



Hak cipta dan penggunaan kembali:

Lisensi ini mengizinkan setiap orang untuk menggubah, memperbaiki, dan membuat ciptaan turunan bukan untuk kepentingan komersial, selama anda mencantumkan nama penulis dan melisensikan ciptaan turunan dengan syarat yang serupa dengan ciptaan asli.

Copyright and reuse:

This license lets you remix, tweak, and build upon work non-commercially, as long as you credit the origin creator and license it on your new creations under the identical terms.

BAB III

RANCANGAN SISTEM

3.1 Fitur dan Spesifikasi Smart Dog Feeder

Fitur dan spesifikasi Smart Dog Feeder yang dirancang adalah sebagai berikut.

- 1) Pengguna dapat mengakses *Smart Dog Feeder* melalui *smartphone* Android;
- 2) *Smart Dog Feeder* dapat memberikan makanan sesuai dengan jadwal yang diatur oleh pengguna;
- 3) *Smart Dog Feeder* dan aplikasi Android dapat mengakses data yang sama;
- 4) Komunikasi berlangsung secara *real time*.
- 5) Stok makanan yang dapat ditampung *Smart Dog Feeder* adalah 1 kg;
- 6) *Smart Dog Feeder* hanya dapat diakses oleh anjing yang terdaftar;
- 7) *Smart Dog Feeder* melakukan notifikasi mengenai kegiatan pemberian makan;
- 8) Memiliki tiga takaran saji dan maksimum pemberian makan tiga kali dalam sehari;
- 9) Pembaruan jumlah stok dihitung oleh sistem dan dikonfirmasi oleh *Smart Dog Feeder*.

3.2 Perancangan Sistem Keseluruhan

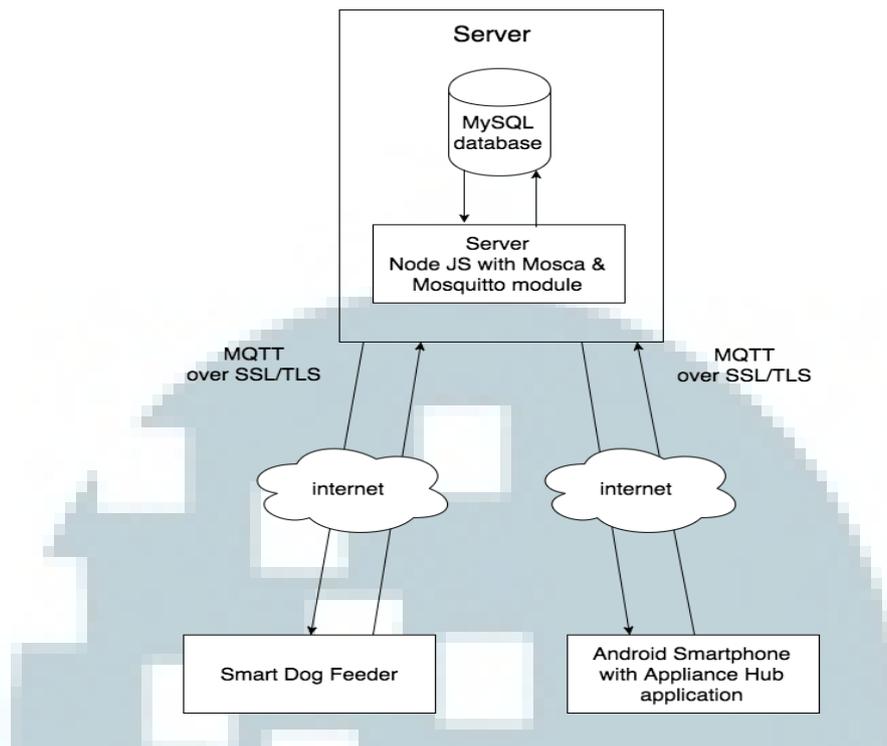
Gambar 3.1 merupakan diagram blok rancangan sistem *Smart Dog Feeder* secara keseluruhan. Pada sistem ini, *Smart Dog Feeder* berkomunikasi dengan server melalui modul WiFi ESP8266 berupa *development board* NodeMCU

dengan protokol MQTT. Data yang didapat oleh ESP8266 dari server akan dikirim dan diolah oleh Arduino UNO melalui komunikasi serial. Untuk mengirimkan perintah ke server, Arduino juga akan mengirimkan data tersebut melalui komunikasi serial sehingga akan diteruskan oleh ESP8266 dan dikirim ke server. Data dalam komunikasi ini berupa *JSON object*.

Pada sisi pengguna, komunikasi ke server juga melalui MQTT dalam aplikasi Android. Pengguna dapat mengakses informasi mengenai perangkat, dirinya, dan mengirimkan perintah. Sebelumnya *user* harus mendaftarkan diri untuk dapat menggunakan aplikasi tersebut dan mendaftarkan perangkat baru. Seluruh perintah dari aplikasi ini akan dikirimkan ke server dalam bentuk *JSON string* kemudian diolah oleh server.

Server bertugas untuk menerima data JSON dari perangkat pintar dan aplikasi Android. Data tersebut kemudian diolah sehingga mendapatkan jenis pesan yang dikirimkan dalam atribut "*msg*" pada *JSON object*. Dari jenis pesan tersebut akan diolah sesuai fungsi pada server dan disimpan dalam *database* untuk digunakan pada instruksi berikutnya.

U
M
N



Gambar 3. 1 Diagram Blok Sistem Smart Dog Feeder

3.3 Perancangan Perangkat Keras Smart Dog Feeder

Komponen-komponen yang digunakan untuk membangun *Smart Dog Feeder* adalah sebagai berikut.

- 1) MFRC522 untuk melakukan otentikasi dengan RFID;
- 2) *Load Cell* diambil dari timbangan emas untuk mengukur porsi makanan

Spesifikasi timbangan:

- Kapasitas Timbangan: 500 gram.
 - Sumber daya: 3V atau 2 buah baterai AAA ;
- 3) Modul HX711 sebagai *amplifier* untuk mendapat hasil pengukuran dari *load cell*;
 - 4) Arduino UNO;
 - 5) Servo DF15RSMG 360° *Continuous Rotation* untuk memutar baling pada Gambar 4.6 sehingga dapat menyajikan makanan;

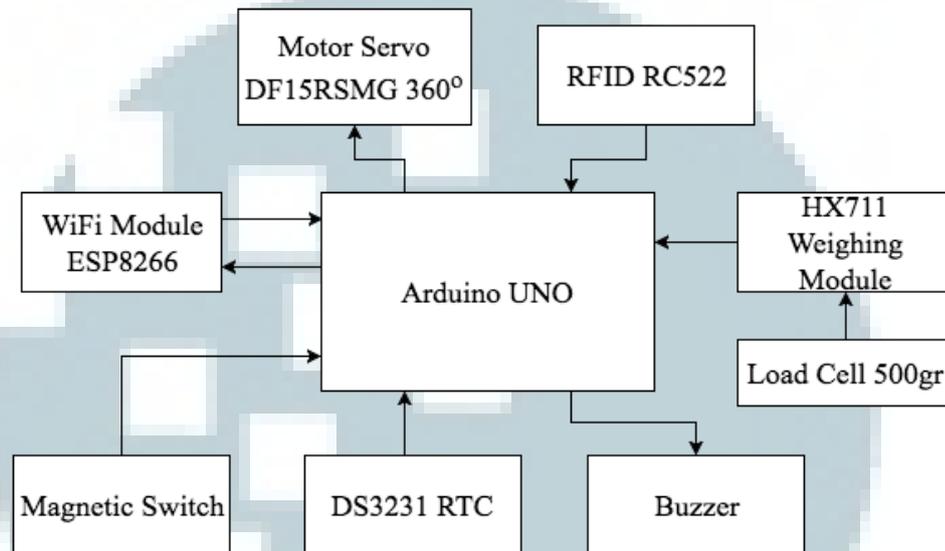
- 6) DS3231 RTC sebagai jam dan alarm;
- 7) Sensor magnet pada tutup untuk mengetahui kondisi tutup (terbuka / tertutup);
- 8) LM2596 untuk menurunkan tegangan 12V ke 5V sebagai *source* bagi seluruh komponen;
- 9) *Screw shield* pada Arduino UNO;
- 10) Modul WiFi ESP8266 berupa *development board* yang telah terpasang *firmware* NodeMCU;
- 11) Adaptor 12V sebagai sumber tegangan.

Gambar 3.1 menggambarkan diagram blok dari *Smart Dog Feeder*. Pada bagian tengah merupakan mikrokontroler yang dipakai pada *Smart Dog Feeder* yaitu Arduino UNO. Arduino UNO mengatur alur program utama, menggerakkan servo, membunyikan *buzzer*, dan mengirimkan *JSON object* secara serial ke ESP8266. Mikrokontroler juga akan menerima masukan berupa pembacaan dari *RFID reader*, modul sensor berat HX711, dan mendapatkan waktu serta alarm dari DS3231 RTC.

RFID reader akan berada pada alat yang berfungsi untuk memastikan bahwa yang mendekat merupakan anjing yang berhak. *RFID tag* akan dipasang pada kalung anjing sehingga saat mendekati *Smart Dog Feeder*, anjing akan dikenali oleh alat.

Perangkat lainnya yang terpasang pada mikrokontroler seperti HX711 merupakan modul yang dipakai untuk mendapatkan nilai pembacaan dari sebuah load cell. Load cell akan berfungsi untuk memastikan berat makanan per *serving* telah sesuai dengan yang diinginkan oleh pengguna (yang disimpan

sebelumnya). *Buzzer* berguna untuk memberi suara bila alat akan mengeluarkan makanan sehingga anjing dapat mengetahui bahwa saat itu adalah waktu makan. DS3231 RTC berguna sebagai jam pada Arduino UNO. RTC akan mengatur *alarm* untuk kemudian diproses oleh Arduino UNO.



Gambar 3. 2 Diagram blok perangkat keras Smart Dog Feeder

Agar *Smart Dog Feeder* dapat berkomunikasi dengan server, maka digunakan NodeMCU yang dihubungkan pada pin serial pada Arduino UNO. NodeMCU akan mengatur segala keperluan yang berhubungan dengan komunikasi antara Arduino UNO dan server.

3.4 Protokol Komunikasi

Dalam protokol komunikasi terdapat dua bagian komunikasi yang terdiri dari komunikasi perangkat keras ke server dan komunikasi perangkat Android ke server.

3.4.1 Komunikasi Perangkat Keras ke Server

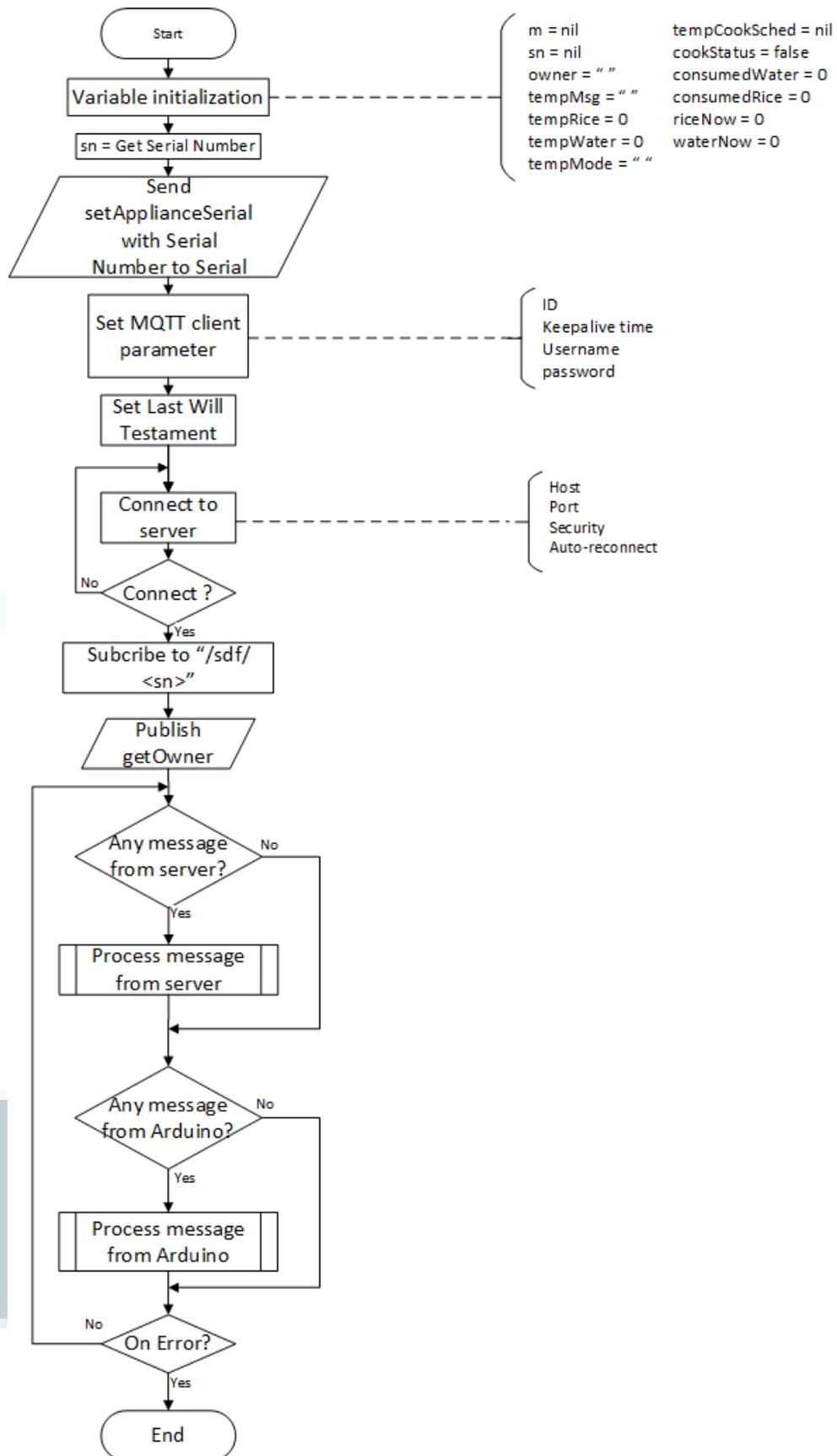
Komunikasi perangkat keras dengan server merupakan komunikasi antara *Smart Dog Feeder* dengan server. Pada *Smart Dog Feeder* dilengkapi dengan

modul WiFi ESP8266 yang berupa *development board* NodeMCU sehingga dapat melakukan komunikasi dengan server melalui protokol MQTT. Keseluruhan jalannya komunikasi akan diolah oleh NodeMCU dengan bahasa pemrograman lua [32].

NodeMCU akan meminta informasi mengenai *owner* ke server pada inisialisasi awal. Kemudian NodeMCU akan mengirimkan perintah *getFeedSchedule* sehingga mendapatkan jadwal pemberian makan dan disampaikan ke Arduino UNO melalui komunikasi serial. Setiap kali pemberian makan dijadwalkan dan selesai dilakukan, NodeMCU juga akan memberikan laporan ke server. Pada Gambar 3.3 dapat dilihat alur program dari NodeMCU.

Dalam komunikasi ini, terdapat beberapa format komunikasi yang dipakai dalam format JSON. Baik server ataupun *Smart Dog Feeder* akan mengambil isi dari atribut *msg* untuk diolah sesuai dengan fungsinya. Berikut adalah format komunikasi yang digunakan.

U M N



Gambar 3. 3 Flowchart NodeMCU (application.lua) untuk Smart Dog Feeder [32]

A. Mengambil Informasi Pemilik Perangkat Pintar

Perintah `getOwner` dipakai untuk mengambil informasi pemilik perangkat pintar. Informasi mengenai `username` ini diperlukan untuk melakukan komunikasi lain. Pesan ini akan dikirimkan melalui topik `/sdf/<serial_number>`. Berikut adalah format pengiriman yang dikirimkan yang dapat dilihat pada Tabel 3.1.

