa bahwa proses bisnis sudah sesuai bila perbedaan diperbaiki untuk jenis sampel yang memiliki konsentrasi zat yang rendah yang mengakibatkan sensitifitas turun. Juga pengguna 2 merasa sistem hanya sesuai jika mendeteksi dan membandingkan warna saja. Jika terdapat sampel mengandung zat lain yang bukan termasuk jenis uji yang dipilih sistem belum mampu mendeteksi perbedaan tersebut. Serta perbandingan warna dirasa kurang akurat untuk menentukan kadar dari suatu zat. Pengguna 2 merasa bahwa proses bisnis dari alat yang digunakan telah sesuai.

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini dijelaskan kesimpulan yang dapat diambil dalam pengerjaan Tugas Akhir dan saran untuk pengembangan lebih lanjut dari sistem yang dibuat ini.

6.1 Kesimpulan

Dalam proses pengerjaan Tugas Akhir dari tahap pendahuluan, kajian pustaka, analisis, perancangan, implementasi dan pengujian sistem alat bantu uji makanan diperoleh kesimpulan sebagai berikut.

1. Desain alat ini dibuat dengan perangkat keras untuk mendukung proses uji makanan.Perangkat keras ini diantaranya Raspberry Pi Model B *Revision* 2, modul kamera Raspberry Pi, 2 lampu LED warna putih, Kabel Lan dengan konektor RJ-45, ekstensi RJ-45, wadah dengan bahan akrilik berwarna hitam dan layar sentuh LCD TFT Waveshare 3.5 inci. Secara keseluruhan bentuk fisik dari alat yang dibuat ditunjukkan pada Gambar 4.25

2. Sistem menggunakan sebuah kontainer berwarna putih yang berbahan plastik untuk tempat tabung reaksi sampel. Kalibrasi dilakukan dengan dua buah lampu LED ditembakkan menyamping ke arah kontainer dengan posisi ke atas. Cahaya yang ditembakkan menyebar ke dalam kontainer dan menghasilkan warna natural. Gambar 4.24 menunjukkan bentuk dari kontainer untuk sampel dan lampu LED yang digunkan.

3. Pada komputer kecil Raspberry Pi ditanamkan aplikasi yang dibuat dengan bahasa pemrograman Java dengan *library* Swing GUI. Pada Swing terdapat fungsi getRGB() yang merupakan salah satu fungsi dari kelas BufferedImage yang terdapat pada Swing. Dibuktikan pada pengujian PW-01

dimana sistem telah dapat menghasilkan prediksi kelas dari sampel dengan perbandingan warna dari gambar sampel dengan warna referensi.

4. Metode Klasifikasi KNN digunakan untuk menentukan kelas dari daftar warna sampel sebagai data uji dan daftar warna referensi sebagai data latih. Terbukti pada pengujian PW-01 sistem telah dapat menghasilkan prediksi kelas dari sampel.

5. Sistem merespon perintah dengan kelas ServerManager yang berjalan di belakang sistem sebagai sebuah *thread* menggunakan *socket* untuk terhubung. Kelas ini akan menunggu perintah atau permintaan yang masuk dan meresponnya. Terbukiti dari pengujian PW-07 dan PW-08 dimana sistem dapat merespon perintah untuk mengirim *log* ataupun menghapus semua *log* yang disimpan.

6.2 Saran

Berikut ini merupakan beberapa saran mengenai pengembangan lebih lanjut sistem alat bantu uji makanan berdasarkan hasil rancangan, implementasi dan uji coba yang telah dilakukan.

1. Dibutuhkan perbaikan dari perangkat keras sistem sehingga bentuk sistem lebih kokoh dan mampu menahan guncangan.

2. Bentuk dari wadah dapat diperbaiki untuk mengurangi gangguan cahaya ketika mengambil gambar sampel sehingga dapat meningkatkan akurasi sistem.

3. Perbaikan tampilan antarmuka dan panduan penggunaan sistem dalam pengujian makanan untuk memberikan kemudahan bagi pengguna.

4. Aplikasi memiliki kelemahan dalam memproses gambar sampel dengan metode KNN membuthukan waktu yang lama. Untuk itu dibutuhkan pengoptimalan dari algoritma yang digunakan.

5. Aplikasi ini masih memiliki kelemahan dalam membedakan kelas warna yang sangat mirip. Untuk itu dibutuhkan metode yang lebih baik untuk membandingkan warna tersebut.
6. Sistem sebaiknya mampu memprediksi nilai dari rentang warna standar. Tidak hanya nilai yang terdapat dari warna standar. Karena terdapat warna yang kelasnya bisa berada diantara 2 kelas yang berdekatan.
7. Pengujian dari sistem sebaiknya dilakukan di tempat yang lebih steril. Hal ini dilakukan untuk mengurangi kontaminasi dan gangguna dari faktor luar.

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini dijelaskan kesimpulan yang dapat diambil dalam pengerjaan Tugas Akhir dan saran untuk pengembangan lebih lanjut dari sistem yang dibuat ini.

6.1 Kesimpulan

Dalam proses pengerjaan Tugas Akhir dari tahap pendahuluan, kajian pustaka, analisis, perancangan, implementasi dan pengujian sistem alat bantu uji makanan diperoleh kesimpulan sebagai berikut.

1. Desain alat ini dibuat dengan perangkat keras untuk mendukung proses uji makanan.Perangkat keras ini diantaranya Raspberry Pi Model B *Revision* 2, modul kamera Raspberry Pi, 2 lampu LED warna putih, Kabel Lan dengan konektor RJ-45, ekstensi RJ-45, wadah dengan bahan akrilik berwarna hitam dan layar sentuh LCD TFT Waveshare 3.5 inci. Secara keseluruhan bentuk fisik dari alat yang dibuat ditunjukkan pada Gambar 4.25

2. Sistem menggunakan sebuah kontainer berwarna putih yang berbahan plastik untuk tempat tabung reaksi sampel. Kalibrasi dilakukan dengan dua buah lampu LED ditembakkan menyamping ke arah kontainer dengan posisi ke atas. Cahaya yang ditembakkan menyebar ke dalam kontainer dan menghasilkan warna natural. Gambar 4.24 menunjukkan bentuk dari kontainer untuk sampel dan lampu LED yang digunkan.

