BAB III

RANCANGAN SISTEM

3.1. Spesifikasi Wadah

Spesifikasi wadah pintar yang dirancang adalah sebagai berikut.

a. Maksimum berat total yang dapat ditampung adalah 7 kg.
b. Dapat menggunakan protokol *Bluetooth* atau *Zigbee* yang dapat ditentukan dengan menyambungkan atau tidak menyambungkan pin dengan pin yang lain.
c. Pengecekan persentase dilakukan di frekuensi yang ditentukan dengan satuan menit. Maksimum frekuensi adalah 65.535 menit. Pengiriman update hanya dilakukan jika ada perubahan persentase.
d. Pengiriman umur dilakukan di frekuensi yang ditentukan dengan satuan hari. Maksimum frekuensi adalah 65.535 hari.
e. Umur kembali ke 0 hari saat tombol 100% ditekan.

3.2. Perancangan *Hardware* Wadah

Komponen-komponen yang digunakan untuk membangun wadah pintar adalah sebagai berikut.

a. *Strain gauge* diambil dari timbangan digital sf-400.

Spesifikasi timbangan:

- Kapasitas Timbangan: 7kg.

- Ketelitian Timbangan: 1gr.
- Sumber daya: 3V (2 buah baterai AA).

b. Mikrokontroler Atmega8535 beserta sistem minimumnya.
c. *Operational Amplifier AD620* untuk menguatkan sinyal dari *strain gauge*.
d. Rangkaian pembagi tegangan dari 6V ke 2,6V untuk sumber daya *op-amp*.
e. Baterai AA *Alkaline* 1.5V 4 buah sebagai sumber daya untuk wadah pintar.
f. Modul *bluetooth* HC-05.
g. Modul *zigbee* ETRX2.
h. *Push button* untuk button 0%, 100%, dan *checkID*.
i. Saklar ON/OFF.
j. Resistor 330Ω 4 buah.
k. Dioda 10 buah.

Gambar 3.1 menggambarkan skematik wadah saat menggunakan *bluetooth* sebagai media komunikasi. Gambar 3.2 menggambarkan skematik wadah saat menggunakan *zigbee* sebagai media komunikasi. Persamaan kedua skematik ada di rangkaian tegangan, *op-amp*, *strain gauge*, dan tombol input. Rangkaian XTAL dan kapasitor merupakan rangkaian sistem minimum ATMega8535. XTAL yang digunakan adalah 4MHz. Kutub positif baterai disambungkan ke saklar ON/OFF sebelum disambungkan ke rangkaian.

ATMega8535 dapat menerima input dari 4.5V sampai 5.5V. Maka dari itu, voltase input tidak dapat diambil dari langsung dari baterai yang bernilai 6V. Voltase diambil dari keluaran dioda yang diseri sebanyak dua buah yang masing-masing dioda menurunkan 0.3V sehingga masukan yang diterima ATMega8535 adalah 6V-2*0.3V = 5.4V.

Strain gauge yang diambil dari timbangan memiliki empat kabel, yaitu warna merah untuk E+ (voltase input), warna hitam untuk E- (GND), warna biru untuk S+ (*sensing* positif), dan warna putih untuk S- (*sensing* negatif). Karena sudah ada empat kabel, sensor dari timbangan tersebut sudah mengandung jembatan *wheatstone*. Kabel S+ dan S- masuk ke pin +IN dan –IN pada *op-amp* untuk dinaikkan tegangannya. Perbedaan voltase pada S+ dan S- bernilai sangat kecil. Perubahan voltase saat diberi beban dapat hanya sekitar 0.01mA sedangkan ADC pada ATMega8535 memiliki ketelitian sekitar 5mA. Voltase input sensor adalah 6V yang diambil langsung dari baterai.