Tabel 3.1 Tabel Format Pengiriman `getOwner` [30]

Name	Value	Keterangan
<code>msg</code>	<code>getOwner</code>	Identifikasi pesan.
<code>applianceserial</code>	<code><applianceserial></code>	Nomor serial perangkat pintar yang didapat dari alamat MAC perangkat pintar.
Contoh: { <code>"msg"</code> : <code>"getOwner"</code> , <code>"applianceserial"</code> : <code>"123456789012"</code> }		

Server akan mengirimkan format balasan sebagai berikut pada Tabel 3.2.

Tabel 3.2 Tabel Format Balasan `getOwner` dari Server [30]

Name	Value	Keterangan
<code>msg</code>	<code>owner</code>	Identifikasi pesan.
<code>owner</code>	<code><username_pemilik> / NOTFOUND</code>	Berisi <code>username</code> pemilik perangkat pintar tersebut. Mengembalikan <code>NOTFOUND</code> jika tidak ditemukan.
Contoh: { <code>"msg"</code> : <code>"owner"</code> , <code>"owner"</code> : <code>"user1"</code> }		

Balasan dari server akan bernilai *“NOTFOUND”* bila perangkat pintar belum memiliki pemilik. Sedangkan, bila sudah memiliki pemilik, akan menghasilkan nilai berupa *username* pemilik yang akan dipakai untuk melakukan komunikasi pada topik */sdf/<username>*.

B. Menyatakan Status Perangkat Tidak Aktif

Status perangkat dibutuhkan untuk memastikan segala perintah dapat dilaksanakan dengan baik. Bila status perangkat sedang tidak aktif, aplikasi Android tidak bisa memberikan perintah apapun kepada perangkat. Segala kegiatan harus dilakukan saat perangkat sedang aktif. Server akan mengubah status perangkat menjadi aktif secara otomatis bila perangkat pintar mengirimkan pesan *getOwner*.

Pesan yang diberikan untuk menyatakan status tidak aktif adalah *offline*. Topik yang dipakai untuk pesan ini adalah */sdf/<username>*. Berikut format pengiriman yang dikirimkan pada Tabel 3.3.

Tabel 3. 3 Tabel Format Pengiriman Status Offline [30]

Name	Value	Keterangan
msg	offline	Identifikasi pesan.
applianceserial	<applianceserial>	Nomor serial perangkat pintar yang didapat dari alamat MAC perangkat pintar.
Contoh:		
<pre>{ "msg" : "offline", "applianceserial" : "123456789012" }</pre>		

C. Mengambil jadwal pemberian makan dari database

Untuk mengambil jadwal pemberian makan, *Smart Dog Feeder* akan mengirimkan sebuah JSON *object* ke server dengan topik */sdf/<username>*.

Format perintah yang dikirimkan adalah sebagai berikut pada Tabel 3.4.

Tabel 3. 4 Tabel Format Pengiriman *getFeedSchedule* [30]

Name	Value	Keterangan
msg	getFeedSchedule	Identifikasi pesan.
owner	<username>	Username pemilik perangkat pintar.
applianceSerial	<applianceSerial>	Nomor serial perangkat pintar yang didapat dari alamat MAC perangkat pintar.
Contoh: { "msg" : "getFeedSchedule", "owner" : "user1", "applianceSerial" : "789012345612" }		

Kemudian server akan mengolah perintah yang dikirimkan oleh *Smart Dog Feeder* dan mengirimkan dua balasan. Format balasan dari server adalah sebagai berikut pada Tabel 3.5 dan Tabel 3.6.

Tabel 3. 5 Tabel Format Balasan Server untuk Pesan Feed [30]

Name	Value	Keterangan
msg	Feed	Identifikasi pesan.
owner	<username>	Username yang telah terdaftar.
applianceSerial	<applianceSerial>	Nomor serial perangkat pintar yang didapat dari alamat MAC perangkat pintar.
amount	<jumlah_makanan> / -1	Jumlah makanan yang diberikan (dalam gram).

times	<array_jadwal> / -1	Array berisi jadwal pemberian makanan.
timeout	<timeout>	Periode pengecekan apakah makanan sudah dikonsumsi atau belum (dalam menit).
Contoh:		
<pre>{ "msg" : "feed", "owner" : "user1", "applianceserial" : "789012345612", "amount" : 200, "times" : ["07.00","12.00","19.00"], "timeout" : 15 }</pre>		

Tabel 3. 6 Tabel Format Balasan cmdid dari Server [30]

Name	Value	Keterangan
msg	cmdid	Identifikasi pesan.
applianceserial	<applianceserial>	Nomor serial perangkat pintar yang didapat dari alamat MAC perangkat pintar.
cmdid	<id_perintah> / -1	Berisi id perintah yang dikirimkan, didapat dari <i>auto-increment</i> pada basis data.
Contoh:		
<pre>{ "msg" : "cmdid", "applianceserial" : "789012345612", "cmdid" : 3 }</pre>		

D. Memberikan Notifikasi Jadwal Telah Diatur

Perintah yang dipakai untuk mengabarkan bahwa jadwal telah diatur adalah *feedScheduled*. Pesan ini akan dikirimkan melalui topik */sdf/<username>*. Format pengiriman adalah sebagai berikut pada Tabel 3.7.

Tabel 3. 7 Tabel Format Pengiriman *feedScheduled* [30]

Name	Value	Keterangan
msg	feedScheduled	Identifikasi pesan.
cmdid	<id_perintah>	Id perintah yang didapat pada saat menerima perintah.
ctimestamp	<timestamp_perintah_diterima>	Waktu perintah penjadwalan diterima berdasarkan RTC pada perangkat pintar.
Contoh:	<pre>{ "msg" : "feedScheduled", "cmdid" : 2, "ctimestamp" : "2016-04-15 10:00:00" }</pre>	

E. Memberikan Notifikasi Pemberian Makan

Perintah *feedReport* digunakan untuk memberi informasi ke server dan *smartphone* Android mengenai kegiatan pemberian makan pada waktu tersebut. Informasi yang dilaporkan terkait makanan sudah disajikan atau belum, sudah dimakan atau belum bila telah disajikan, dan kehadiran anjing pada waktu pemberian makan.

Pesan yang dikirimkan apabila anjing tidak datang pada waktu pemberian makan adalah *"NODOG"*. Server yang menerima pesan tersebut akan

mengirimkan informasi tersebut ke pengguna dengan menyampaikannya ke aplikasi Android. Sedangkan untuk laporan bahwa alat telah menyajikan makan pada waktu yang ditentukan adalah “SERVED”. Untuk menginformasikan bahwa makanan telah habis dimakan selama waktu tunggu yang ditentukan oleh pengguna, *Smart Dog Feeder* akan mengirimkan pesan “EATEN”. Jika makanan masih tersisa, pesan yang dikirimkan adalah “LEFT”. Topik yang dipakai saat memberikan *feedReport* adalah */sdf/<username>*. Berikut adalah format lengkap pengiriman laporan pada Tabel 3.8.

Tabel 3. 8 Tabel Format Pengiriman Laporan feedReport [30]

Name	Value	Keterangan
msg	feedReport	Identifikasi pesan.
rtimestamp	<timestamp_laporan_kondisi_makanan>	Waktu pelaporan kondisi makanan saat jadwal makan tiba.
appliance serial	<appliance serial>	Nomor serial perangkat pintar yang didapat dari alamat MAC perangkat pintar.
rppt	SERVED / EATEN / LEFT / NODOG	Berisi laporan apakah makanan telah disajikan, dimakan, atau tidak dimakan.
cmdid	<id_perintah>	Berisi id perintah penjadwalan yang telah diselesaikan.
Contoh:		
<pre>{ "msg" : "feedReport", "rtimestamp" : "2016-04-15 19:00:00", "appliance serial" : "789012345612", "rppt" : "EATEN", "cmdid" : 2 }</pre>		

F. Mengambil Informasi Stok Makanan

Perintah `getFeederStock` digunakan untuk mendapatkan informasi stok makanan yang tersimpan dalam server. Perintah akan dikirimkan pada topik `/sdf/<username>`. Berikut format pengiriman yang dapat dilihat pada Tabel 3.9.

Tabel 3. 9 Tabel Format Pengiriman `getFeederStock` [30]

Name	Value	Keterangan
msg	<code>getFeederStock</code>	Identifikasi pesan.
applianceserial	<code><applianceserial></code>	Nomor serial perangkat pintar yang didapat dari alamat MAC perangkat pintar.
Contoh: { "msg" : "getFeederStock", "applianceserial" : "789012345612" }		

Server akan membalas dengan format sebagai berikut pada Tabel 3.10.

Tabel 3. 10 Tabel Format Balasan Pesan `getFeederStock` dari Server [30]

Name	Value	Keterangan
msg	<code>sdfStock</code>	Identifikasi pesan.
applianceserial	<code><applianceserial></code>	Nomor serial perangkat pintar yang didapat dari alamat MAC perangkat pintar.
dogfood	<code><jumlah_makanan_anjing></code> <code>/ -1</code>	Berisi jumlah makanan anjing saat ini (dalam gram).
Contoh: { "msg" : "sdfStock", "applianceserial" : "789012345612", "dogfood" : 1000 }		

G. Memberikan Konfirmasi Perubahan Stok

Perintah *confirmFeederStock* digunakan untuk memberikan persetujuan perubahan stok yang diberikan oleh aplikasi Android. Hal ini diperlukan agar perubahan yang dilakukan pada aplikasi sesuai dengan keadaan yang terbaca pada *Smart Dog Feeder*, yakni keadaan tutup terbuka (yang menandakan bahwa pengguna sedang melakukan pengisian). Perintah akan dikirimkan pada topik */sdf/<username>*.

Bila pada saat menerima pesan adanya perubahan stok keadaan tutup terbaca terbuka, *Smart Dog Feeder* akan mengirimkan pesan konfirmasi dengan atribut *confirmation* “OK”. Sedangkan, bila keadaan tutup tertutup, pesan konfirmasi akan berisi “NOK”. Berikut format pengiriman konfirmasi ke server pada Tabel 3.11.

3.4.2 Komunikasi Perangkat Android ke Server

Komunikasi perangkat Android dengan server merupakan komunikasi antara aplikasi Appliance Hub dengan server. Dalam komunikasi ini, terdapat beberapa format komunikasi yang dipakai dalam format JSON. Baik server ataupun aplikasi Android akan mengambil isi dari atribut *msg* untuk diolah sesuai dengan fungsinya. Berikut adalah format komunikasi yang digunakan.

Tabel 3. 11 Tabel Format Pengiriman *confirmFeederStock* [30]

Name	Value	Keterangan
msg	confirmFeederStock	Identifikasi pesan.
applianceSerial	<applianceSerial>	Nomor serial perangkat pintar yang didapat dari alamat MAC perangkat pintar.
confirmation	OK / NOK	Konfirmasi update stok berdasarkan dibuka atau tidaknya tutup tempat makanan anjing.
dogfood	<jumlah_makanan_anjing>	Berisi jumlah makanan anjing saat ini (dalam gram).
<p>Contoh:</p> <pre>{ "msg" : "updateFeederStock", "applianceSerial" : "789012345612", "confirmation" : "OK", "dogfood" : 1000 }</pre>		

A. Mendaftarkan Pengguna Baru

Perintah *signup* digunakan untuk mendaftarkan pengguna baru pada aplikasi Appliance Hub. Pengguna akan memasukkan data-data yang diperlukan dan dikemas dalam format JSON. Pesan ini akan dikirimkan pada topik */signup/<username>*. *Username* akan didapat dari hasil masukan data dari pengguna pada halaman *sign up*. Berikut format pengiriman yang dapat dilihat pada Tabel 3.12.

Tabel 3. 12 Tabel Format Pengiriman Perintah Sign Up [30]

Name	Value	Keterangan
msg	signup	Identifikasi pesan.
username	<username>	Username baru yang akan didaftarkan.
password	<password>	Password dari username baru yang akan didaftarkan dan akan di- <i>hashing</i> SHA2-256.
fullname	<fullname>	Nama lengkap pengguna yang akan didaftarkan.
email	<email>	Email pengguna yang akan didaftarkan.
phone	<phone_number>	Nomor telepon pengguna yang akan didaftarkan.
Contoh:		
<pre>{ "msg" : "signup", "username" : "user1", "password" : "0a041b9462caa4a31bac3567e0b6e6fd9100787db2ab433d96f6d178cabfce90", "fullname" : "User One", "email" : "user1@domain.com", "phone" : "081818181818" }</pre>		

Server akan memberikan balasan seperti yang dapat dilihat pada Tabel 3.13.

Tabel 3. 13 Tabel Format Balasan Pesan Sign Up dari Server [30]

Name	Value	Keterangan
msg	signupStatus	Identifikasi pesan.
status	SUCCESS / FAIL	Menyatakan apakah pengguna berhasil didaftarkan atau tidak.
Contoh:		
<pre>{ "msg" : "signupStatus", "status" : "SUCCESS" }</pre>		

B. Mendapatkan Perangkat Pintar

Untuk mendapatkan perangkat pintar yang dimiliki, aplikasi akan mengirimkan pesan *getMyAppliance*. Pesan ini dikirimkan melalui topik */sdf/<username>*. Berikut format pengiriman pesan yang dapat dilihat pada Tabel 3.14.

Tabel 3. 14 Tabel Format Pengiriman *getMyAppliance* [30]

Name	Value	Keterangan
msg	getMyAppliance	Identifikasi pesan.
owner	<username_pemilik>	Berisi username pemilik yang menginginkan daftar perangkat yang dimilikinya.
Contoh:		
<pre>{ "msg" : "getMyAppliance", "owner" : "user1" }</pre>		

Server akan mengirimkan balasan seperti pada Tabel 3.15.

Tabel 3. 15 Tabel Format Balasan *getMyAppliance* dari Server [30]

Name	Value	Keterangan
msg	myAppliance	Identifikasi pesan.
dat	<array_appliance_detail> / NOTFOUND	Berisi array JSON object yang terdiri atas applianceserial, applianceid, nickname, status. Mengembalikan NOTFOUND jika tidak ditemukan.
applianceserial	<applianceserial>	Nomor serial perangkat pintar yang didapat dari alamat MAC perangkat pintar.
applianceid	<applianceid>	Jenis perangkat pintar.
nickname	<nickname>	Nama perangkat yang diberikan oleh pemilik.
status	ACTIVE / INACTIVE	Status perangkat, apakah

		aktif/tidak.
<p>Contoh:</p> <pre>{ "msg" : "myAppliance" "dat" : [{ "applianceserial" : "123456789012", "applianceid" : "rck", "nickname" : "Kitchen Rice Cooker", "status" : "INACTIVE" }, { "applianceserial" : "210987654321", "applianceid" : "smd", "nickname" : "Grandma's Pill Dispenser", "status" : "ACTIVE" }, { "applianceserial" : "789012345612", "applianceid" : "sdf", "nickname" : "Brownie's Feeder", "status" : "ACTIVE" }] }</pre>		

C. Mendaftarkan Perangkat Pintar

Perintah yang digunakan untuk mendaftarkan perangkat pintar adalah *registerMyAppliance*. Dengan perintah ini, sebuah perangkat akan terdaftar telah memiliki pemilik dan dapat diakses melalui aplikasi Android. Perintah ini akan dikirimkan melalui topik */general/<username>*. Berikut format pengiriman yang dapat dilihat pada Tabel 3.16.

Tabel 3. 16 Tabel Format Pengiriman Pesan *registerMyAppliance* [30]

Name	Value	Keterangan
msg	registerMyAppliance	Identifikasi pesan.
owner	<username_pemilik>	Berisi username pemilik perangkat pintar tersebut.
applianceserial	<applianceserial>	Nomor serial perangkat pintar yang didapat dari alamat MAC perangkat pintar.
Contoh: <pre>{ "msg" : "registerMyAppliance", "owner" : "user1", "applianceserial" : "123456789012" }</pre>		

Server akan mengirimkan balasan dengan format yang dapat dilihat pada Tabel 3.17.