3. Pada komputer kecil Raspberry Pi ditanamkan aplikasi yang dibuat dengan bahasa pemrograman Java dengan *library* Swing GUI. Pada Swing terdapat fungsi getRGB() yang merupakan salah satu fungsi dari kelas BufferedImage yang terdapat pada Swing. Dibuktikan pada pengujian PW-01

dimana sistem telah dapat menghasilkan prediksi kelas dari sampel dengan perbandingan warna dari gambar sampel dengan warna referensi.

4. Metode Klasifikasi KNN digunakan untuk menentukan kelas dari daftar warna sampel sebagai data uji dan daftar warna referensi sebagai data latih. Terbukti pada pengujian PW-01 sistem telah dapat menghasilkan prediksi kelas dari sampel.

5. Sistem merespon perintah dengan kelas ServerManager yang berjalan di belakang sistem sebagai sebuah *thread* menggunakan *socket* untuk terhubung. Kelas ini akan menunggu perintah atau permintaan yang masuk dan meresponnya. Terbukiti dari pengujian PW-07 dan PW-08 dimana sistem dapat merespon perintah untuk mengirim *log* ataupun menghapus semua *log* yang disimpan.

6.2 Saran

Berikut ini merupakan beberapa saran mengenai pengembangan lebih lanjut sistem alat bantu uji makanan berdasarkan hasil rancangan, implementasi dan uji coba yang telah dilakukan.

1. Dibutuhkan perbaikan dari perangkat keras sistem sehingga bentuk sistem lebih kokoh dan mampu menahan guncangan.

2. Bentuk dari wadah dapat diperbaiki untuk mengurangi gangguan cahaya ketika mengambil gambar sampel sehingga dapat meningkatkan akurasi sistem.

3. Perbaikan tampilan antarmuka dan panduan penggunaan sistem dalam pengujian makanan untuk memberikan kemudahan bagi pengguna.

4. Aplikasi memiliki kelemahan dalam memproses gambar sampel dengan metode KNN membuthukan waktu yang lama. Untuk itu dibutuhkan pengoptimalan dari algoritma yang digunakan.

5. Aplikasi ini masih memiliki kelemahan dalam membedakan kelas warna yang sangat mirip. Untuk itu dibutuhkan metode yang lebih baik untuk membandingkan warna tersebut.
6. Sistem sebaiknya mampu memprediksi nilai dari rentang warna standar. Tidak hanya nilai yang terdapat dari warna standar. Karena terdapat warna yang kelasnya bisa berada diantara 2 kelas yang berdekatan.
7. Pengujian dari sistem sebaiknya dilakukan di tempat yang lebih steril. Hal ini dilakukan untuk mengurangi kontaminasi dan gangguna dari faktor luar.

LAMPIRAN A – PANDUAN PENGGUNA

Berikut adalah panduan untuk pengguna PhotoD-X01 untuk melakukan uji sampel :