Op-amp AD620 menggunakan tiga input voltase, yang terdiri dari pin +Vs, -Vs, dan REF. Nilai +Vs diambil langsung dari baterai, yaitu 6V. Nilai –Vs dihubungkan ke *ground*. Nilai REF diambil dari pembagi tegangan yang terdiri dari dua resistor 330Ω yang mengeluarkan voltase 3V. Nilai resistansi yang terhubung di kedua pin RG pada AD620 menentukan nilai *gain*. Nilai *gain* yang didapat dari 165Ω (dua buah 330Ω yang diparalel), adalah 49400/165+1=300.39394 kali. Nilai resistansi tersebut diambil dari hasil percobaan pembacaan ADC. *Gain* tidak terlalu besar sehingga nilai

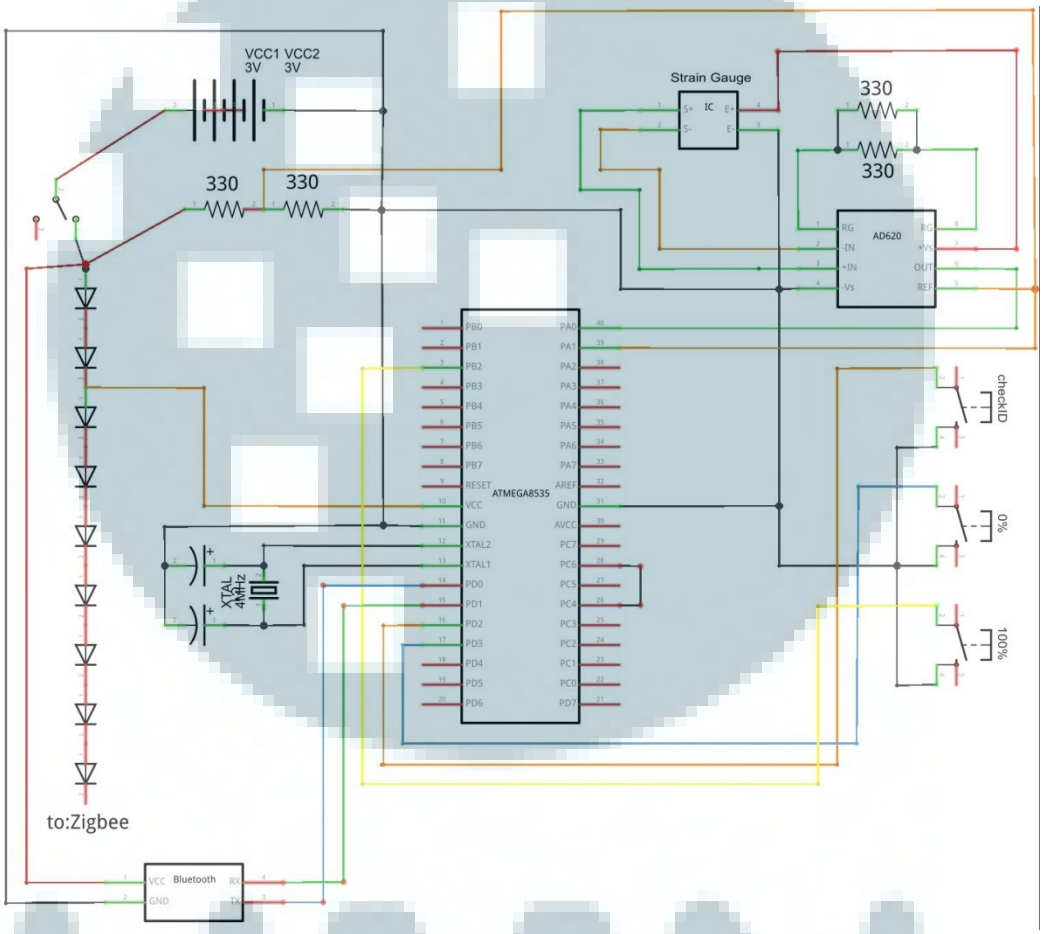
keluaran *op-amp* saat beban 7 kg masih dapat terbaca (nilai voltase keluaran tidak sama dengan nilai voltase keluaran saat beban di bawah 7 kg). *Gain* juga tidak terlalu kecil sehingga voltase keluaran masih dapat terbaca saat beban tidak banyak.