Tabel 3. 17 Tabel Format Balasan Pesan *registerMyAppliance* dari Server [30]

Name	Value	Keterangan
msg	registrationStatus	Identifikasi pesan.
status	SUCCESS / FAIL	Menyatakan apakah pemilik perangkat berhasil didaftarkan.
Contoh: <pre>{ "msg" : "registrationStatus", "status" : "SUCCESS" }</pre>		

D. Menghapus Perangkat

Perintah *unregisterMyAppliance* dapat dipakai bila pengguna ingin menghapus perangkat dari *database*. Dengan memberikan perintah ini, seluruh data yang berkaitan dengan perangkat tersebut akan hilang. Pesan ini dikirimkan

melalui topik `/general/<username>`. Berikut format lengkap pengiriman yang dapat dilihat pada Tabel 3.18.

Tabel 3. 18 Tabel Format Pengiriman Pesan unregisterAppliance [30]

Name	Value	Keterangan
msg	unregisterMyAppliance	Identifikasi pesan.
owner	<username_pemilik>	Berisi username pemilik perangkat pintar tersebut.
password	<password>	Password dari username baru yang akan didaftarkan dan akan di- <i>hashing</i> SHA2-256.
applianceserial	<applianceserial>	Nomor serial perangkat pintar yang didapat dari alamat MAC perangkat pintar.
Contoh:		
<pre>{ "msg" : "unregisterMyAppliance", "owner" : "user1", "applianceserial" : "123456789012" }</pre>		

Server akan memberikan balasan dengan format yang dapat dilihat pada Tabel 3.19.

Tabel 3. 19 Tabel Format Balasan Pesan unregisterMyAppliance dari Server [30]

Name	Value	Keterangan
msg	unregisterStatus	Identifikasi pesan.
status	SUCCESS / FAIL	Menyatakan apakah pemilik perangkat berhasil didaftarkan.
Contoh:		
<pre>{ "msg" : "unregisterStatus", "status" : "SUCCESS" }</pre>		

E. Mengambil Profil Pengguna

Perintah *getProfile* digunakan untuk mengambil informasi terkait profil pengguna. Pesan ini dikirimkan melalui topik */general/<username>*. Berikut format pengiriman yang dapat dilihat pada Tabel 3.20.

Tabel 3. 20 Tabel Format Pengiriman Pesan *getProfile* [30]

Name	Value	Keterangan
msg	getProfile	Identifikasi pesan.
username	<username>	Username yang telah terdaftar.
password	<password>	Password yang di- <i>hashing</i> SHA2-256.
Contoh:		
<pre>{ "msg" : "getProfile", "username" : "user1", "password" : "0a041b9462caa4a31bac3567e0b6e6fd9100787db2ab433d96f6d178cabfce90" }</pre>		

Server kemudian akan mengirimkan balasan dengan format seperti pada Tabel 3.21.

Tabel 3. 21 Tabel Format Balasan Pesan *myProfile* dari Server [30]

Name	Value	Keterangan
msg	myProfile	Identifikasi pesan.
username	<username> / NOTFOUND	Username yang telah terdaftar. Mengembalikan nilai NOTFOUND jika tidak ditemukan.
fullname	<fullname>	Nama lengkap pengguna yang terdaftar.
email	<email>	Email baru pengguna.
phone	<phone_number>	Nomor telepon pengguna yang baru.

Contoh:

```
{
  "msg"      : "myProfile",
  "username" : "user1",
  "fullname" : "User One",
  "email"    : "user1@domain.com",
  "phone"    : "081616161616"
}
```

F. Memperbarui Profil Pengguna

Perintah *updateProfile* digunakan untuk membarui profil pengguna selain *username* dan *password*. Pesan ini dikirimkan melalui topik */general/<username>*. Berikut format pengiriman yang dapat dilihat pada Tabel 3.22.

Tabel 3. 22 Tabel Format Pengiriman Pesan *updateProfile* [30]

Name	Value	Keterangan
msg	updateProfile	Identifikasi pesan.
username	<username>	Username yang telah terdaftar.
password	<password>	Password yang di- <i>hashing</i> SHA2-256.
fullname	<fullname>	Nama lengkap pengguna yang terdaftar.
email	<email>	Email baru pengguna.
phone	<phone_number>	Nomor telepon pengguna yang baru.

Contoh:

```
{
  "msg"      : "updateProfile",
  "username" : "user1",
  "password" :
  "0a041b9462caa4a31bac3567e0b6e6fd9100787db2ab433d96f6d178cabfce90",
  "fullname" : "User One",
  "email"    : "user1@domain.com",
}
```

```

“phone”      : “081616161616”
}

```

Server akan membalas dengan format pada Tabel 3.23.

Tabel 3. 23 Tabel Format Balasan Pesan updateProfileStatus dari Server [30]

Name	Value	Keterangan
msg	updateProfileStatus	Identifikasi pesan.
status	SUCCESS / FAIL	Menyatakan apakah profil pengguna berhasil diperbarui atau tidak.
Contoh:		
<pre> { “msg” : “updateProfileStatus”, “status” : “SUCCESS” } </pre>		

G. Memperbarui Password Pengguna

Untuk mengganti *password*, aplikasi akan mengirimkan pesan *updatePassword* melalui topik */general/<username>*. Format pengiriman pesan secara lengkap dapat dilihat pada Tabel 2.24.

Tabel 3. 24 Tabel Format Pengiriman Pesan updatePassword dari Server [30]

Name	Value	Keterangan
msg	updatePassword	Identifikasi pesan.
username	<username>	Username yang telah terdaftar.
oldpassword	<old_password>	Password lama yang di- <i>hashing</i> SHA2-256.
newpassword	<new_password>	Password baru yang di- <i>hashing</i> SHA2-256.
Contoh:		
<pre> { “msg” : “updatePassword”, “username” : “user1”, </pre>		

```

“oldpassword”:
“0a041b9462caa4a31bac3567e0b6e6fd9100787db2ab433d96f6d178cabfce90”,
“newpassword”      :
“0aa0s97fg8y0k09ausd0ahsa0jdj0safyad807fd9as0d8as9d8a0s09d7fhgdfh99d”
}

```

Server akan mengirimkan balasan dengan format seperti pada Tabel 2.25.

Tabel 3. 25 Tabel Format Balasan Pesan *updatePasswordStatus* dari Server [30]

Name	Value	Keterangan
msg	updatePasswordStatus	Identifikasi pesan.
status	SUCCESS / FAIL	Menyatakan apakah password pengguna berhasil diperbarui atau tidak.
Contoh:		
<pre> { “msg” : “updatePasswordStatus”, “status” : “SUCCESS” } </pre>		

H. Merubah Nama Panggilan Perangkat

Perangkat-perangkat pintar dalam sistem Appiance Hub dapat diberi nama panggilan (*nickname*) sesuai keinginan pemilik. Perintah yang dipakai untuk merubah nama tersebut adalah *renameAppliance*. Pesan ini akan dikirim melalui topik */general/<username>*. Berikut format pengiriman yang dapat dilihat pada Tabel 3.26.

Tabel 3. 26 Tabel Format Pengiriman Pesan *renameAppliance* [30]

Name	Value	Keterangan
msg	renameAppliance	Identifikasi pesan.
applianceserial	<applianceserial>	Nomor serial perangkat pintar yang didapat dari alamat MAC perangkat pintar.
owner	<owner>	Berisi username pemilik

		perangkat pintar tersebut.
nickname	<nickname>	Berisi nama perangkat pintar
Contoh:		
<pre>{ "msg" : "renameAppliance", "applianceSerial" : "123456789012", "owner" : "user1", "nickname" : "Smart Rice Cooker" }</pre>		

Server akan membalas dengan format seperti pada Tabel 3.27.

Tabel 3. 27 Tabel Format Balasan Pesan *renameStatus* dari Server [30]

Name	Value	Keterangan
msg	renameStatus	Identifikasi pesan.
status	SUCCESS / FAIL	Menyatakan apakah nama perangkat pengguna berhasil diperbarui atau tidak.
Contoh:		
<pre>{ "msg" : "renameAppliance", "status" : "SUCCESS" }</pre>		

I. Mendapatkan Status Perangkat

Perintah *getStatus* digunakan untuk mendapatkan informasi perangkat yang dikirimkan melalui topik */general/username*. Pesan ini menghasilkan nilai status aktif dan tidak aktif. Berikut format pengiriman yang dapat dilihat pada Tabel 3.28.

Tabel 3. 28 Tabel Format Pengiriman Pesan *getStatus* [30]

Name	Value	Keterangan
msg	getStatus	Identifikasi pesan.
applianceSerial	<applianceSerial>	Nomor serial perangkat pintar yang didapat dari alamat MAC perangkat pintar.
owner	<owner>	Berisi username pemilik perangkat pintar tersebut.
Contoh:		
<pre>{ "msg" : "getStatus", "applianceSerial" : "123456789012", "owner" : "user1" }</pre>		

Server akan membalas dengan format seperti pada Tabel 2.29.

Tabel 3. 29 Tabel Format Balasan Pesan *applianceStatus* dari Server [30]

Name	Value	Keterangan
msg	applianceStatus	Identifikasi pesan.
status	ACTIVE / INACTIVE / NOTFOUND	Berisi status perangkat pintar. Mengembalikan NOTFOUND jika tidak ditemukan.
Contoh:		
<pre>{ "msg" : "applianceStatus", "status" : "ACTIVE" }</pre>		

J. Mendapatkan Log

Perintah *getLog* digunakan untuk mendapatkan catatan perintah dari aplikasi Android dan perangkat pintar. Pesan ini dikirimkan melalui topik */general/<username>*. Berikut format pengiriman yang dapat dilihat pada Tabel 3.30.

Tabel 3. 30 Tabel Format Pengiriman Pesan getLog [30]

Name	Value	Keterangan
msg	getLog	Identifikasi pesan.
owner	<username>	Username yang telah terdaftar.
applianceserial	<applianceserial>	Nomor serial perangkat pintar yang didapat dari alamat MAC perangkat pintar.
Contoh:		
<pre>{ "msg" : "getLog", "owner" : "user1", "applianceserial" : "123456789012" }</pre>		

Server akan membalas dengan format seperti pada Tabel 3.31.

Tabel 3. 31 Tabel Format Balasan Pesan reportLog dari Server [30]

Name	Value	Keterangan
msg	reportLog	Identifikasi pesan.
dat	<array_appliance_detail> / NOTFOUND	Berisi array JSON object yang terdiri atas ctimestamp, cmd, rtimestamp, rppt. Mengembalikan NOTFOUND jika tidak ditemukan.
rtimestamp	<timestamp_report>	Berisi timestamp laporan diterima.
rppt	<laporan>	Berisi string JSON yang berisi laporan.
Contoh:		
<pre>{ "msg" : "reportLog", "dat" : [{ "rtimestamp" : "2016-04-15 15:15:00", "rppt" : "Cook rice finished" }, { "rtimestamp" : "2016-04-16 15:15:00", </pre>		

```

    "rprt"      : "Cook porridge finished"
  }
]
}

```

K. Memberikan Jadwal Pemberian Makan

Perintah yang digunakan untuk memberikan jadwal pemberian makan ke server adalah *feed*. Pesan ini dikirimkan melalui topik */sdf/<username>*. Berikut format pengiriman yang dapat dilihat pada Tabel 3.32.

Tabel 3. 32 Tabel Format Pengiriman Pesan Feed [30]

Name	Value	Keterangan
msg	Feed	Identifikasi pesan.
owner	<username>	Username yang telah terdaftar.
applianceserial	<applianceserial>	Nomor serial perangkat pintar yang didapat dari alamat MAC perangkat pintar.
amount	<jumlah_makanan>	Jumlah makanan yang diberikan (dalam gram).
times	<array_jadwal>	Array berisi jadwal pemberian makanan.
timeout	<timeout>	Periode pengecekan apakah makanan sudah dikonsumsi atau belum (dalam menit).
Contoh:		
<pre> { "msg" : "feed", "owner" : "user1", "applianceserial" : "789012345612", "amount" : 200, "times" : ["07:00","12:00","19:00"], "timeout" : 15 } </pre>		

Server kemudian akan memberi balasan dengan format seperti pada Tabel 3.33.

Tabel 3. 33 Tabel Format Balasan Pesan feed dari Server [30]

Name	Value	Keterangan
msg	Cmdid	Identifikasi pesan.
applianceserial	<applianceserial>	Nomor serial perangkat pintar yang didapat dari alamat MAC perangkat pintar.
cmdid	<id_perintah> / -1	Berisi id perintah yang dikirimkan, didapat dari <i>auto-increment</i> pada basis data.
Contoh:		
<pre>{ "msg" : "cmdid", "applianceserial" : "789012345612", "cmdid" : 3 }</pre>		

L. Memperbarui Stok Makanan

Perintah *updateFeederStock* digunakan untuk memperbarui stok makanan. Perintah ini dikirim melalui topik */sdf/<username>* dan kemudian dikonfirmasi oleh *Smart Dog Feeder*. Berikut format pengiriman yang dapat dilihat pada Tabel 3.34.

Tabel 3. 34 Tabel Format Pengiriman Pesan *updateFeederStock* [30]

Name	Value	Keterangan
msg	updateFeederStock	Identifikasi pesan.
applianceserial	<applianceserial>	Nomor serial perangkat pintar yang didapat dari alamat MAC perangkat pintar.
dogfood	<jumlah_makanan_anjing>	Berisi jumlah makanan anjing saat ini (dalam gram).

Contoh:

```
{
  "msg"          : "updateFeederStock",
  "applianceSerial" : "789012345612",
  "dogfood"      : 1000
}
```

M. Mengambil Informasi Stok Makanan

Seperti pada perintah di *Smart Dog Feeder*, perintah *getFeederStock* digunakan untuk mendapatkan informasi stok makanan yang tersimpan dalam server untuk ditampilkan pada aplikasi Android. Perintah akan dikirimkan pada topik */sdf/<username>*. Berikut format pengiriman yang dapat dilihat pada Tabel 3.35.

Tabel 3. 35 Tabel Format Pengiriman getFeederStock [30]

Name	Value	Keterangan
msg	getFeederStock	Identifikasi pesan.
applianceSerial	<applianceSerial>	Nomor serial perangkat pintar yang didapat dari alamat MAC perangkat pintar.
Contoh:		
<pre>{ "msg" : "getFeederStock", "applianceSerial" : "789012345612" }</pre>		

Server akan membalas dengan format sebagai berikut pada Tabel 3.35.

Tabel 3. 36 Tabel Format Balasan Pesan getFeederStock dari Server [30]

Name	Value	Keterangan
msg	sdfStock	Identifikasi pesan.
applianceSerial	<applianceSerial>	Nomor serial perangkat pintar yang didapat dari alamat MAC perangkat pintar.

dogfood	<jumlah_makanan_anjing> / -1	Berisi jumlah makanan anjing saat ini (dalam gram).
Contoh:		
<pre>{ "msg" : "sdfStock", "applianceSerial" : "789012345612", "dogfood" : 1000 }</pre>		

3.5 Perancangan Perangkat Lunak Smart Dog Feeder

Program dibuat menggunakan bahasa C. Komunikasi *serial* dengan modul WiFi ESP8266 menggunakan *baud rate* 9600 bps. Pada alur program utama yang dapat dilihat pada Gambar 3.4 terdapat beberapa sub proses, yaitu *set_alarm_1*, *set_alarm_2*, dan *set_next_alarm*. Sub proses *set_alarm_1* dapat dilihat pada Gambar 3.5. Sub proses *set_alarm_2* dapat dilihat pada Gambar 3.6. Sedangkan sub proses *set_next_alarm* dapat dilihat pada Gambar 3.7.