1. Siapkan sampel pada tabung reaksi dengan takaran minimal 5 ml.

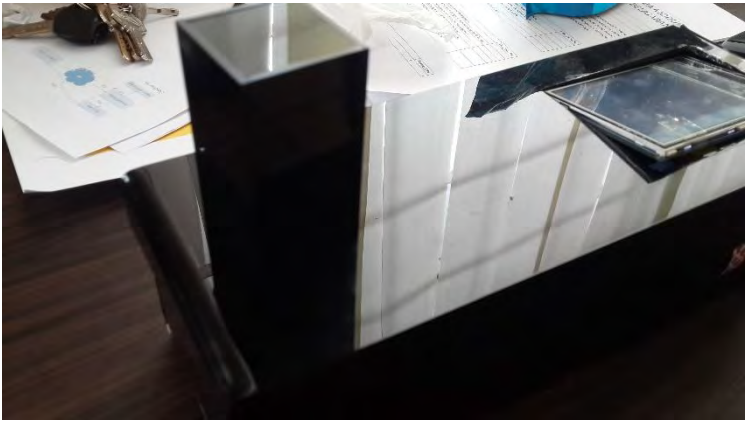
Gambar A.1. Sampel dalam Tabung Reaksi

2. Bersihkan bagian luar tabung reaksi dengan tisu bersih untuk mengoptimalkan gambar sampel yang diambil

3. Masukan ke dalam lubang utama alat PhotoD-X01

Gambar A.2. Masukan Sampel

4. Gunakan penutup yang disediakan untuk meminimalisir gangguan cahaya

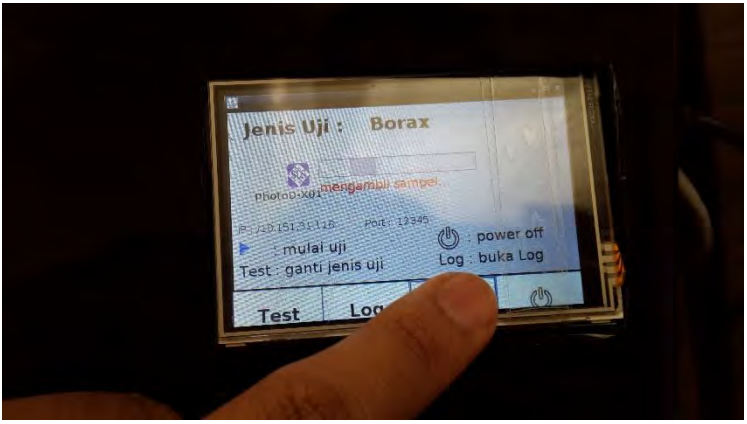


Gambar A.3. Tutup Lubang Sampel

5. Pilih jenis uji yang diinginkan dengan menekan tombol Test untuk mengganti jenis uji yang tampil

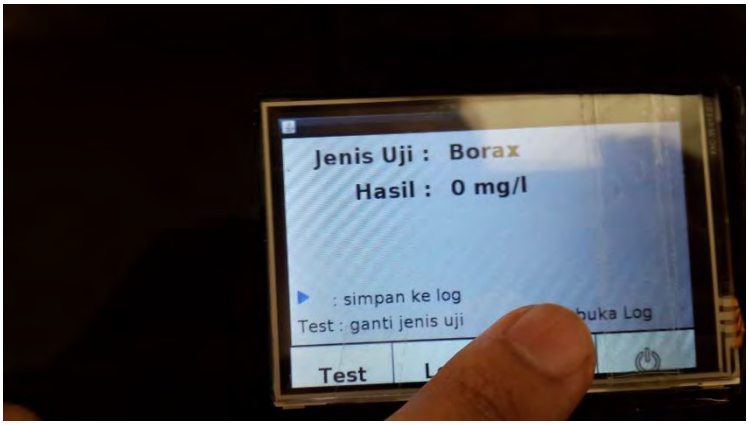
Gambar A.4. Pilih Jenis Uji

6. Klik tombo ketiga atau tombol play untuk memulai uji yang dipilih



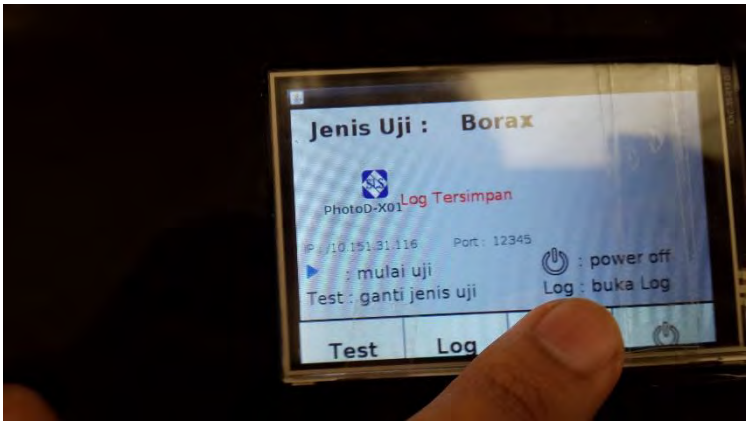
Gambar A.5. Mulai Uji Tekan Tombol Play

7. Hasil uji akan tampil setelah progress selesai



Gambar A.6. Hasil Uji Tampil

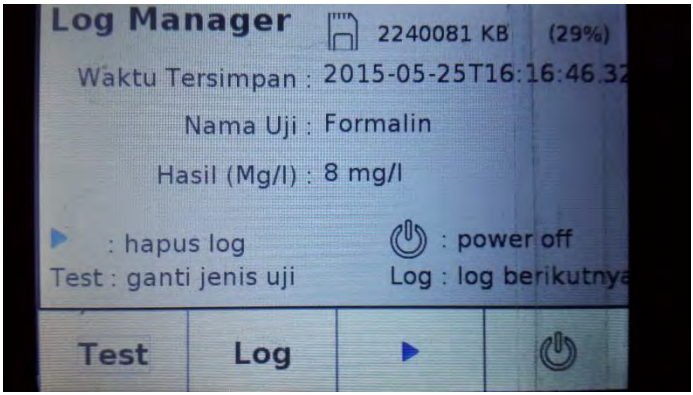
8. Klik tombol ketiga atau tombol play untuk menyimpan hasil uji ke log atau klik tombol lain untuk tidak menyimpan log dan berpindah ke menu lainnya



Gambar A.7. Menyimpan Hasil Uji

Berikut adalah panduan untuk pengguna PhotoD-X01 untuk mengatur log :

1. Klik tombol Log untuk membuka menu log
2. Log yang tampil adalah log yang terakhir kali disimpan, jika belum pernah mengisi log makan tidak ada log yang tampil dan notifikasi muncul