Karena voltase REF pada *op-amp* tidak sama dengan GND ATMega8535, ADC pada ATMega8535 menggunakan mode yang memiliki input dua buah yaitu REF dari keluaran *op-amp* dan keluaran *op-amp* itu sendiri. Keluaran *op-amp* terhubung ke ADC0 yang berada di pin PA0 pada ATMega8535. Voltase REF dihubungkan ke ADC1 yang berada di pin PA1.

Ketiga tombol input terhubung ke pin *external interrupt* pada ATMega8535. Tombol checkID terpasang di pin PD2 yang merupakan *external interrupt 0*. Tombol 0% terpasang di pin PD3 yang merupakan *external interrupt 1*. Tombol 100% terpasang di pin PB2 yang merupakan *external interrupt 2*. Pin-pin tersebut memiliki resistor *pull-up* yang berada di dalam *chip* sehingga normalnya pin-pin tersebut bernilai *high*. Saat tombol ditekan, nilai pin yang terhubung dengan tombol yang bersangkutan akan menjadi *low* karena terhubung dengan *ground*. *External interrupt* akan ter-*trigger* saat ada perubahan nilai dari *high* ke *low*.

Berdasarkan Gambar 3.1, saat menggunakan mode *bluetooth*, pin PC4 dan PC6 disambungkan dengan *jumper*. PC4 membaca nilai PC6 yang selalu bernilai 0. Modul *bluetooth* dapat menerima voltase input dari

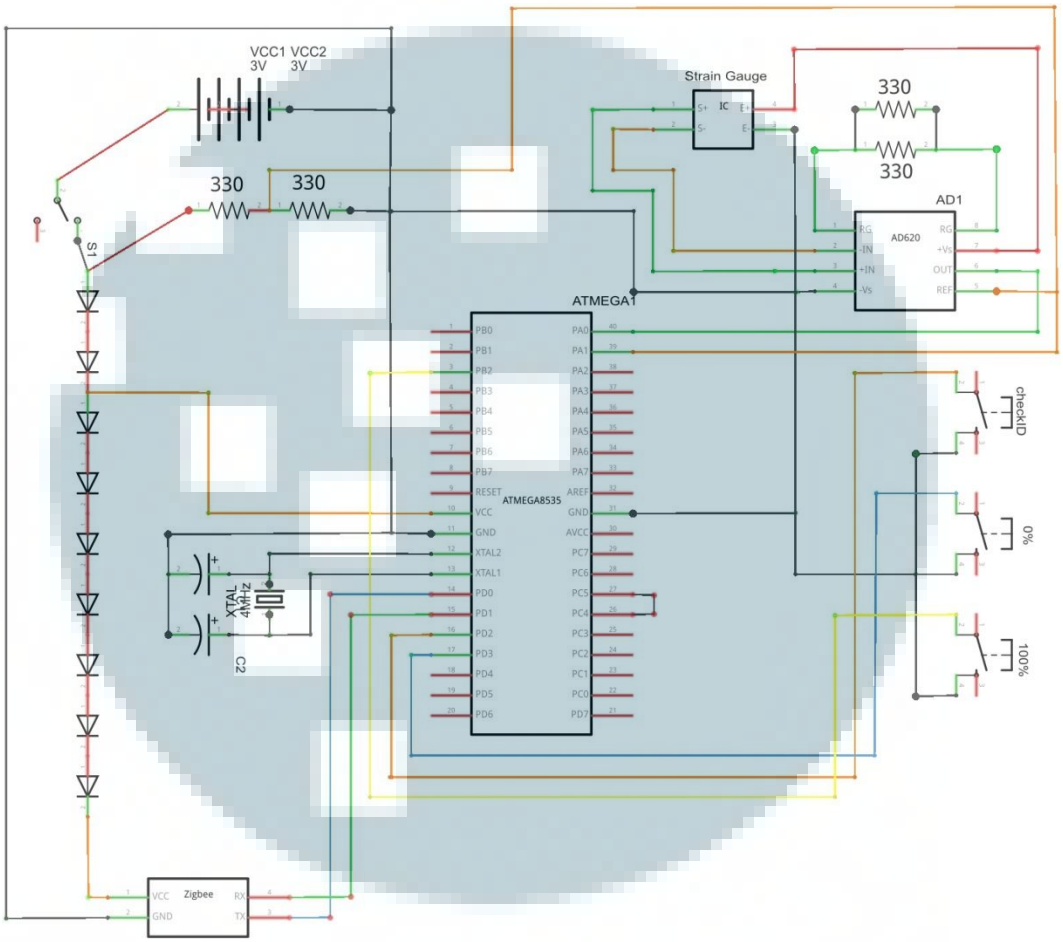
3.6V sampai 6V. Pada skematik, modul *bluetooth* diberikan voltase 6V langsung dari baterai. RX pada modul *bluetooth* disambungkan ke TX pada ATMega8535 yang berada di pin PD0. TX modul *bluetooth* disambungkan ke RX pada ATMega8535 yang berada di pin PD1.