Pada alur program utama, Arduino sebagai mikrokontroler akan membaca hasil yang diterima pada *port serial*. Arduino membaca pesan berformat JSON tersebut dengan membandingkan isi dari atribut *"msg"*. Bila *"msg"* berupa *"feed"*, Arduino akan mengisi nilai dari variabel *"porasi"* yang sebelumnya telah disediakan dengan nilai yang didapat dari *server* kemudian dilanjutkan dengan mengatur alarm pada sub proses *set_alarm_1*. Bila *"msg"* berupa *"updateFeederStock"*, Arduino akan mengecek *magnetic switch* pada tutup perangkat. Nilai 1 untuk keadaan tutup terbuka dan nilai 0 adalah nilai untuk tutup yang tertutup. Bila terbuka, pembaruan stok akan dilakukan dengan merubah nilai pada variabel *"stock"* dengan yang diterima dan mengirim laporan berupa *"OK"* ke *server*. Sedangkan jika tertutup, konfirmasi pembaruan akan

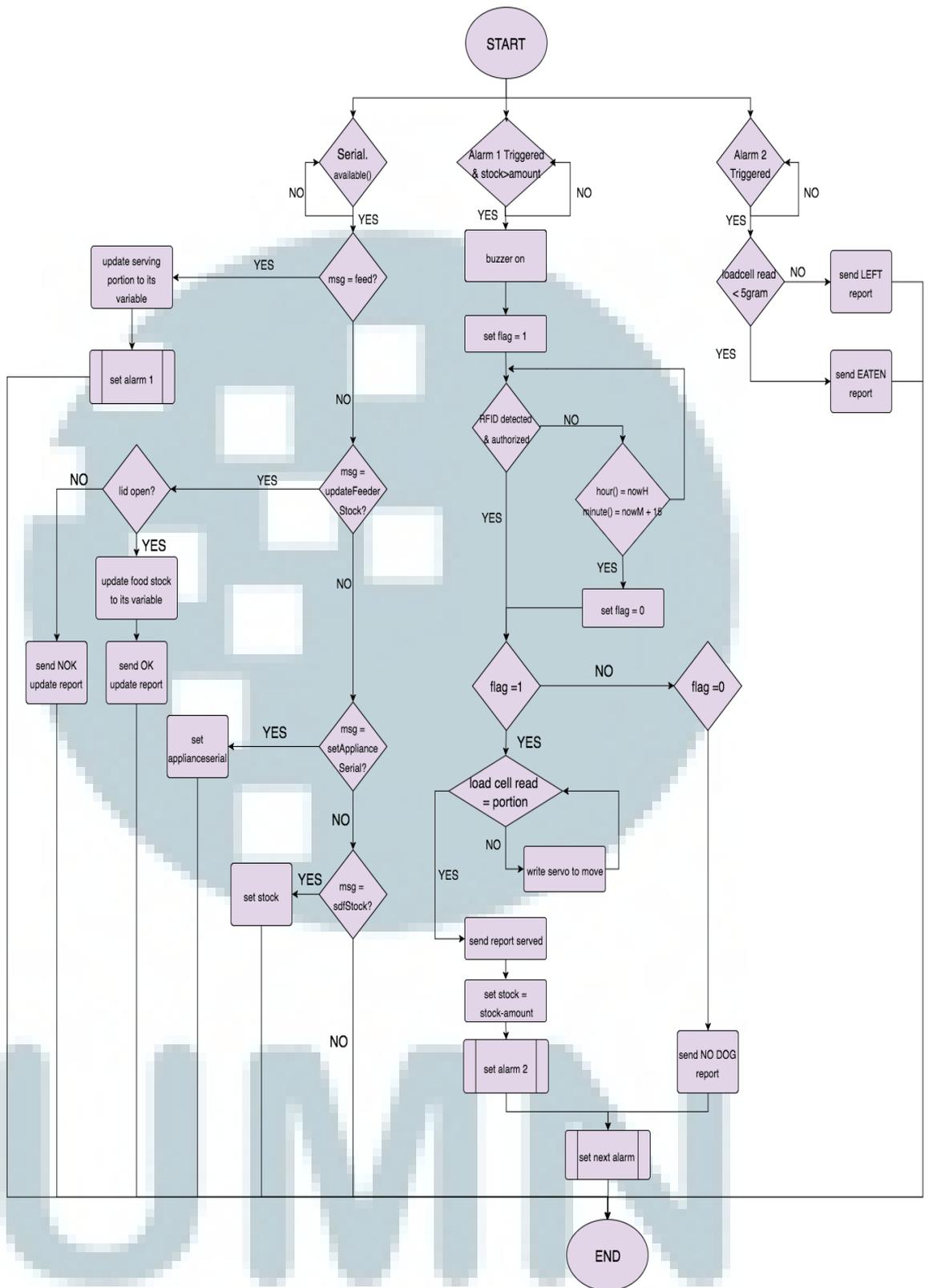
dianggap tidak sah dan mengirimkan laporan berupa “NOK” ke *server*. Bila “*msg*” berupa “*setApplianceSerial*”, Arduino akan mengisi nilai dari variabel “*applianceSerial*” dengan nilai yang didapat dari *server*. Sedangkan bila “*msg*” berupa “*sdfStock*”, Arduino akan mengisi nilai dari variabel “*stock*” dengan nilai yang diberikan oleh *server*.

Pada saat alarm 1 *triggered*, Arduino akan mengecek stok. Bila stok mencukupi untuk memberikan makanan, fungsi berikutnya akan dijalankan. Sedangkan bila tidak, fungsi berikutnya tidak akan dijalankan. Fungsi berikutnya adalah menyalakan *buzzer* dan memberi nilai pada variabel “*flag*” berupa 1. Kemudian Arduino akan membaca kehadiran RFID *tag*. Bila tidak terdapat RFID *tag* setelah 15 menit dari waktu pemberian makan, variabel “*flag*” akan diubah menjadi bernilai 0. Bila RFID *tag* terdeteksi sebelum 15 menit, “*flag*” akan tetap bernilai 1. “*flag*” bernilai satu berarti fungsi penumpahan makanan ke wadah makan anjing akan dilakukan. *Loadcell* akan membaca berat dari wadah, kemudian *servo* akan berputar sampai *loadcell* membaca berat wadah sama dengan porsi yang diinginkan pengguna kemudian berhenti. Namun, jika *loadcell* membaca berat wadah belum sesuai dengan porsi dan *servo* telah berputar 2 kali lipat dari jumlah putaran seharusnya, *servo* akan dimatikan dan dianggap telah memberikan porsi yang sesuai, tetapi makanan telah dimakan selama proses pengeluaran. Setelah penyajian selesai, Arduino akan mengirim laporan berupa “*SERVED*” ke *server* kemudian menjadwalkan alarm 2 pada sub proses *set_alarm_2* dan jadwal pemberian makan setelahnya pada sub proses *set_next_alarm*. Namun, bila “*flag*” bernilai 0, Arduino tidak akan menggerakkan *servo* melainkan langsung mengirim laporan ke *server* berupa

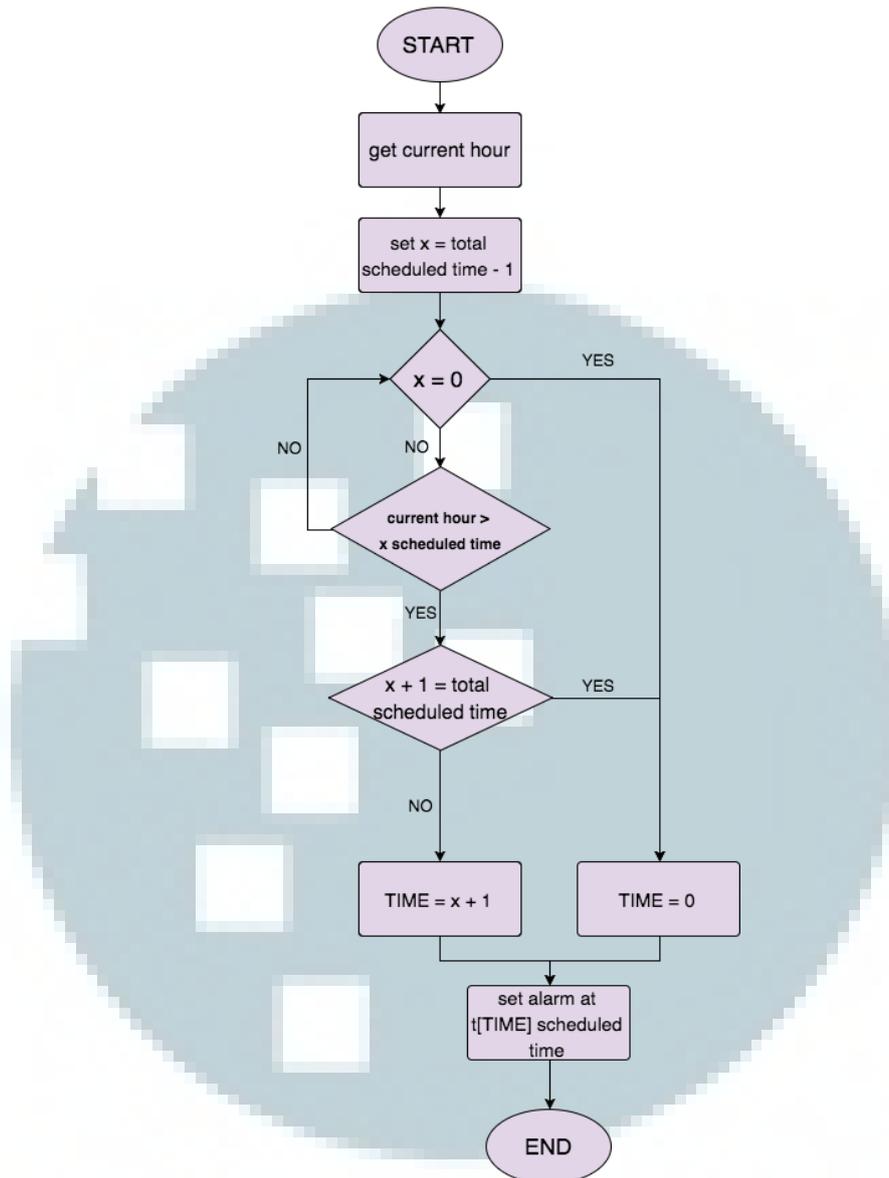
“*NO DOG*” kemudian menjadwalkan alarm 1 untuk pemberian makan berikutnya.

Pada saat alarm 2 *triggered*, *loadcell* akan membaca berat pada wadah saat ini. Bila bernilai 0 sampai dengan 5 gram, yang berarti makanan telah dimakan, Arduino akan mengirimkan laporan berupa “*EATEN*”. Sedangkan bila makanan dibaca masih tersisa, laporan yang dikirimkan adalah “*LEFT*”.



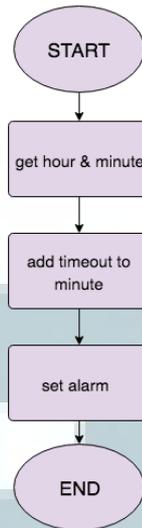


Gambar 3. 4Flowchart Main Program Smart Dog Feeder



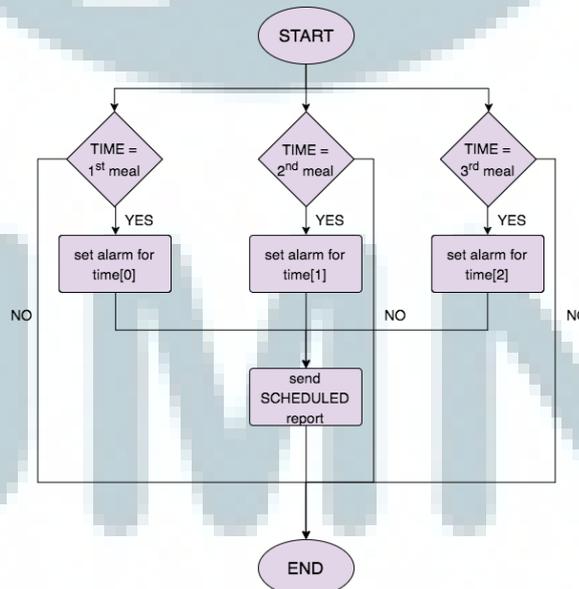
Gambar 3. 5 Flowchart set_alarm_1

Pada sub proses *set_alarm_1*, Arduino akan mengambil waktu pada saat proses dilakukan dari RTC. Kemudian mendeklarasikan sebuah variabel *x* yang berisi jumlah waktu pemberian makan yang dikurangi dengan 1. Bila *x* sama dengan nol (yang berarti jadwal yang diberikan hanya berjumlah satu), variabel global “*TIME*” akan diisi dengan nilai nol. Bila tidak, Arduino akan mengecek jadwal pemberian makan sampai mendapatkan jadwal yang paling mendekati dari waktu saat itu dan memberi nilai variabel “*TIME*” sesuai posisi *array* dari waktu yang diterima dari *server*.



Gambar 3. 6 Flowchart set_alarm_2 Smart Dog Feeder

Sub proses *set_alarm_2* merupakan fungsi untuk mengatur penjadwalan alarm 2 yang berfungsi untuk melakukan pengecekan pada kondisi makanan yang telah disajikan. Pada saat masuk ke dalam sub proses, Arduino akan meminta informasi waktu pada saat tersebut pada RTC kemudian ditambahkan dengan nilai *timeout* yang ditetapkan oleh pengguna melalui aplikasi Appliance Hub.



Gambar 3. 7 Flowchart set_next_alarm Smart Dog Feeder

Sub proses *set_next_alarm* berfungsi untuk melakukan penjadwalan pemberian makanan pada waktu yang telah ditentukan setelah pemberian makanan yang sudah diselesaikan. Pada saat masuk ke dalam sub proses ini, Arduino mengecek nilai variabel global “*TIME*”. Bila bernilai 0, alarm akan dijadwalkan untuk waktu pemberian makan yang pertama (*array 0*). Sedangkan bila bernilai 1, dijadwalkan untuk waktu pemberian makan kedua dan bila bernilai 2, dijadwalkan untuk waktu pemberian ketiga.

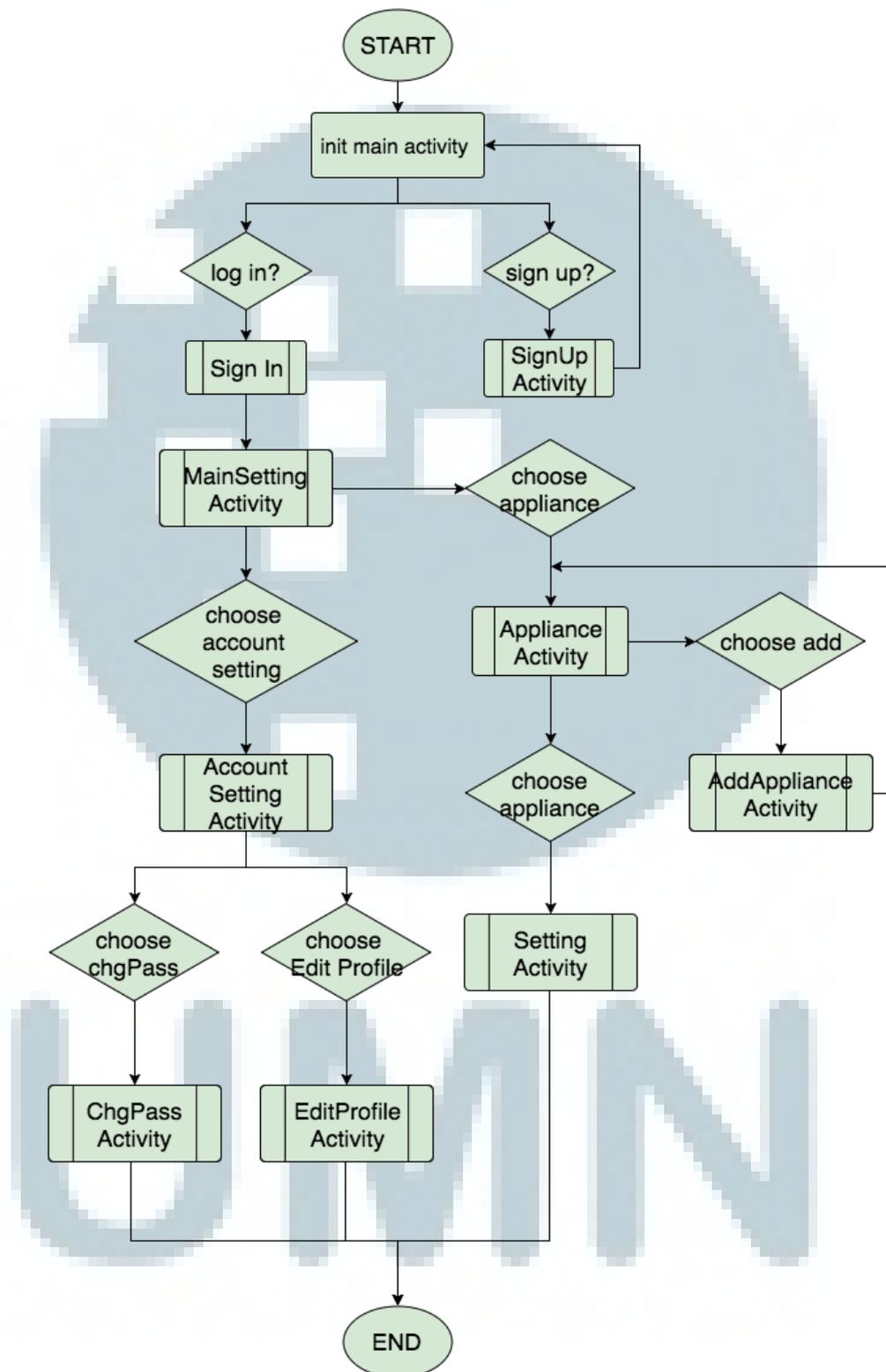
3.6 Perancangan Aplikasi Android

Secara umum aplikasi Appliance Hub terdiri dari 8 *activity*, yaitu *sign in*, *sign up*, *main setting*, *appliance*, *setting*, *account setting*, *change password*, dan *edit profile*. Pada *appliance* terdapat sebuah *activity*, yaitu *add appliance*. Pada *setting* terdapat 5 *activity*, yaitu *feed* dan *feed time* atau *cook menu* dan *cook* atau *pill* dan *sol* (tergantung pada jenis *appliance*), *log*, *update stock*, dan *unregister*. Pada Gambar 3.8 adalah *flowchart* utama dari aplikasi Appliance Hub.