Gambar A.8. Tampilan Manajemen *Log*

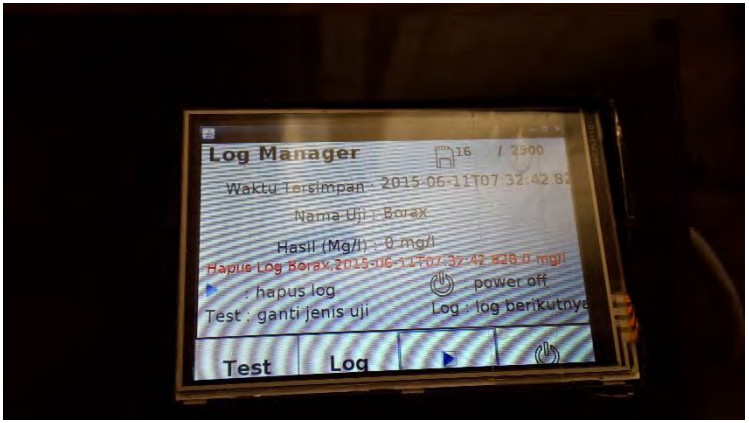
Gambar A.9. Tampilan ika *Log* Kosong

3. Klik tombol Log untuk mengganti *log* yang tampil

Gambar A.10. Ganti *Log* Tampil

4. Klik tombol ketiga atau tombol play untuk menghapus log yang tampil

5. Log tampil akan diganti dan notifikasi muncul

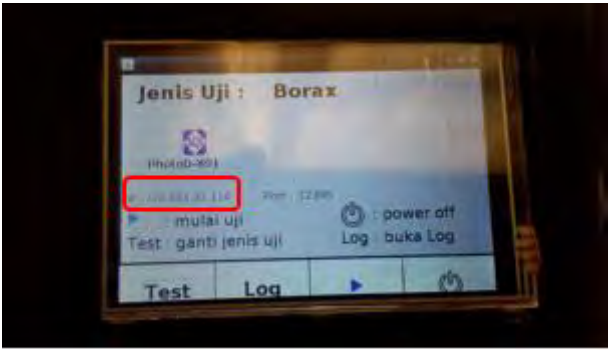


Gambar A.11. Log Terhapus

LAMPIRAN B – PANDUAN PENGEMBANG

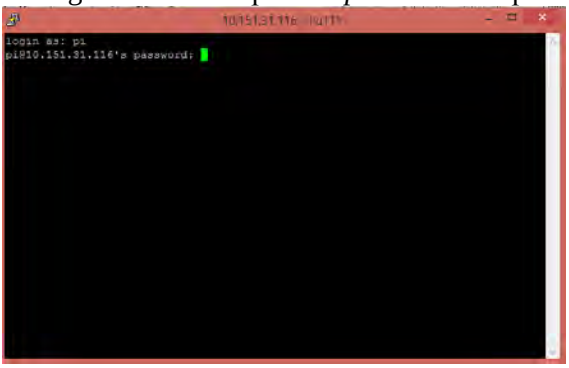
Berikut adalah panduan pengembang untuk mengambil gambar warna referensi jenis uji yang baru :

1. Sambungkan alat PhotoD-X01 dengan kabel LAN(*Local Area Network*) dalam satu jaringan pada konektor yang tersedia
2. Buka koneksi SSH(bisa menggunakan Putty) dengan memasukan alamat IP yang tampil di antarmuka pengujian



Gambar B.1. IP yang Tertera pada PhotoD-X01

3. Masuk dengan *username* “pi” dan *password* “raspberry”



Gambar B.2. Masuk ke Pi

4. Siapkan warna referensi ke dalam tabung reaksi

Gambar B.3. Warna Referensi ke Tabung Reaksi

5. Ambil gambar dari setiap warna referensi menggunakan PhotoD-X01 dengan format perintah pada terminal raspistill –vf -hf –o [path_direktori]/[namafile.format] –w 480 –h 360.

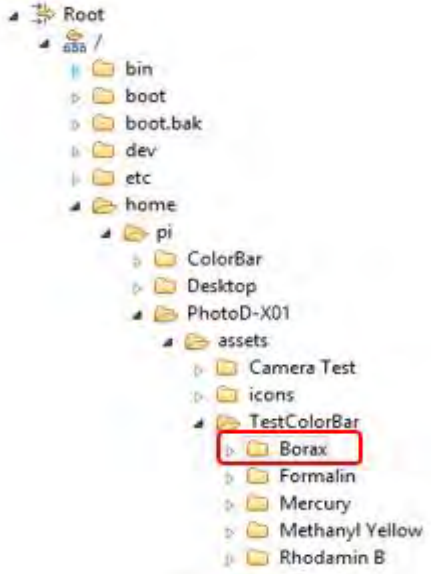
```
raspistill -vf -hf -o /home/pi/ColorBar/Borax/0.jpg -w
480 -h 360
raspistill -vf -hf -o /home/pi/ColorBar/Borax/50.jpg -w
480 -h 360
raspistill -vf -hf -o /home/pi/ColorBar/Borax/100.jpg -
w 480 -h 360
raspistill -vf -hf -o /home/pi/ColorBar/Borax/150.jpg -
w 480 -h 360
raspistill -vf -hf -o /home/pi/ColorBar/Borax/200.jpg -
w 480 -h 360
raspistill -vf -hf -o /home/pi/ColorBar/Borax/5000.jpg -
w 480 -h 360
raspistill -vf -hf -o /home/pi/ColorBar/Borax/10000.jpg
-w 480 -h 360
```

Gambar B.4. Contoh Daftar Perintah Ambil Gambar

6. Gambar disimpan ke dalam direktori sesuai yang disebutkan pada setiap baris perintah Gambar B.4.

Berikut adalah panduan pengembang untuk menambah jenis uji baru :

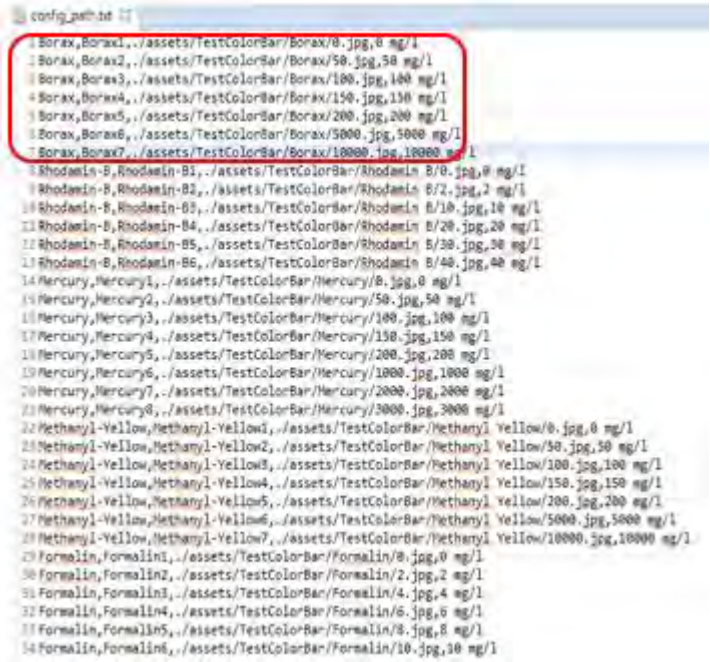
1. Pindahkan direktori yang berisi aset gambar warna referensi yang telah diambil sebelumnya ke direktori /home/pi/PhotoD-X01/assets/TestColorBar/ yang terdapat dalam PhotoD-X01