Gambar 3.1 Skematik Wadah dengan *Bluetooth*

Berdasarkan Gambar 3.2, saat menggunakan mode *zigbee*, pin PC4 dan PC5 disambungkan dengan *jumper*. PC4 membaca PC5 yang selalu diset 1. Modul *zigbee* dapat menerima voltase input dari 2.1V sampai 3.6V. Modul *zigbee* mengambil voltase input dari dioda yang diseri sebanyak 9 buah sehingga nilai voltase turun dari 6V (voltase baterai) menjadi 6V-

9*0.3V=3.3V. Pin RX dan TX pada modul *zigbee* juga terhubung ke ATMega8535 seperti pada modul *bluetooth*.



Gambar 3.2 Skematik Wadah dengan *Zigbee*

3.3. Protokol Komunikasi Lokal

Komunikasi lokal adalah komunikasi antara *device* dan komputer *gateway* atau antar *device* yang berada di satu rumah yang sama.

3.3.1. Protokol Umum

3.3.1.1. Tipe *Device*

Terdiri dari dua huruf kapital berpasangan. Komputer *gateway* menggunakan tipe *device* ZZ. Wadah pintar menggunakan tipe *device* FS.

3.3.1.2. Nomor *Device*

Untuk menentukan nomor seri suatu *device* yang sama. Terdiri dari 3 digit heksadesimal (000 – FFF). Misalnya lampu 1 000, lampu 2 00A, lampu 3 00F.

3.3.1.3. Device ID

Tipe dan Nomor Device akan digabung sebagai suatu identifier bagi suatu device. Misalnya kulkas adalah BZ 000, lampu 1 AB 000, lampu 3 AB 00F.

3.3.2. Protokol Komunikasi Lokal dengan *Bluetooth*

3.3.2.1. Menambahkan/mendaftarkan wadah baru ke *database*

a. COM port dimasukkan manual dari aplikasi di komputer *gateway*.
b. ID modul *bluetooth* wadah: username: HC-05; *password*: 1234
c. Komputer *gateway* memberikan *device ID* komputer *gateway* ke *device* baru (lewat COM port yang sudah diinput)

   Format:

GateID#[*Device ID* komputer *gateway*]#<CR><LF>

   Contoh:

   GateID#ZZ 001#<CR>

d. *Device* baru membalas dengan *device ID*-nya

Format:

DeviceID#[*Device ID*]#<CR><LF>

Contoh:

DeviceID#FS 001#<CR>

3.3.2.2. *Liveness Update*

a. Komputer *gateway* mengirimkan “PING<CR><LF>”

b. *Device* akan mengirimkan “PING ACK<CR><LF>”

c. Jika *timeout*, bisa dicoba 3x sampai wadah dinyatakan mati

3.3.2.3. Mengirimkan *Update* ke Komputer *Gateway*

a. *Device* mengirimkan *update_packet* ke *gateway*

Tabel 3.1 Format *Update_Packet Bluetooth* Wadah

Fungsi	Data_Type	Format dan Contoh Update_Packet
***Update* Persentase**	Percent	Format: [Device ID]#percent#[persentase dalam desimal]#<CR><LF> Contoh: FS 001#percent#55#<CR><LF>
***Update* Umur**	Age	Format:

dalam Satuan Hari		[Device ID]#age#[umur (hari) dalam desimal]#<CR><LF> Contoh: FS 001#age#5#<CR><LF>
***Reset* Umur dan Persentase (Saat Tombol 100% Ditekan)**	Reset	Format: [Device ID]#reset#1<CR><LF> Contoh: FS 001#reset#1#<CR><LF>
Menampilkan *device ID* yang bersangkutan lewat *popup* (Saat tombol checkID ditekan)	showID	Format: [Device ID]#showID#1#<CR><LF> Contoh: FS 001#showID#1#<CR><LF>

b. *Gateway* membalas dengan ACK

Format ACK:

ACK#[Data_Type]#<CR><LF>

Contoh:

ACK#percent#<CR>

3.3.2.4. Komputer *Gateway* Mengganti Pengaturan Wadah

a. *Gateway* mengirimkan setting_packet ke device.

i. Wadah

Tabel 3.2 Format *Setting_Packet Bluetooth* Wadah

Fungsi	Data_Type	Format dan Contoh Setting_Packet
Mengganti *setting* frekuensi cek dan *update* (jika ada perubahan nilai) persentase stok tersisa. Satuan: menit (default: 5 menit)	freq-percent	Format: SETTING#freq-percent#[frekuensi dalam menit]#<CR><LF> Contoh: SETTING#freq-percent#10#<CR><LF>
Mengganti setting frekuensi update umur. Satuan: hari (default: 1 hari)	freq-age	Format: SETTING#freq-age#[frekuensi dalam hari]#<CR><LF> Contoh: SETTING#freq-age#2#<CR><LF>

b. *Device* mengatur *setting* sesuai dengan permintaan *gateway* lalu memberikan ACK.

Format ACK yang diberikan *device*:

ACK-SETTING<CR><LF>

Contoh:

ACK-SETTING<CR><LF>

3.3.3. Protokol Komunikasi Lokal dengan *Zigbee*

3.3.3.1. Menambahkan/mendaftarkan wadah baru ke *database*

a. *Gateway* mengirimkan introduction_message ke semua device

Gateway mengirimkan *introduction_message* dengan format:

at+annce<CR>

b. Menerima *introduction_message* dari *gateway*

Format yang diterima:

<CR><LF>FFD:[EUI-64 *gateway*],[*Node ID gateway*]<CR><LF>

Contoh:

<CR><LF>FFD:000D6F0002382C14,0000<CR><LF>

Catatan:

*jika status pada *device* sudah bernilai 1 (sudah ditambahkan), maka pesan yang diterima akan diabaikan

**device* baru akan mengubah status menjadi 1 dan lanjut ke langkah berikutnya

c. Mengirimkan alamat EUI-64-nya dan *Device* ID-nya ke *gateway*.