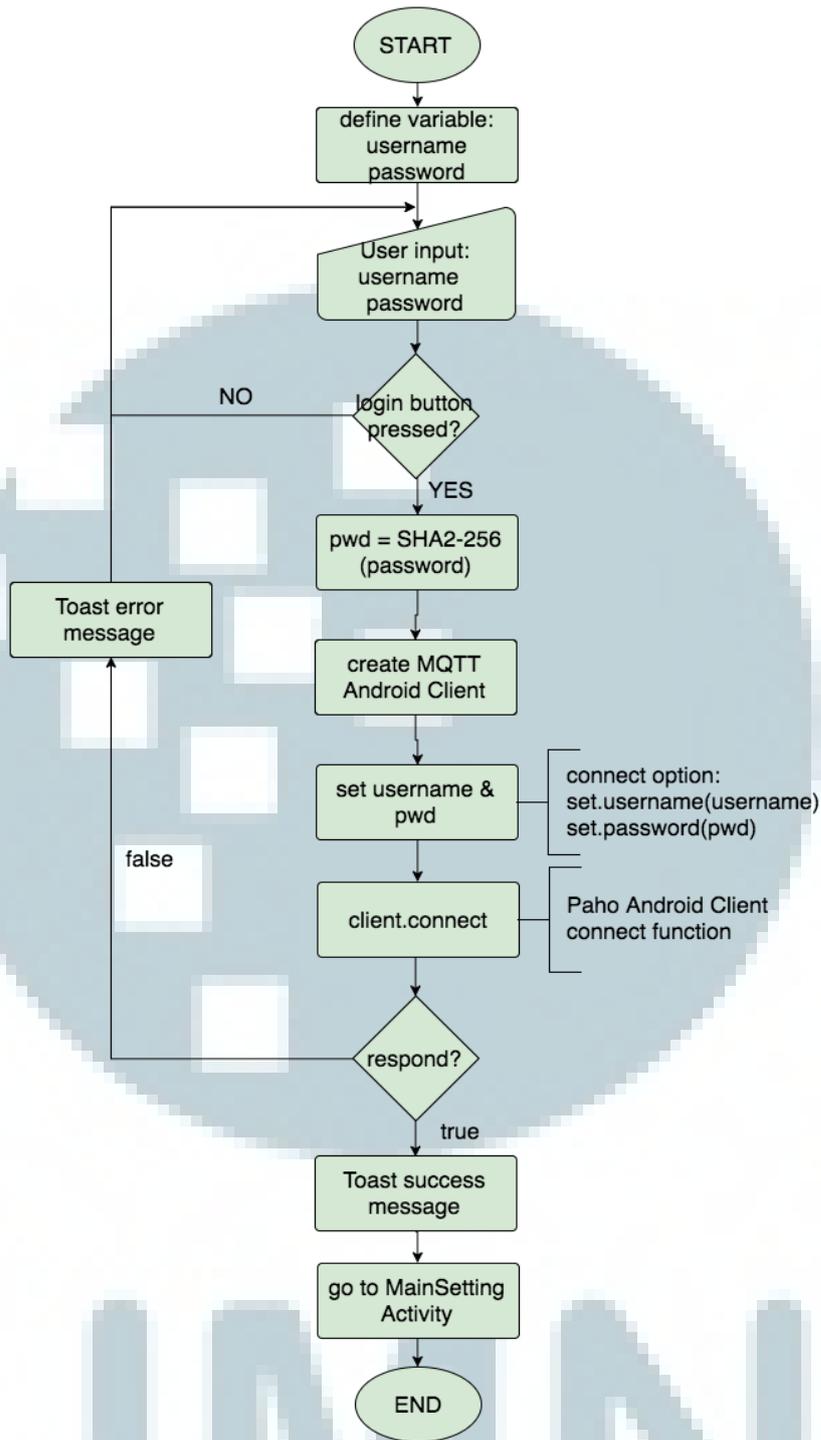
3.6.1 Rancangan Halaman Sign In

Halaman ini bertugas untuk mengambil input *username* dan *password*. Kedua data ini diperlukan untuk koneksi pada MQTT server. Bila tombol *log in* ditekan, aplikasi akan mengirimkan perintah *connect* yang telah dilengkapi dengan *username* dan *password* dari *user*. Dari perintah *connect* tersebut, aplikasi akan mendapat balasan. Apabila data sesuai, maka *user* akan masuk ke halaman *Main Setting*. Selain itu, pada halaman *Sign In*, *user* dapat mendaftarkan diri bila baru pertama kali menggunakan aplikasi Appliance Hub.

Pada halaman ini, objek MQTTAndroidClient dibuat dan dipakai untuk seluruh *Activity*.



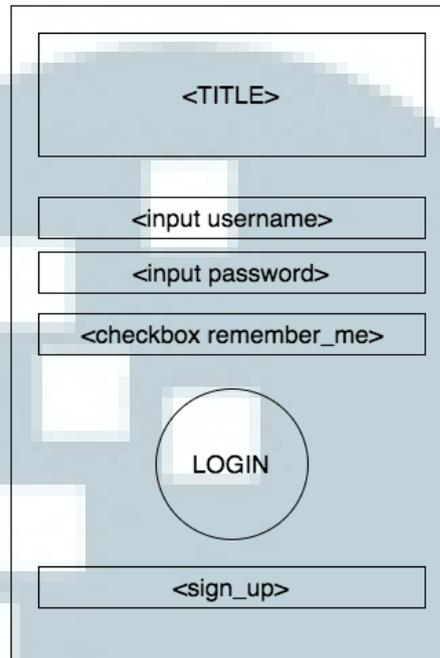
Gambar 3. 8 Flowchart Utama Appliance Hub



Gambar 3. 9 Flowchart Halaman Sign In

Gambar 3.10 menampilkan rancangan halaman awal aplikasi, yaitu halaman *sign in*. Pada kolom <TITLE> akan ditempatkan logo dari aplikasi Appliance Hub. Pengguna akan menuliskan *username* dan *password* pada *EditText* yang ditandai dengan <input username> dan <input password>. *CheckBox* bertuliskan

remember me berfungsi untuk mengingat *username* dan *password* pengguna untuk mempermudah proses *login*. Tombol *login* berupa *ImageButton* dan tulisan *Sign Up* akan dituliskan dalam *TextView*.



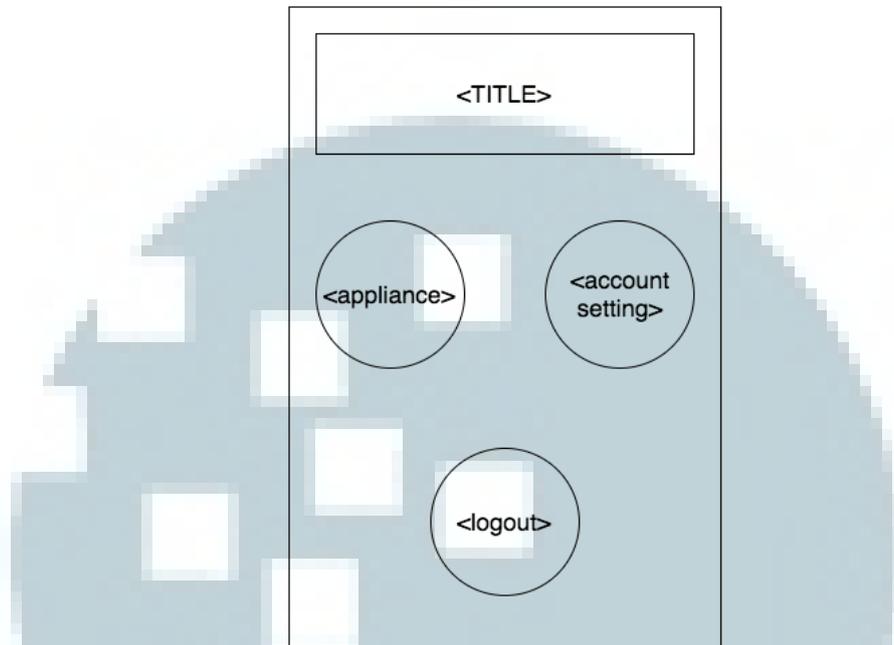
Gambar 3. 10 Rancangan Tampilan Halaman Sign In

3.6.2 Rancangan Halaman Main Setting

Halaman ini berisi 3 buah pilihan, yaitu *my appliance*, *account setting*, dan *logout*. Pilihan *my appliance* akan membawa *user* ke halaman *Appliance* yang berisi daftar perangkat yang dimiliki *user*. Pilihan *account setting* akan membawa *user* ke halaman *Account Setting* yang berfungsi untuk mengubah informasi mengenai *user* tersebut. Sedangkan pilihan *logout* berfungsi untuk kembali ke halaman *Sign In*. Diagram alur program dari halaman ini dapat dilihat pada Lampiran.

Gambar 3.11 menampilkan rancangan halaman pengaturan utama, yaitu *main setting* yang berisi pilihan pengaturan pengguna dan menu perangkat. Pada <TITLE> akan diisi dengan *Image* berupa logo dari aplikasi. Tombol pada

halaman ini, yaitu <appliance>, <account setting>, dan <logout> merupakan *ImageButton*.

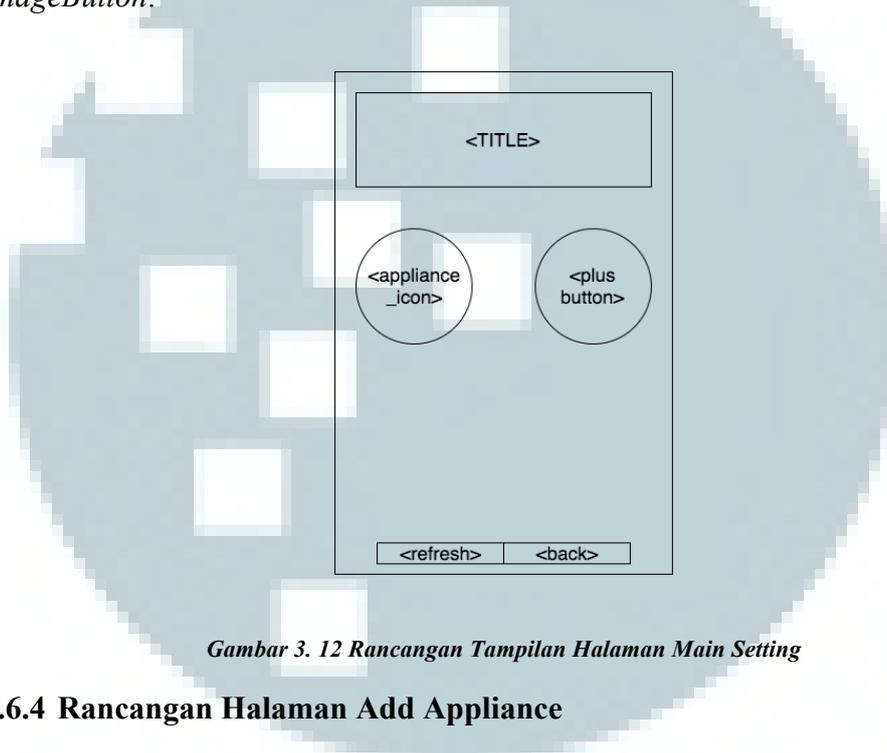


Gambar 3. 11 Rancangan Tampilan Halaman Main Setting

3.6.3 Rancangan Halaman Appliance

Halaman ini menampilkan daftar perangkat yang dimiliki *user*. Pada awal halaman *Appliance* dibuka, aplikasi akan mengirim perintah ke server untuk mendapatkan informasi mengenai perangkat *user*. Gambar perangkat akan berwarna abu-abu bila status perangkat tidak aktif. Bila *user* belum memiliki perangkat, halaman ini hanya akan berisi tombol dengan simbol tambah (+) yang berguna untuk melakukan pendaftaran perangkat baru. Pendaftaran perangkat akan berada di halaman *Add Appliance*. Bila *user* memilih salah satu perangkat yang tertera, akan muncul halaman *Setting*. Selain itu terdapat pilihan *refresh* yang berfungsi untuk mengulang kembali halaman *Appliance* dan pilihan *back* untuk kembali ke halaman *main setting*. Diagram alur untuk halaman ini dapat dilihat pada Lampiran.

Gambar 3.12 menampilkan rancangan halaman *appliance* yang akan berisi perangkat-perangkat yang dimiliki pengguna. Pada bagian <TITLE> merupakan *Image* berupa logo aplikasi dan sebuah *TextView* bertuliskan judul halaman, yaitu “*My Appliance*”. <appliance_icon> merupakan *ImageButton* sesuai dengan jenis perangkat. Pada <plus button> akan diisi gambar simbol (+) pada sebuah *ImageButton*.



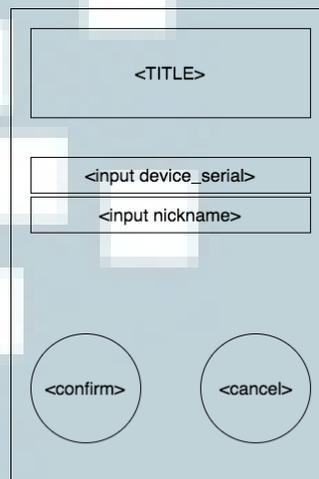
Gambar 3. 12 Rancangan Tampilan Halaman Main Setting

3.6.4 Rancangan Halaman Add Appliance

Halaman ini merupakan halaman untuk menambah perangkat pintar baru. Pada halaman ini terdapat dua buah masukan dari *user* berupa nomor seri perangkat pintar dan nama panggilan untuk perangkat tersebut. Kedua data ini akan dikirim ke server untuk dimasukkan ke dalam *database*. Bila berhasil, tampilan aplikasi akan kembali ke halaman *appliance* dan bila tidak, akan tetap pada halaman *add appliance* dengan pesan gagal. Diagram alur untuk halaman ini dapat dilihat pada Lampiran.

Gambar 3.13 menampilkan rancangan halaman *add appliance* yang berguna untuk mendaftarkan perangkat baru. Pada <TITLE> akan berisi *Image* berupa

logo aplikasi dan *TextView* yang berisi judul halaman yaitu “*Add Appliance*”. `<input device_serial>` akan diisi dengan *TextView* yang menuliskan “*Serial Number*” dan sebuah *EditText* untuk pengguna memasukkan nomor serial perangkat. Pada `<input nickname>` juga akan diisi dengan *TextView* yang menuliskan “*Nickname*” dan sebuah *EditText* untuk pengguna memasukkan nama panggilan perangkat.