Gambar B.5. Pindahkan Direktori Gambar ke Direktori assets

2. Tambahkan daftar warna referensi baru tersebut ke dalam file config_path.txt yang terdapat dalam direktori /home/pi/PhotoD-X01/ dengan format [nama_jenis

_uji],[nama_jenis_uji] [indeks_gambar], [path_*file*_gambar],[nilai_kandungan]

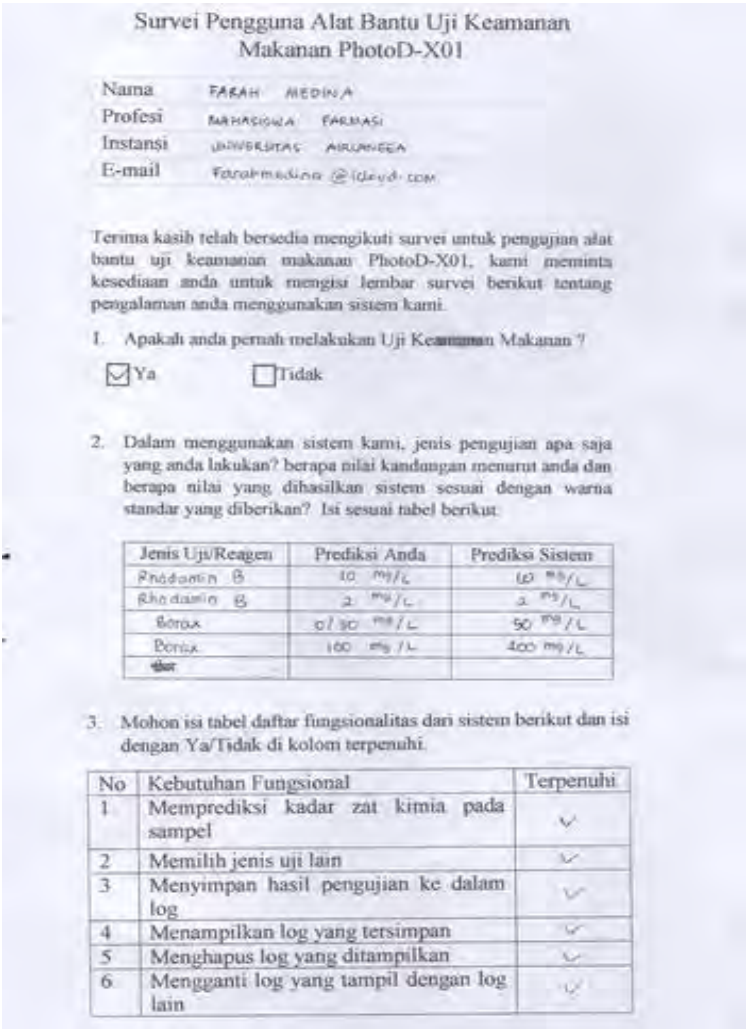


Gambar B.6. Daftar Warna Referensi config_path.txt

3. Muat ulang daftar aset dengan melakukan reboot pada PhotoD-X01

4. Jenis uji yang baru dapat dipilih pada antarmuka pengujian

LAMPIRAN C – LEMBAR SURVEI DAN FEEDBACK



Survei Pengguna Alat Bantu Uji Keamanan Makanan PhotoD-X01

Nama FARAH MEDINA
Profesi MAHASISWA FARMASI
Instansi UNIVERSITAS AIRLANGGA
E-mail farahmedina@ideyd.com

Terima kasih telah bersedia mengikuti survei untuk pengujian alat bantu uji keamanan makanan PhotoD-X01, kami meminta kesediaan anda untuk mengisi lembar survei berikut tentang pengalaman anda menggunakan sistem kami.

1. Apakah anda pernah melakukan Uji Keamanan Makanan ?

☑ Ya ☐ Tidak

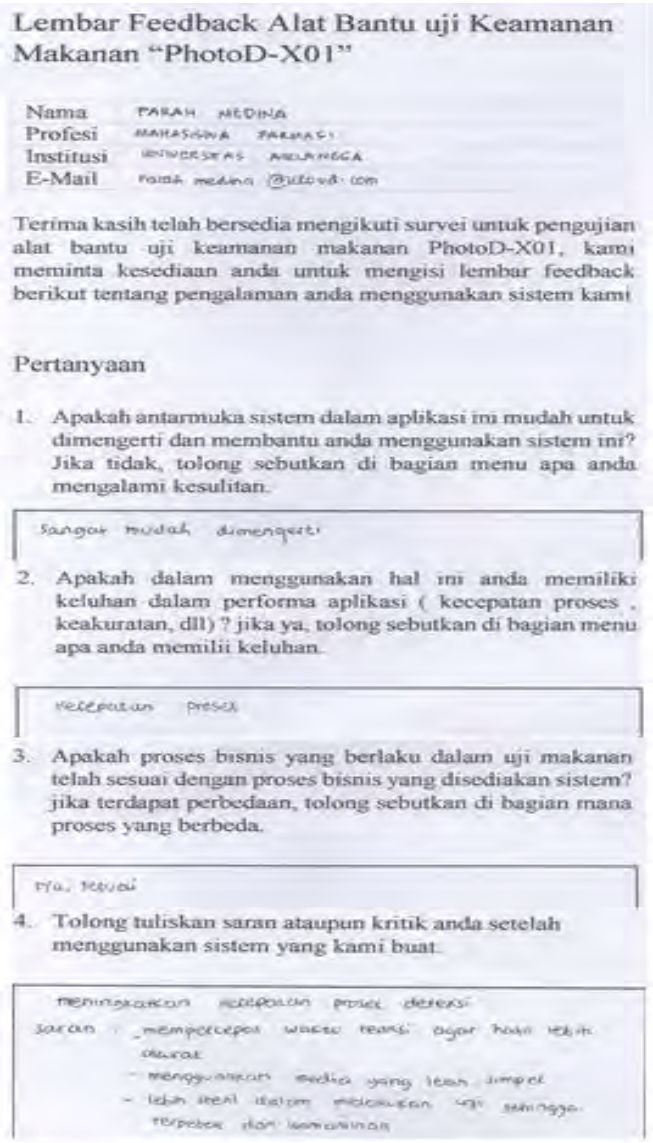
2. Dalam menggunakan sistem kami, jenis pengujian apa saja yang anda lakukan? berapa nilai kandungan menurut anda dan berapa nilai yang dihasilkan sistem sesuai dengan warna standar yang diberikan? Isi sesuai tabel berikut