Format pengiriman ke *gateway*:

at+ucast:[EUI-64 *gateway*]=DeviceID#[*Device ID*]#<CR>

Contoh format pengiriman:

at+ucast:000d6f0002382c14=DeviceID#FS 001#<CR>

d. *Gateway* menerima EUI-64 dan Device ID dari device baru

Gateway akan menerima dengan format:

<CR><LF>UCAST:[EUI-64 *device*],10=DeviceID#[*Device ID*]#<CR><LF>

Contoh format yang diterima *gateway*:

<CR><LF>UCAST:000D6F00023832D3,10=DeviceID#FS 001#<CR><LF>

Catatan:

*data yang didapat dimasukkan ke *database*

e. *Gateway* mengirimkan *device ID* komputer *gateway* ke *device* baru

Gateway akan mengirimkan dengan format:

at+ucast:[EUI-64 *device*]=GateID#[*Device ID gateway*]#<CR>

Contoh:

at+ucast:000D6F00023832D3=GateID#ZZ 001#<CR>

f. *Device* baru akan mendapatkan GateID dari *gateway*

Format yang diterima oleh *device*

<CR><LF> UCAST:[EUI-64 *gateway*],0E=GateID#[Device ID *gateway*]#<CR><LF>

Contoh:

<CR><LF> UCAST:000d6f0002382c14,0E=GateID#ZZ 001#<CR><LF>

3.3.3.2. *Liveness Update*

a. *Komputer gateway* mengirimkan *Liveness Update* ke device untuk mengetahui apakah device masih hidup

Format pengiriman *Liveness Update*:

at+ucast:[EUI-64 *device* yang ingin diketahui *liveness*-nya]=<CR>

Contoh:

at+ucast:000d6f00023832d3=<CR>

Jika berhasil, contoh balasan dari modul:

<CR><LF>SEQ:*AD*<CR><LF>

<CR><LF>OK<CR><LF>

<CR><LF>**ACK**:*AD*<CR><LF>

Jika tidak berhasil, contoh balasan dari modul:

<CR><LF>SEQ:*AB*<CR><LF>

<CR><LF>OK<CR><LF>

<CR><LF>**NACK**:*AB*<CR><LF>

**gateway* tidak mendapatkan balasan dari *device*. Untuk mengetahui, dapat dilihat dari ACK atau NACK.

b. *Device* menerima *Liveness Update* dari *gateway*

Format data yang diterima:

<CR><LF>UCAST:[EUI-64 *gateway*],01=<CR><LF>

Contoh format data yang diterima:

UCAST:000D6F0002382C14,01=<CR>

CATATAN:

*Data yang diterima tidak diproses oleh *device*

*"*Gateway*" dapat diganti dengan "*device* lain"

3.3.3.3. Mengirimkan *Update* ke *Komputer Gateway*

a. Update_packet ditransmisikan oleh *device* ke *gateway*

i. Wadah

Tabel 3.3 Format *Update_Packet Zigbee* Wadah

Fungsi	Data_Type	Format dan Contoh Update_Packet
Update **Persentase**	Percent	Format: at+ucast:[EUI-64 *gateway*]=[*Device ID*]#percent#[persentase dalam desimal]#<CR> Contoh: at+ucast:000D6F0002382C14=FS001#percent#55#<CR>
Update **Umur dalam Satuan Hari**	Age	Format: at+ucast:[EUI-64 *gateway*]=[*Device ID*]#age#[umur (hari) dalam desimal]#<CR> Contoh:

		at+ucast:000D6F0002382C14=FS 001#age#5#<CR>
***Reset* Umur dan Persentase (Saat Tombol 100% Ditekan)**	Reset	Format: at+ucast:[EUI-64 *gateway*]=[*Device ID*]#reset#1#<CR> Contoh: at+ucast:000D6F0002382C14=FS 001#reset#1#<CR>
Menampilkan *device ID* yang bersangkutan lewat *popup* (Saat tombol *checkID* ditekan)	showID	Format: at+ucast:[EUI-64 *gateway*]=[*Device ID*]#showID#1#<CR> Contoh: at+ucast:000D6F0002382C14=FS 001#showID#1#<CR>

ii. Kulkas [**31**]

Tabel 3.4 Format *Update_Packet Zigbee* Kulkas

Fungsi	Data_Type	Format dan Contoh Update_Packet
Update Jumlah Telur dan	stok-suhu	Format: at+ucast:[EUI-64 core server]=[Device