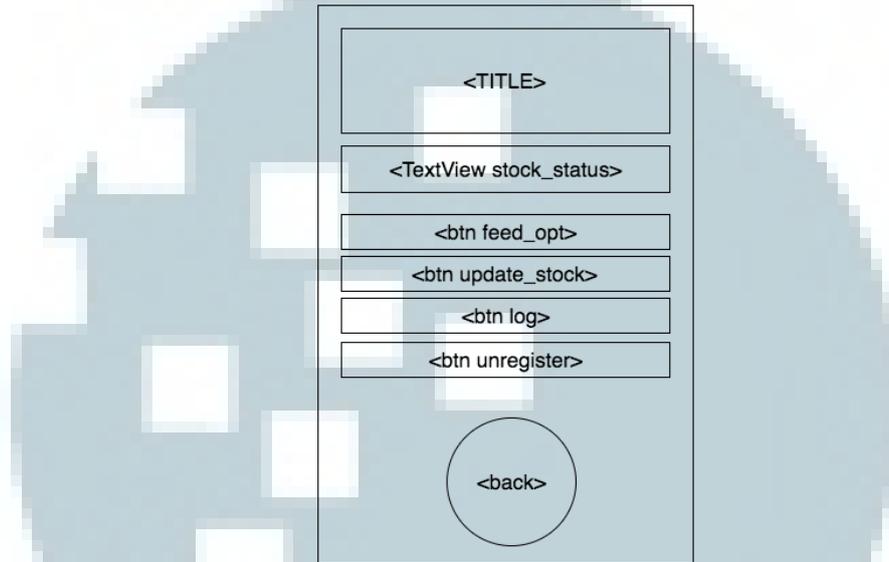
Gambar 3.13 Rancangan Tampilan Halaman Add Appliance

3.6.5 Rancangan Halaman Setting

Halaman ini merupakan halaman utama untuk segala pengaturan mengenai perangkat pintar. Di dalamnya terdapat sebuah *TextView* yang berguna untuk memberi informasi mengenai stok dari perangkat pintar. Pada *Smart Dog Feeder* akan diberi informasi mengenai stok makanan anjing yang terdapat pada perangkat. Selain itu, pada halaman *setting*, terdapat tombol *Feed*, tombol *Update Stock*, tombol *Log*, dan tombol *Unregister*. Diagram alur untuk halaman ini dapat dilihat pada Lampiran.

Berikut pada Gambar 3.14 merupakan tampilan halaman *setting* untuk *Smart Dog Feeder*. Pada `<TITLE>` akan berisi *Image* berupa logo aplikasi dan

TextView yang berisi nama panggilan perangkat. `<TextView stock_status>` adalah *TextView* yang berisi informasi mengenai kondisi stok pada perangkat. Tombol `<btn feed_opt>`, `<btn update_stock>`, `<btn log>`, dan `<btn unregister>` adalah *Button*. Sedangkan tombol `<back>` merupakan *ImageButton*.



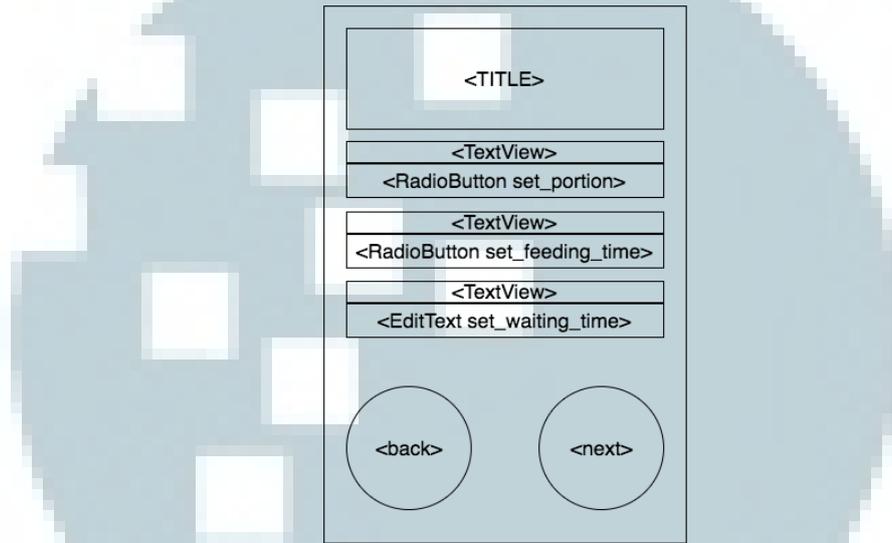
Gambar 3. 14 Rancangan Tampilan Halaman Setting Smart Dog Feeder

3.6.6 Rancangan Halaman Feed

Halaman ini merupakan halaman untuk mengatur *Smart Dog Feeder*. Pada halaman ini terdapat dua buah *radio button* yang berfungsi untuk *user* memilih besar porsi per sajian dan jumlah pemberian makan. Selain itu terdapat sebuah *EditText* untuk memasukkan waktu tunggu pengecekan makanan. Dari ketiga masukan data *user*, akan di-*passing* melalui fitur *Intent* untuk dipakai di halaman *Feed Time*. Diagram alur untuk halaman ini dapat dilihat pada Lampiran 5.

Gambar 3.15 menampilkan rancangan halaman *feed* untuk pengaturan *Smart Dog Feeder*. `<TITLE>` merupakan *Image* logo aplikasi dan *TextView* berisi nama panggilan perangkat. Masing-masing `<TextView>` berisi teks yang akan memandu pengguna dalam memilih pilihan-pilihan dibawahnya. `<RadioButton`

set_portion> dan <RadioButton set_feeding_time> dipilih agar pengguna dapat langsung menentukan pilihan. Sedangkan <EditText set_waiting_time> memakai *EditText* sehingga pengguna dapat memasukkan nilai sesuai dengan keinginannya. Namun, *EditText* ini diatur hanya dapat dimasukkan format angka desimal.



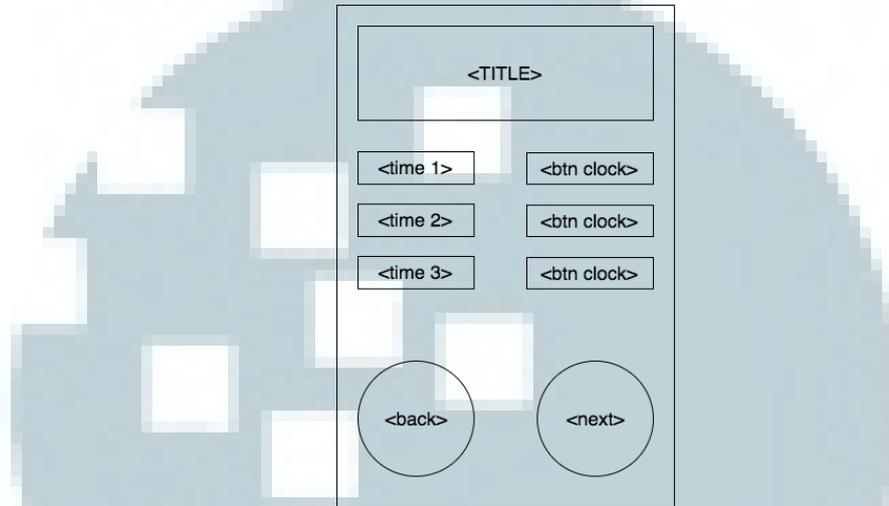
Gambar 3.15 Rancangan Tampilan Halaman Feed

3.6.7 Rancangan Halaman Feed Time

Halaman ini merupakan lanjutan dari halaman *Feed*. Halaman ini akan berisi *TextView* untuk menandakan waktu dan sebuah tombol untuk mengatur waktu. Banyaknya baris pilihan waktu ditentukan pada halaman *Feed*, yaitu jumlah pemberian makan. Setelah mendapat input dari *user* dan *user* menekan tombol *confirm*, aplikasi akan mengirim perintah yang berisi data dari halaman *Feed* dan *Feed Time* ke server. Diagram alur untuk halaman ini dapat dilihat pada Lampiran.

Gambar 3.16 menampilkan rancangan halaman *feed time* untuk pengaturan waktu *Smart Dog Feeder*. <TITLE> merupakan *Image* logo aplikasi dan

TextView berisi nama panggilan perangkat. `<time 1>`, `<time 2>`, dan `<time 3>` adalah *TextView* yang jumlahnya dipengaruhi oleh masukkan pengguna pada halaman *Feed*. `<btn clock>` merupakan *Button* yang akan menampilkan *TimePicker* bila ditekan. Tombol `<back>` dan `<next>` adalah *ImageButton*.



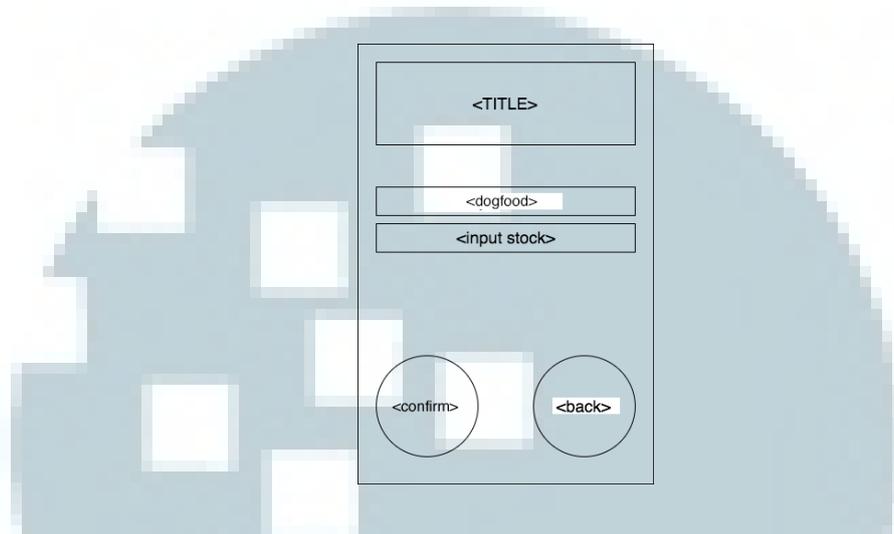
Gambar 3. 16 Rancangan Tampilan Halaman Feed Time

3.6.8 Rancangan Halaman Update Stock

Halaman ini bertugas untuk mengambil *input* dari *user* mengenai stok yang ingin ditambah ke dalam perangkat pintar. Tampilan dari halaman ini ditentukan oleh jenis perangkat pintar. Apabila *Smart Dog Feeder*, akan berisi *EditText* untuk memasukkan jumlah makanan anjing. Sedangkan pada *Smart Rice Cooker* terdapat dua *EditText* untuk memasukkan jumlah beras dan air dan pada *Smart Medicine Dispenser* akan berisi dua *EditText* untuk memasukkan jumlah pil dan obat cair. Stok akan diperbaharui ke server setelah mendapat persetujuan dari perangkat pintar. Diagram alur untuk halaman ini dapat dilihat pada Lampiran.

Tampilan halaman *update stock* akan menyesuaikan dengan jenis perangkat pintar. Gambar 3.17 adalah rancangan tampilan untuk *Smart Dog Feeder*. `<TITLE>` merupakan *Image* logo aplikasi dan *TextView* berisi judul “*Update*

Stock". `<input stock>` merupakan *EditText* yang digunakan untuk memasukkan jumlah stok yang ingin ditambahkan dan diberi *hint* agar pengguna tahu bahwa pengisian stok dalam satuan gram. Tombol `<back>` dan `<next>` adalah *ImageButton*.

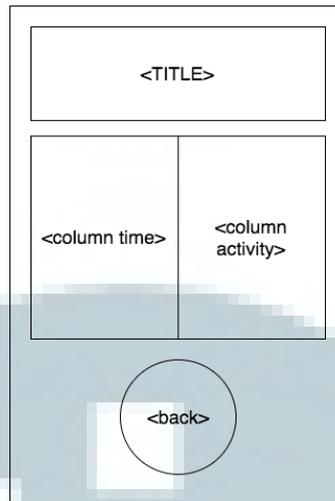


Gambar 3.17 Rancangan Tampilan Halaman Update Stock Smart Dog Feeder

3.6.9 Rancangan Halaman Log

Halaman ini bertugas untuk menampilkan seluruh laporan mengenai aktifitas yang dilakukan perangkat pintar. Laporan tersebut akan ditampilkan dalam bentuk tabel. Bila tidak terdapat laporan, hanya akan terdapat tulisan “*No Log*”. Diagram alur untuk halaman ini dapat dilihat pada Lampiran.

Gambar 3.18 adalah rancangan tampilan halaman *log* yang menampilkan catatan aktivitas setiap perangkat. `<TITLE>` merupakan *Image* logo aplikasi dan *TextView* berisi judul “*Log*”. `<column time>` dan `<column activity>` berada pada sebuah *TableLayout* untuk memetakan kolom waktu dan kolom aktivitas. Waktu ditulis pada sebuah *TextView* di dalam `<column time>`. Aktivitas ditulis pada sebuah *TextView* di dalam `<column activity>`. Tombol `<back>` merupakan *ImageButton*.

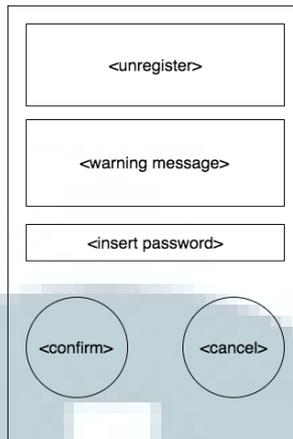


Gambar 3. 18 Rancangan Tampilan Halaman Log

3.6.10 Rancangan Halaman Unregister

Halaman ini berfungsi untuk menghapus sebuah perangkat dari data *user*. Untuk menghapus perangkat, *user* diminta memasukkan *password* pada *EditText* yang disediakan pada halaman ini. Setelah menekan tombol *confirm*, aplikasi akan mengirimkan perintah ke server. Apabila berhasil akan kembali ke halaman *appliance*, jika tidak akan tetap pada halaman ini dan diberi pesan gagal. Diagram alur untuk halaman ini dapat dilihat pada Lampiran.

Gambar 3.19 adalah rancangan tampilan halaman *unregister*. Pada *<unregister>* merupakan judul yang ditulis pada *TextView*. *<warning message>* merupakan *TextView* yang berfungsi sebagai tempat pesan peringatan bahwa dengan melakukan proses tersebut data pengguna akan dihapus. Pada *<insert password>* merupakan *EditText* tempat pengguna memasukkan *password* sebagai persetujuan. Tombol *<confirm>* dan *<cancel>* adalah *ImageButton*.



Gambar 3. 19 Rancangan Tampilan Halaman Unregister

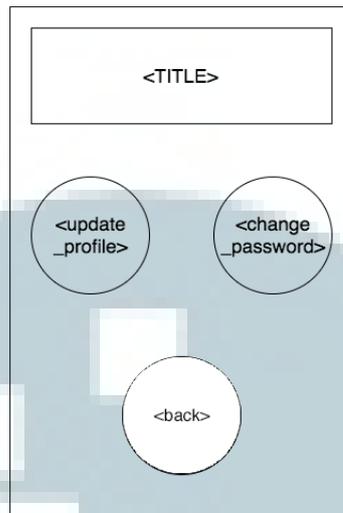
3.6.11 Rancangan Halaman Account Setting

Halaman ini berguna untuk memilih antara melakukan perubahan profil atau mengganti *password*. Pada halaman ini juga terdapat tombol *back* untuk kembali ke halaman *Main Setting*. Diagram alur untuk halaman ini dapat dilihat pada Lampiran.

Gambar 3.20 adalah rancangan tampilan halaman *account setting*. <TITLE> merupakan *Image* logo aplikasi dan *TextView* berisi judul “*Account Setting*”. Ketiga tombol, yakni <update_profile>, <change_password>, dan <back> adalah *ImageButton*.