Jenis Uji/Reagen	Prediksi Anda	Prediksi Sistem
Rhodamin B	10 mg/L	10 mg/L
Rhodamin B	2 mg/L	2 mg/L
Borax	0/50 mg/L	50 mg/L
Borax	100 mg/L	400 mg/L
dst		

3. Mohon isi tabel daftar fungsionalitas dari sistem berikut dan isi dengan Ya/Tidak di koloom terpenuhi.

No	Kebutuhan Fungsional	Terpenuhi
1	Memprediksi kadar zat kimia pada sampel	✓
2	Memilih jenis uji lain	✓
3	Menyimpan hasil pengujian ke dalam log	✓
4	Menampilkan log yang tersimpan	✓
5	Menghapus log yang ditampilkan	✓
6	Mengganti log yang tampil dengan log lain	✓

Gambar C.1. Lembar Survei Pengguna 1



Lembar Feedback Alat Bantu uji Keamanan Makanan "PhotoD-X01"

Nama FARAH MEDINA
Profesi MAHASISWA FARMASI
Institusi UNIVERSITAS MULAWARMAN
E-Mail farah.medina@ubt.ac.id

Terima kasih telah bersedia mengikuti survei untuk pengujian alat bantu uji keamanan makanan PhotoD-X01, kami meminta kesediaan anda untuk mengisi lembar feedback berikut tentang pengalaman anda menggunakan sistem kami

Pertanyaan

1. Apakah antarmuka sistem dalam aplikasi ini mudah untuk dimengerti dan membantu anda menggunakan sistem ini? Jika tidak, tolong sebutkan di bagian menu apa anda mengalami kesulitan.

sangat mudah dimengerti

2. Apakah dalam menggunakan hal ini anda memiliki keluhan dalam performa aplikasi (kecepatan proses , keakuratan, dll) ? jika ya, tolong sebutkan di bagian menu apa anda memilii keluhan.

kecepatan proses

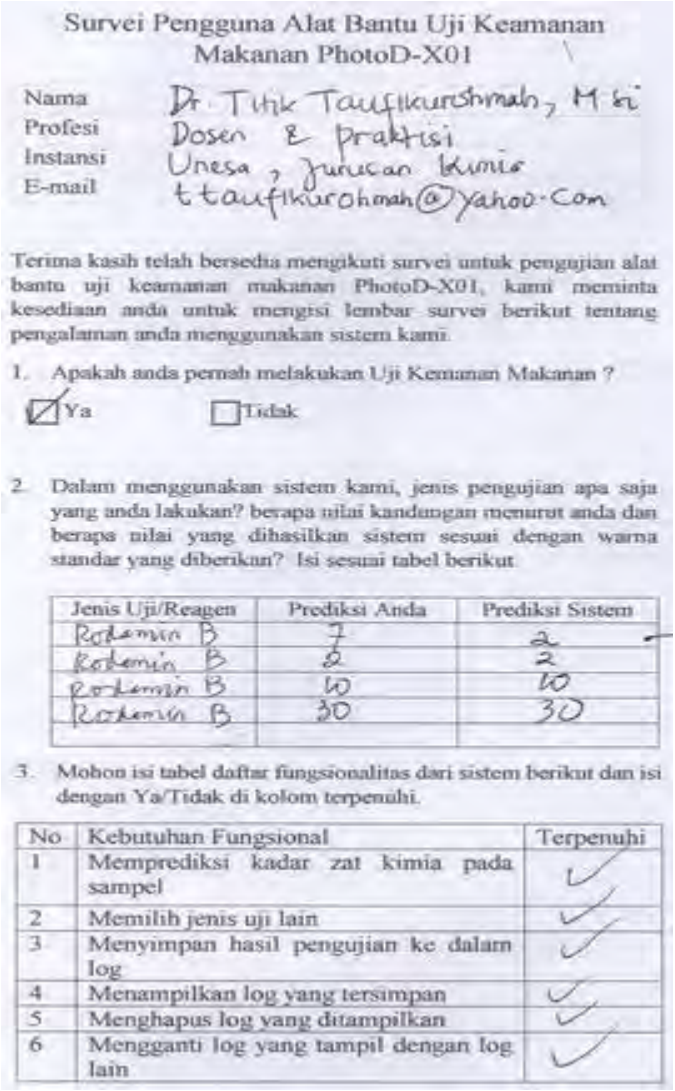
3. Apakah proses bisnis yang berlaku dalam uji makanan telah sesuai dengan proses bisnis yang disediakan sistem? jika terdapat perbedaan, tolong sebutkan di bagian mana proses yang berbeda.

n/a, sesuai

4. Tolong tuliskan saran ataupun kritik anda setelah menggunakan sistem yang kami buat.

meningkatkan kecepatan proses deteksi
saran : - mempercepat waktu reaksi agar hasil lebih akurat
- menggunakan media yang lebih simpel
- lebih teliti dalam melakukan uji sehingga terhindar dari kesalahan

Gambar C.2. Lembar *Feedback* Pengguna 1



Survei Pengguna Alat Bantu Uji Keamanan Makanan PhotoD-X01

Nama: Dr. Titik Taufikurohmah, M.Si
Profesi: Dosen & praktisi
Instansi: Unesa, jurusan kimia
E-mail: ttaufikurohmah@yahoo.com

Terima kasih telah bersedia mengikuti survei untuk pengujian alat bantu uji keamanan makanan PhotoD-X01, kami meminta kesediaan anda untuk mengisi lembar survei berikut tentang pengalaman anda menggunakan sistem kami.