3.6.12 Rancangan Halaman Edit Profile

Halaman ini menampilkan profil pengguna dalam *EditText* yang dapat diubah sesuai keinginan *user*. Setelah mengisi data dan menekan tombol *confirm*, aplikasi akan mengirimkan data ke server. Bila berhasil akan diberi pesan berhasil dan kembali ke halaman *Account Setting*. Jika gagal akan diberi pesan gagal dan tetap berada pada halaman *Edit Profile*. Diagram alur untuk halaman ini dapat dilihat pada Lampiran.



Gambar 3. 20 Rancangan Tampilan Halaman Account Setting

Gambar 3.21 adalah rancangan tampilan halaman *edit profile* yang berfungsi untuk melakukan pembaruan profil pengguna. <TITLE> merupakan *Image* logo aplikasi dan *TextView* berisi judul “*Update Profile*”. <input username> berisi sebuah *TextView* bertuliskan “*Username*” dan *EditText* yang tidak dapat diubah dan berisi *username* pengguna. Pada <input full_name> berisi sebuah *TextView* bertuliskan “*Full Name*” dan *EditText* untuk diisi oleh pengguna bila ingin perubahan. <input email> dan <input phone_number> masing-masing berisi sebuah *TextView* bertuliskan “*Email*” dan “*Phone Number*” dan *EditText* untuk diisi oleh pengguna. Tombol <confirm> dan <cancel> adalah sebuah *ImageButton*.

```

<TITLE>
<input username>
<input fullname>
<input email>
<input phone_number>
<confirm> <cancel>

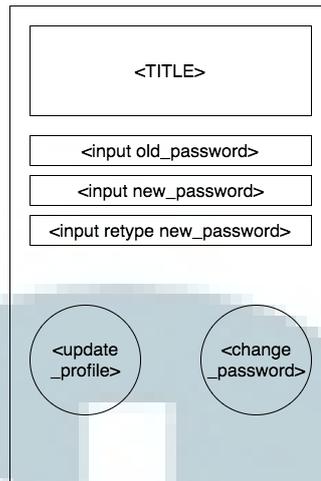
```

Gambar 3. 21 Rancangan Tampilan Halaman Edit Profile

3.6.13 Rancangan Halaman Change Password

Halaman ini berfungsi untuk merubah *password*. *User* diminta untuk memasukkan *password* lama, *password* baru, dan mengulangi *password* baru. *Password* baru akan diperiksa kembali agar tidak sama dengan *password* sebelumnya. Setelah data masukan benar aplikasi akan mengirimkan perintah ke server. Bila berhasil akan kembali ke halaman *Account Setting*. Sedangkan jika gagal, akan tetap pada halaman tersebut dan menampilkan pesan gagal. Diagram alur untuk halaman ini dapat dilihat pada Lampiran 12.

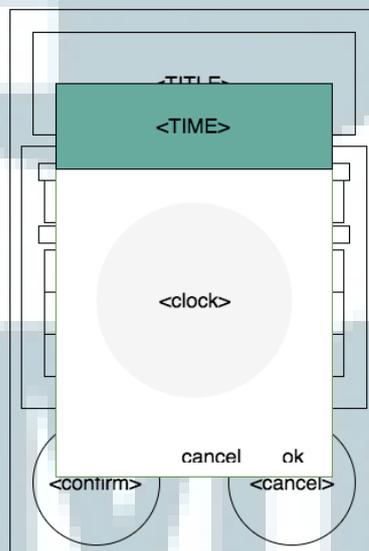
Gambar 3.22 adalah rancangan tampilan halaman *change password*. `<input old_password>`, `<input new_password>`, dan `<input retype new_password>` masing-masing memiliki *TextView* dan *EditText*. *EditText* masing-masing diberi tipe *password* sehingga bila diberi masukan akan tidak terbaca.



Gambar 3. 22 Rancangan Tampilan Halaman Change Password

3.6.14 Rancangan Tampilan Halaman Saat Memilih Waktu

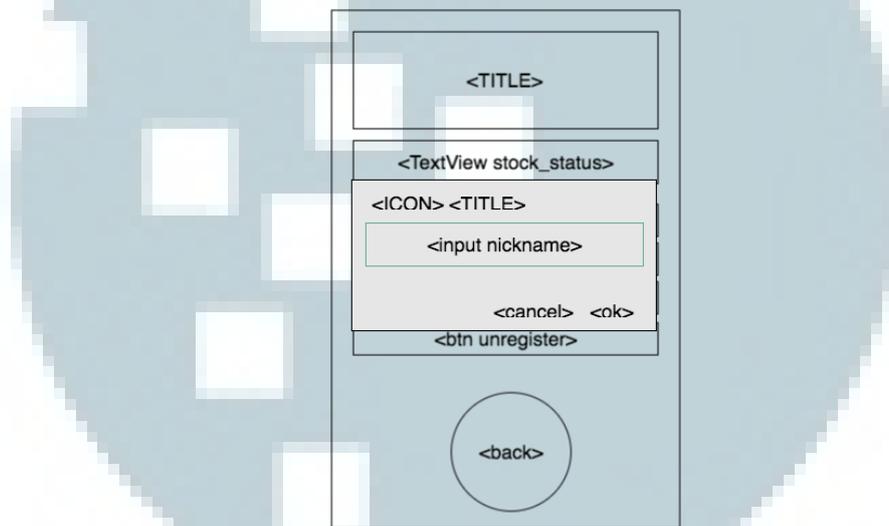
Gambar 3.23 merupakan rancangan tampilan halaman saat pengguna memilih waktu pemberian makan, waktu memasak nasi atau bubur, dan waktu pemberian obat. Pada tiap halaman pengaturan waktu, akan muncul jendela *pop-up* untuk mengatur jam dari *TimePicker*.



Gambar 3. 23 Rancangan Tampilan Halaman Saat Memilih Waktu

3.6.15 Rancangan Tampilan Halaman Saat Mengubah Nickname

Gambar 3.24 merupakan rancangan tampilan halaman saat pengguna menekan tulisan nama panggilan perangkat untuk mengubah nama perangkat tersebut. Sebuah *AlertDialog* akan muncul dan berisi <ICON> yaitu logo aplikasi, <TITLE> berupa *TextView* bertuliskan “Rename Appliance”, <input nickname> yang berupa *EditText* tempat pengguna memberi masukan, <cancel>, dan <ok> yang berupa *button* bawaan dari *AlertDialog*.



Gambar 3. 24 Rancangan Tampilan Halaman Saat Mengubah Nickname

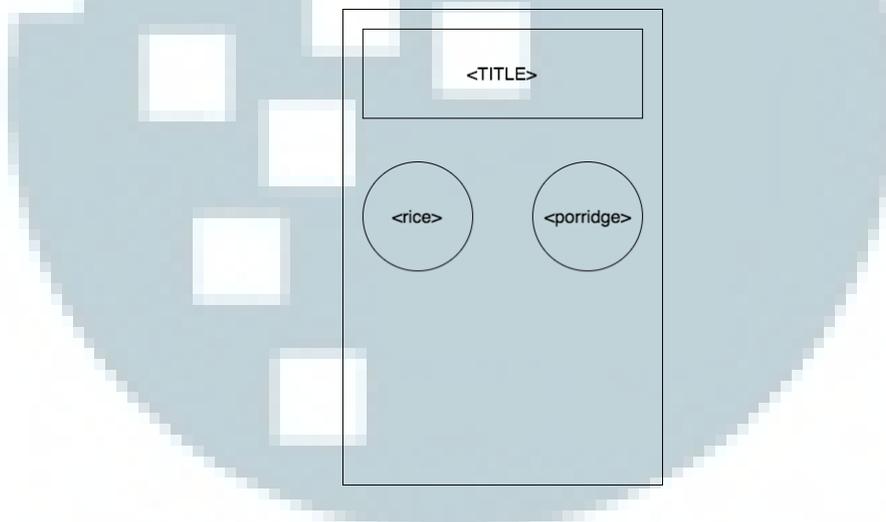
3.6.16 Rancangan Tampilan Aplikasi Pada Perangkat Pintar Lain

Pada penelitian tugas akhir ini, penulis juga merancang tampilan aplikasi untuk dua perangkat pintar lain, yaitu *Smart Rice Cooker* dan *Smart Medicine Dispenser*. Oleh karena itu, aplikasi Appliance Hub dapat digunakan pada perangkat pintar tersebut. Berikut adalah rancangan tampilan menu untuk perangkat lainnya.

A. Rancangan Tampilan Halaman Cook Menu Smart Rice Cooker

Halaman ini merupakan halaman khusus untuk perangkat *Smart Rice Cooker*. Halaman ini menampilkan dua macam hidangan, yaitu nasi dan bubur. Setelah *user* memilih akan berpindah ke halaman *Cook* untuk meneruskan pengaturan. Diagram alur untuk halaman ini dapat dilihat pada Lampiran.

Gambar 3.25 adalah rancangan tampilan halaman *cook menu* untuk *Smart Rice Cooker*. `<TITLE>` merupakan *Image* logo aplikasi dan *TextView* berisi nama panggilan perangkat. Tombol `<rice>` dan `<porridge>` adalah *ImageButton*.



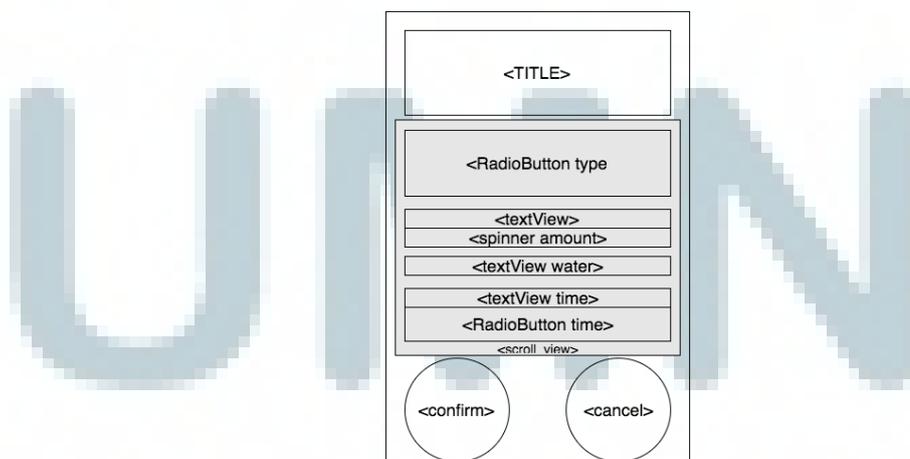
Gambar 3. 25 Rancangan Tampilan Halaman Cook Menu

B. Rancangan Tampilan Halaman Cook Smart Rice Cooker

Halaman ini merupakan lanjutan pengaturan dari halaman *Cook Menu*. Pilihan nasi atau bubur pada halaman sebelumnya akan berpengaruh pada pengaturan takaran beras dan air. Pada halaman ini akan menampilkan sebuah *RadioButton* untuk memilih tekstur nasi atau bubur, seperti *thick* untuk tekstur padat, *medium* untuk sedang, dan *thin/soft* untuk tekstur halus. Setelah memilih tekstur, *user* akan memilih jumlah porsi yang akan dimasak pada sebuah *Spinner* yang isinya secara otomatis menyesuaikan dengan pilihan tekstur sebelumnya.

Kadar air juga akan otomatis mengikuti jumlah beras yang dimasak. Selanjutnya *user* akan memilih waktu masak yang diinginkan. Bila memilih pilihan “*NOW*”, menandakan setelah pengiriman perintah perangkat akan langsung mengerjakan perintah tersebut. Bila memilih “*Set Time*”, *user* akan diminta mengatur waktu yang diinginkan. Kemudian setelah melalui *confirm*, aplikasi akan mengirimkan perintah ke server. Bila berhasil, akan kembali ke halaman *Setting*. Bila tidak, akan tetap pada halaman tersebut dan menampilkan pesan gagal. Diagram alur untuk halaman ini dapat dilihat pada Lampiran.

Gambar 3.26 adalah rancangan tampilan *halaman cook* untuk *Smart Rice Cooker*. <TITLE> merupakan *Image* logo aplikasi dan *TextView* berisi nama panggilan perangkat. <RadioButton type> merupakan *RadioButton* yang digunakan untuk memilih tipe nasi atau bubur yang diinginkan (*thick*, *medium*, atau *soft/thin*). Di bawah *RadioButton* terdapat *ScrollView* yang didalamnya terdapat tiga *TextView*, sebuah *Spinner* yang berisi pilihan porsi masak, dan *RadioButton* untuk memilih waktu masak. <confirm> dan <cancel> merupakan *ImageButton*.



Gambar 3. 26 Rancangan Tampilan Halaman Cook

C. Rancangan Tampilan Halaman Pill Smart Medicine Dispenser

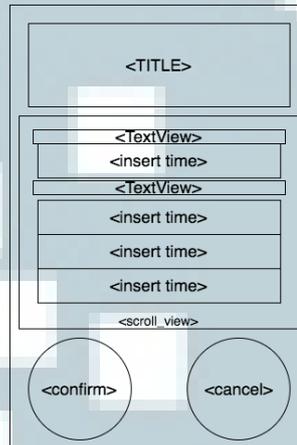
Halaman ini merupakan halaman khusus perangkat *Smart Medicine Dispenser*. Halaman ini bertugas menerima data jumlah pil yang dikeluarkan pada satu waktu beserta jadwal pemberian obat yang memiliki maksimal enam waktu. Jumlah pil akan dimasukkan pada sebuah *EditText* dan waktu akan diatur dengan menekan tombol yang kemudian menampilkan *TimePicker*. Setelah *user* memilih *next*, data akan disimpan dan diteruskan pada halaman *Sol*. Diagram alur untuk halaman ini dapat dilihat pada Lampiran.

Gambar 3.27 adalah rancangan tampilan halaman *pill* untuk *Smart Medicine Dispenser*. `<TITLE>` merupakan *Image* logo aplikasi dan *TextView* berisi nama panggilan perangkat. `<TextView>` merupakan *TextView* yang berfungsi untuk memberi informasi kepada pengguna dalam pengisian data. `<insert amount>` merupakan *EditText* yang bertipe angka desimal untuk pengguna menentukan jumlah obat. `<insert time>` berisi *TextView* bertuliskan “*Time 1*” sampai dengan “*Time 6*” dan *Button* yang berfungsi untuk menampilkan *TimePicker*. Keseluruhan `<TextView>`, `<insert amount>`, dan `<insert time>` berada pada `<scroll_view>` yang merupakan *ScrollView*. `<confirm>` dan `<cancel>` merupakan *ImageButton* yang terpisah dari *ScrollView*.

D. Rancangan Tampilan Halaman Sol Smart Medicine Dispenser

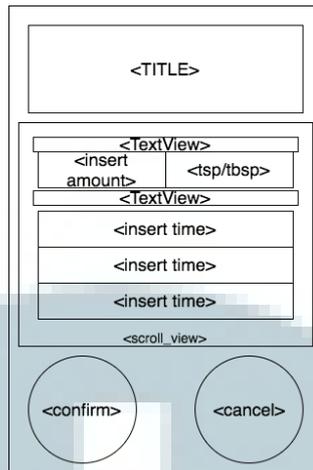
Halaman ini akan mengambil data dari halaman *Pill*. Bila halaman *Pill* tidak memiliki jumlah dan jadwal, halaman *Sol* harus memiliki penjadwalan serta jumlah obat cair yang diinginkan. Hal ini diperlukan untuk memastikan *user* minimal mengatur satu jenis obat dan mengirimkan kepada server. Pada halaman *Sol*, *user* akan memasukkan jumlah obat cair yang diberikan dalam satu kali

pemberian obat dan waktu-waktu yang diinginkan seperti pada halaman *Pill*. Setelah menekan tombol *confirm*, aplikasi akan mengirimkan data ke server untuk diteruskan kembali ke perangkat pintar. Diagram alur untuk halaman ini dapat dilihat pada Lampiran.



Gambar 3. 27 Rancangan Tampilan Halaman Pill

Gambar 3.28 adalah rancangan tampilan halaman *sol* untuk *Smart Medicine Dispenser*. <TITLE> merupakan *Image* logo aplikasi dan *TextView* berisi nama panggilan perangkat. <TextView> merupakan *TextView* yang berfungsi untuk memberi informasi kepada pengguna dalam pengisian data. <insert amount> merupakan *EditText* tempat pengguna menuliskan jumlah obat dan <tsp/tbsp.> merupakan *Spinner* untuk pengguna memilih ukuran takaran obat. <insert time> berisi *TextView* bertuliskan “*Time 1*” sampai dengan “*Time 6*” dan *Button* yang berfungsi untuk menampilkan *TimePicker*. Keseluruhan <TextView>, <insert amount>, <tsp/tbsp.> dan <insert time> berada pada <scroll_view> yang merupakan *ScrollView*. <confirm> dan <cancel> merupakan *ImageButton* yang terpisah dari *ScrollView*.

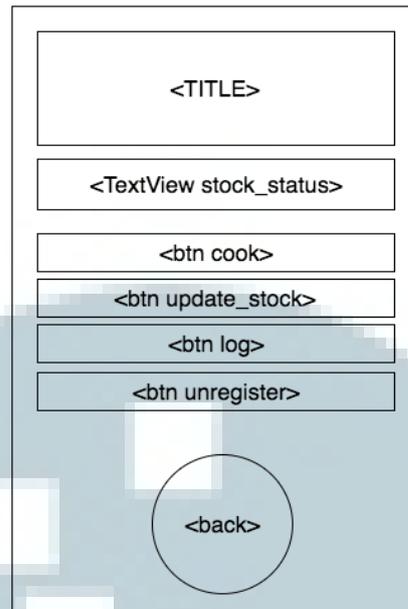


Gambar 3. 28 Rancangan Tampilan Halaman Sol

E. Rancangan Tampilan Halaman Setting Smart Rice Cooker

Halaman *Setting* pada umumnya memiliki tampilan yang sama dengan *Smart Dog Feeder*. Hanya saja terdapat beberapa perbedaan yang sesuai dengan pengaturan perangkat. Pada *Smart Rice Cooker* terdapat informasi mengenai beras dan air. Selain itu, pada halaman *setting*, terdapat tombol *Cook*, tombol *Update Stock*, tombol *Log*, dan tombol *Unregister*.

Pada Gambar 3.29 akan terlihat perbedaan tampilan pada tombol pertama yang berubah menjadi *cook*. Gambar 3.29 merupakan tampilan halaman *setting* khusus untuk perangkat *Smart Rice Cooker*. *<TITLE>* merupakan *Image* logo aplikasi dan *TextView* berisi nama panggilan perangkat. *<TextView stock_status>* merupakan *TextView* yang berisi informasi stok, *<btn cook>*, *<btn update_stock>*, *<btn_log>*, dan *<btn unregister>* merupakan *Button*. Sedangkan *<back>* merupakan *ImageButton*.

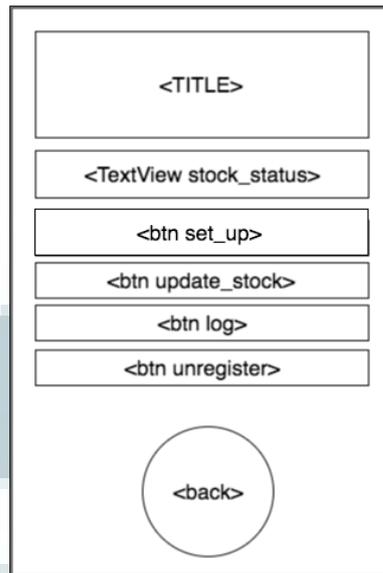


Gambar 3. 29 Rancangan Tampilan Halaman Setting Smart Rice Cooker

F. Rancangan Tampilan Halaman Setting Smart Medicine Dispenser

Halaman *Setting* pada *Smart Medicine Dispenser* akan berisi informasi mengenai jumlah pil dan obat cair. Selain itu, pada halaman *setting*, terdapat tombol *Set Up*, tombol *Update Stock*, tombol *Log*, dan tombol *Unregister*.

Pada Gambar 3.30 merupakan tampilan halaman *Setting* untuk perangkat *Smart Medicine Dispenser*. *<TITLE>* merupakan *Image* logo aplikasi dan *TextView* berisi nama panggilan perangkat. *<TextView stock_status>* merupakan *TextView* yang berisi informasi stok, *<btn set_up>*, *<btn update_stock>*, *<btn_log>*, dan *<btn unregister>* merupakan *Button*. Sedangkan *<back>* merupakan *ImageButton*.



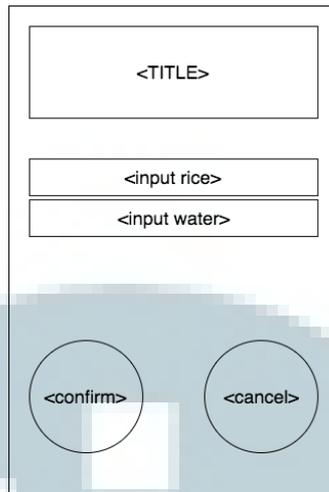
Gambar 3. 30 Rancangan Tampilan Halaman Setting Smart Medicine Dispenser

G. Rancangan Tampilan Halaman Update Stock Smart Rice Cooker

Halaman ini bertugas untuk mengambil *input* dari *user* mengenai stok yang ingin ditambah ke dalam perangkat pintar. Tampilan dari halaman ini pada *Smart Rice Cooker* terdapat dua *EditText* untuk memasukkan jumlah beras dan air. Stok akan diperbaharui ke server setelah mendapat persetujuan dari perangkat pintar. Diagram alur untuk halaman ini dapat dilihat pada Lampiran.

Gambar 3.31 adalah rancangan tampilan untuk *Smart Rice Cooker*.

<TITLE> merupakan *Image* logo aplikasi dan *TextView* berisi judul “*Update Stock*”. <input rice> dan <input water> masing-masing berisi *TextView* untuk panduan pengguna dalam memasukkan stok dan *EditText* yang digunakan untuk memasukkan jumlah stok yang ingin ditambahkan dan diberi *hint* agar pengguna tahu bahwa pengisian nasi dalam satuan gram dan pengisian air dalam ml. Tombol <back> dan <cancel> adalah *ImageButton*.



Gambar 3. 31 Rancangan Tampilan Halaman Update Stock Smart Rice Cooker

H. Rancangan Tampilan Halaman Update Stock Smart Medicine Dispenser

Halaman ini bertugas untuk mengambil *input* dari *user* mengenai stok yang ingin ditambah ke dalam perangkat pintar. Tampilan dari halaman ini pada *Smart Medicine Dispenser* akan berisi dua *EditText* untuk memasukkan jumlah pil dan obat cair. Stok akan diperbaharui ke server setelah mendapat persetujuan dari perangkat pintar. Diagram alur untuk halaman ini dapat dilihat pada Lampiran.

Gambar 3.32 adalah rancangan tampilan untuk *Smart Medicine Dispenser*.

<TITLE> merupakan *Image* logo aplikasi dan *TextView* berisi judul “*Update Stock*”. <input pill> dan <input liquid> masing-masing berisi *TextView* untuk panduan pengguna dalam memasukkan stok dan *EditText* yang digunakan untuk memasukkan jumlah stok yang ingin ditambahkan dan diberi *hint* agar pengguna tahu bahwa pengisian pil dalam satuan butir dan pengisian obat cair dalam ml. Tombol <back> dan <cancel> adalah *ImageButton*.

The diagram shows a rectangular form with a white background and a thin black border. At the top is a wide, shallow rectangular box containing the text '<TITLE>'. Below this are two stacked rectangular boxes. The top one is wider and contains '<insert pill>', while the bottom one is narrower and contains '<insert liquid>'. At the bottom of the form are two circular buttons. The left button contains '<confirm>' and the right button contains '<cancel>'. The entire form is centered on a light blue background that has a faint, large watermark of the letters 'UMN'.

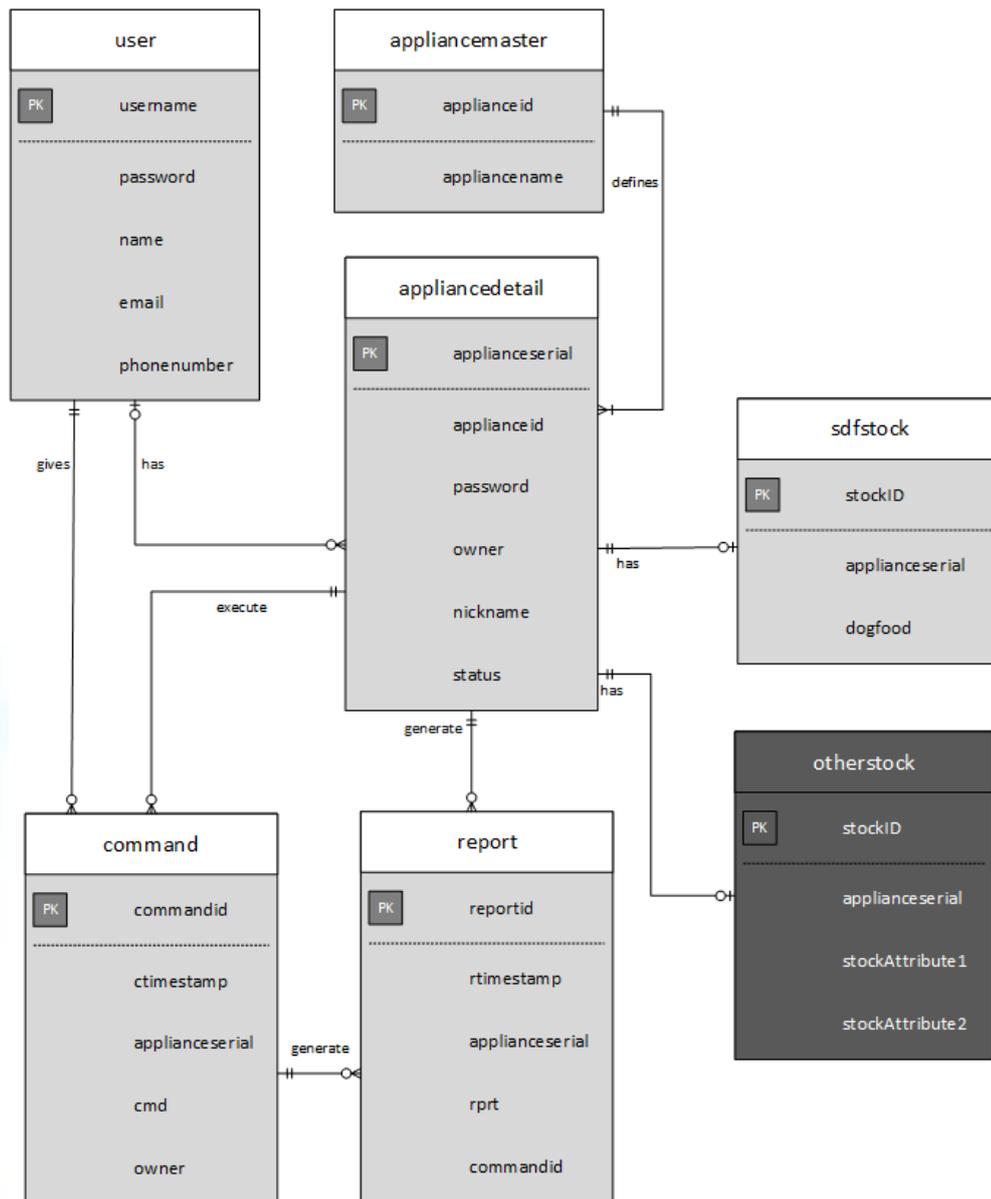
Gambar 3. 32 Rancangan Tampilan Halaman Update Stock Smart Medicine Dispenser

3.7 Perancangan Database

Sistem *Smart Dog Feeder* menggunakan sebuah *database* yang menggunakan MySQL dan diletakkan pada *cloud server*. Gambar 3.33 merupakan *Entity Relationship Diagram* (ERD) dari *database* sistem Appliance Hub khusus perangkat *Smart Dog Feeder*, dengan struktur dari setiap tabel adalah sebagai berikut.

3.7.1 Tabel *appliance*master

Tabel *appliance*master dapat dilihat pada Tabel 3.37. Tabel ini berisi *applianceid* dan *appliance*name. Kedua data tersebut merupakan identitas dari perangkat pintar yang terdapat di sistem Appliance Hub. *Applianceid* berisi ID dari perangkat, seperti 02 dan memiliki *Appliance*name yang merupakan nama dari *applianceid*, seperti ID 02 memiliki nama *Smart Dog Feeder*.



Gambar 3. 33 Entity Relation Diagram dari sistem Appliance Hub [30]

Tabel 3. 37 Tabel *applianceidmaster*

#	Name	Type
1	applianceid	varchar(3)
2	appliance name	varchar(50)

3.7.2 Tabel *applianceidetail*

Tabel *applianceidetail* merupakan tabel yang berisi informasi lebih detail dari sebuah perangkat pintar. Tabel ini terdiri dari *appliance serial*, *applianceid*

yang merupakan turunan dari *appliancemaster*, *password*, *owner*, *nickname*, dan *status*. *Applianceserial* akan diambil dari alamat MAC perangkat. *Password* digunakan perangkat pintar untuk mendapat akses server. *Owner* dan *nickname* akan terisi bila perangkat pintar sudah memiliki pemilik. *Status* merupakan keadaan terakhir dari perangkat, *active* atau *inactive*. Tabel ini dapat dilihat pada Tabel 3.38.

Tabel 3. 38 Tabel *appliancedetail*

#	Name	Type
1	<i>applianceserial</i> 🔑	varchar(15)
2	<i>applianceid</i>	varchar(3)
3	<i>password</i>	varchar(100)
4	<i>owner</i>	varchar(50)
5	<i>nickname</i>	varchar(50)
6	<i>status</i>	varchar(10)

3.7.3 Tabel *user*

Tabel *user* dapat dilihat pada Tabel 3.39. Tabel ini adalah tabel yang memuat informasi mengenai akun pada sistem Appliance Hub. Tabel ini terdiri dari *username*, *password*, *name*, *email*, dan *phonenummer*. *Username* dan *password* akan digunakan *user* untuk mendapat akses ke server melalui aplikasi Android.

3.7.4 Tabel *sdfstock*

Tabel *sdfstock* berfungsi untuk menyimpan stok makanan dari *Smart Dog Feeder*. Tabel ini berisi *stockID*, *applianceserial*, dan *dogfood*. *StockID* merupakan ID dari setiap stok yang tersimpan dan dibuat *auto increment*, *applianceserial* digunakan untuk memberi relasi antara stok dan perangkat, dan *dogfood* adalah jumlah makanan yang terdapat pada perangkat.

Tabel 3. 39 Tabel user

#	Name	Type
1	username 	varchar(50)
2	password	varchar(100)
3	name	varchar(25)
4	email	varchar(50)
5	phonenumber	varchar(15)

Tabel 3. 40 Tabel sdfStock

#	Name	Type
1	stockID 	int(11)
2	applianceserial	varchar(15)
3	dogfood	int(5)

3.7.5 Tabel command

Tabel *command* menyimpan setiap perintah yang diberikan aplikasi Android ke server. Tabel ini memiliki *commandid*, *ctimestamp* untuk mengetahui waktu perintah diberikan, *applianceserial* untuk mengetahui perangkat, *cmd* yang berisi perintah lengkap yang diterima server, dan *owner* yang berisi *username* pemilik perangkat.

Tabel 3. 41 Tabel command

#	Name	Type
1	commandid 	int(11)
2	ctimestamp	datetime
3	applianceserial	varchar(15)
4	cmd	varchar(256)
5	owner	varchar(50)

3.7.6 Tabel report

Tabel *report* merupakan tabel yang menyimpan laporan dari perangkat pintar. Tabel ini berisi *reportid*, *rtimestamp* untuk menyimpan waktu laporan diterima server, *applianceserial* untuk informasi perangkat, *rprt* yang berisi laporan dari perangkat yang diterima server, dan *commandid* yang merupakan relasi dari tabel *command*.

Tabel 3. 42 Tabel report

#	Name	Type
1	reportid 	int(11)
2	rtimestamp	datetime
3	applianceserial	varchar(15)
4	rprt	varchar(256)
5	commandid	int(11)

U
M
M
N