1. Apakah anda pernah melakukan Uji Kemanan Makanan ?
   ☑ Ya ☐ Tidak

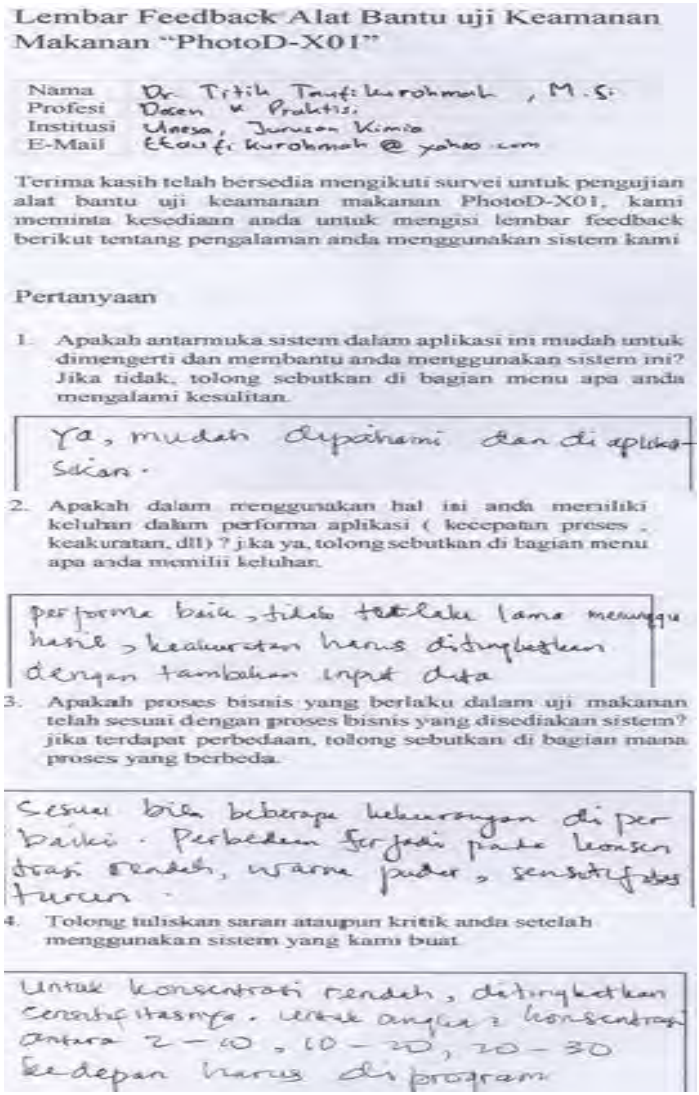
2. Dalam menggunakan sistem kami, jenis pengujian apa saja yang anda lakukan? berapa nilai kandungan menurut anda dan berapa nilai yang dihasilkan sistem sesuai dengan warna standar yang diberikan? Isi sesuai tabel berikut

Jenis Uji/Reagen	Prediksi Anda	Prediksi Sistem
Rodamin B	1	2
Rodamin B	2	2
Rodamin B	10	10
Rodamin B	30	30

3. Mohon isi tabel daftar fungsionalitas dari sistem berikut dan isi dengan Ya/Tidak di kolom terpenuhi.

No	Kebutuhan Fungsional	Terpenuhi
1	Memprediksi kadar zat kimia pada sampel	✓
2	Memilih jenis uji lain	✓
3	Menyimpan hasil pengujian ke dalam log	✓
4	Menampilkan log yang tersimpan	✓
5	Menghapus log yang ditampilkan	✓
6	Mengganti log yang tampil dengan log lain	✓

Gambar C.3. Lembar Survei Pengguna 2



Lembar Feedback Alat Bantu uji Keamanan Makanan "PhotoD-X01"

Nama	Dr. Titik Taufikurohmah, M.Si
Profesi	Dosen & Praktisi
Institusi	Unesa, Jurusan Kimia
E-Mail	titaufikurohmah@yahoo.com

Terima kasih telah bersedia mengikuti survei untuk pengujian alat bantu uji keamanan makanan PhotoD-X01, kami meminta kesediaan anda untuk mengisi lembar feedback berikut tentang pengalaman anda menggunakan sistem kami

Pertanyaan

1. Apakah antarmuka sistem dalam aplikasi ini mudah untuk dimengerti dan membantu anda menggunakan sistem ini? Jika tidak, tolong sebutkan di bagian menu apa anda mengalami kesulitan.

Ya, mudah dipahami dan di aplikasikan.

2. Apakah dalam menggunakan hal ini anda memiliki keluhan dalam performa aplikasi (kecepatan proses , keakuratan, dll) ? jika ya, tolong sebutkan di bagian menu apa anda memilii keluhan.

performa baik, tidak terlalu lama menunggu hasil, keakuratan harus ditingkatkan dengan tambahan input data

3. Apakah proses bisnis yang berlaku dalam uji makanan telah sesuai dengan proses bisnis yang disediakan sistem? jika terdapat perbedaan, tolong sebutkan di bagian mana proses yang berbeda.

Sesuai bila beberapa kekurangan di perbaiki. Perbedaan terjadi pada konsentrasi rendah, warna pudar, sensitifitas turun.

4. Tolong tuliskan saran ataupun kritik anda setelah menggunakan sistem yang kami buat.

Untuk konsentrasi rendah, ditingkatkan sensitifitasnya. Untuk angka : konsentrasi antara 2 – 10, 10 – 20, 20 – 30 kedepan harus di program

Gambar C.4. Lembar *Feedback* Pengguna 2

DAFTAR PUSTAKA

[1] BPOM, "Laporan Tahunan," 4 Juli 2013. [Online]. Available: http://www.pom.go.id/new/index.php/browse/laporan_tahunan/11 -12-2004/11 -12-2014/1.. [Accessed 11 Desember1 2014].

[2] R. Biotech, "RVLM User Manual," 2014. [Online]. Available: http://www.royalbiotech.com/royal-viallab-mutireader.html. [Accessed 18 Desembe 2014].

[3] J. Kalla, "Jurnal Negarawan," 2009. [Online]. Available: http://www.setneg.go.id/index.php?option=com_content&task=view&id=6378&Itemid=286.. [Accessed 11 Desember 2014].

[4] R. Hub, "Raspberry Pi Wiki," 19 Novermber 2014. [Online]. Available: http://elinux.org/RPi_Hub. [Accessed 18 November 2014].

[5] C. Clay, "Raspberry Pi :11 reasons why it's the perfect small server," 14 Januari 2014. [Online]. Available: http://www.zdnet.com/article/raspberry-pi-11 -reasons-why-its-the-perfect-small-server. [Accessed 18 Desember 2014].

[6] R. Savege, "The Pi4J Project," Raspberry Pi, 2012. [Online]. Available: http://pi4j.com/. [Accessed 3 February 2015].

[7] Harwanti, "BAHAN TAMBAHAN PADA PANGAN DAN BAHAYANYA (FORMALIN, BORAKS," 4 November 2014. [Online]. Available: http://portal.bangkabaratkab.go.id/id/informasi/bahan-tambahan-pada-pangan-dan-bahayanya-formalinboraks-dan-pewarna-buatan.. [Accessed 18 Desember 2014].

[8] CV. Surya Pratama Gemilang, Food Test Kits, Bogor.

[9] Z. Thamrin, "Analisis Zat Pemanis Buatan (Sakarin dan Siklamat) Pada Pangan Jajanan Di SD Kompleks," 1 Oktober 2014. [Online]. Available:

http://repository.unhas.ac.id/handle/123456789/11379. [Accessed 18 Desember 2014].

[10] "What Is a Socket?," Oracle, [Online]. Available: https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html. [Accessed 30 April 2015].

[11] "PrimeLab Brochure," PrimeLab, 2003. [Online]. Available: http://www.primelab.org/en/brochure.html. [Accessed 3 Februari 2015].

[12] "How to Setup an LCD Touchscreen on the Raspberry Pi," Circuit Basics, 6 Februari 2015. [Online]. Available: http://www.circuitbasics.com/setup-lcd-touchscreen-raspberry-pi/. [Accessed 30 Maret 2015].

[13] N. Tronnes, "Linux Framebuffer drivers for small TFT LCD display modules," 2013. [Online]. Available: https://github.com/notro/fbtft. [Accessed 30 Maret 2015].

[14] N. Tronnes, "fbtft_device.c," 2013. [Online]. Available: https://github.com/notro/fbtft/blob/master/fbtft_device.c. [Accessed 30 Maret 2015].

[15] N. Tronnes, "Support for Waveshare 3.5 Spotpear," 6 Januari 2015. [Online]. Available: https://github.com/notro/fbtft/issues/215. [Accessed 30 Maret 2015].

[16] J. Vähä-Herttua, "I'd like to have some LCD on my Pi," futurice, 26 Maret 2015. [Online]. Available: http://futurice.com/blog/id-like-to-have-some-lcd-on-my-pi. [Accessed 2015 Maret 30].

[17] "Raspberry Pi Model B 756-8308 Motherboard," Amazon, [Online]. Available: http://www.amazon.com/Raspberry-Pi-756-8308-Motherboard-RASPBRRYPCBA512/dp/B009SQQF9C. [Accessed 30 May 2015].

[18] "Item # PLP001, PrimeLab Photometer," AquaPhoenix Scientific Inc, [Online]. Available:

http://catalog.aquaphoenixsci.com/item/all-categories-meters-colorimeters-2/eters-colorimeters-primelab-multi-range-photometer/plp001?. [Accessed 30 Mei 2015].

[19] "RVLM (Royal Vial Lab Multi_reader)," GENTAUR, [Online]. Available: http://antibody-antibodies.com/product1055792-Royal_Biotech__RB-RVLM_(Royal_Vial_Lab_Multi_reader).html. [Accessed 30 Mei 2015].

BIODATA PENULIS

Muhammad Ruslan Hafiz Syauqi, lahir di Jakarta, pada tanggal 2 April 1993. Penulis menempuh pendidikan mulai dari SDN Padurenan VI Bekasi (1999-2005), SMP Al-Azhar Bekasi (2005-2008), SMAN 30 Jakarta (2008-2011) dan S1 Teknik Informatika ITS (2011-2015). Selama masa kuliah, penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer (HMTC). Diantaranya adalah menjadi staff departemen Pengembangan Profesi himpunan mahasiswa teknik computer ITS 2012-2013 dan staff ahli departemen Pengembangan Profesi himpunan mahasiswa teknik computer ITS 2013-2014. Penulis juga aktif dalam kegiatan kepanitiaan Schematics. Diantaranya penulis pernah menjadi panitia National Seminar of Technology (NST) Schematics 2013. Selama kuliah di teknik informatika ITS, penulis mengambil bidang minat Rekayasa Perangkat Lunak (RPL). Penulis pernah menjadi administrator Laboratorium Rekayasa Perangkat Lunak Informatika ITS. Komunikasi dengan penulis dapat melalui email: mruslanhafiz@gmail.com.