Suhu di Kulkas		ID]#stok-suhu#[stok dalam desimal-suhu dalam desimal]#<CR> Contoh: at+ucast:000D6F0002382C14=RF001#stok-suhu#7-9#<CR>

b. *Gateway* menerima *update_packet*

Format penerimaan:

<CR><LF>UCAST:[EUI-64 *device*],[DataLength]=[*Device ID*]#[Data_Type]#[*value*]#<CR><LF>

Contoh:

<CR><LF>UCAST:000D6F00023832D3,12=FS001#percent#55#<CR><LF>

Komputer *gateway* menyimpan sementara EUI-64 *device* untuk mengirimkan ACK.

c. *Gateway* membalas dengan ACK

Format ACK:

at+ucast:[EUI-64 *device*]=ACK#[Data_Type]#<CR>

Contoh:

at+ucast:000D6F00023832D3=ACK#percent#<CR>

d. *Device* menerima ACK

Format ACK yang diterima:

<CR><LF>UCAST:[EUI-64 *gateway*],[DataLength]=ACK#[Data_Type]#<CR><LF>

Contoh:

<CR><LF>UCAST:000D6F0002382C14,0C=ACK#percent#<CR><LF>

3.3.3.4. Request Data dari *Device* A ke *Device* B

a. *Device* A langsung mengirimkan request_packet ke *device* B untuk mengambil data yang ada di *device* B.

i. Device B: Komputer *Gateway*

Tabel 3.5 Format *Request_Packet Zigbee* Komputer *Gateway*

Fungsi	Data_Type	Format dan Contoh Request_Packet
Meminta alamat EUI-64 wadah	address-FS-[nama isi wadah]	Format: at+ucast:[EUI-64 *gateway*]=[*Device ID gateway*]#address-FS-[nama isi wadah]#[alamat EUI-64 wadah]<CR>

		Contoh: at+ucast:000D6F0002382C14=ZZ 001#address-FS-gula#<CR>
Meminta alamat EUI-64 device lain	address-[Tipe Device]	Format: at+ucast:[EUI-64 wadah]=[*Device ID gateway*]#address-[Tipe *Device*]# <CR> Contoh: at+ucast:000D6F0002382C14=ZZ 001#address-AA#<CR>

ii. Device B: Wadah

Tabel 3.6 Format *Request_Packet Zigbee* Wadah

Fungsi	Data_Type	Format dan Contoh Request_Packet
Meminta nilai persentase isi wadah	Percent	Format: at+ucast:[EUI-64 wadah]=percent#<CR> Contoh: at+ucast:000D6F00023832D3=perc

		ent#<CR>
Meminta nilai umur dalam satuan hari	Age	Format: at+ucast:[EUI-64 wadah]=age#<CR> Contoh: at+ucast:000D6F00023832D3=age#<CR>

b. *Device* B menerima *request_packet* dari *device* A

Format penerimaan:

<CR><LF>UCAST:[EUI-64 *device* A],[DataLength]=[*Data_Type*]#<CR><LF>

Device B menyimpan sementara EUI-64 *Device* A untuk pengiriman update_packet.

c. *Device* B mengirimkan update_packet ke *device* A sesuai dengan data yang diminta.

Langkah c-f sama dengan bagian mengirimkan Update ke Komputer *Gateway* langkah a-d dengan menganggap komputer *gateway* (di bagian tersebut) sebagai *device* A

Update_Packet jika *device* B adalah komputer *gateway*:

Tabel 3.7 Format *Update_Packet Zigbee* Komputer *Gateway*

Fungsi	Data_Type	Format dan Contoh Update_Packet
Memberikan alamat EUI-64 wadah	address-FS-[nama isi wadah]	Format: at+ucast:[EUI-64 *device* A]=[*Device ID gateway*]address-FS-[nama isi wadah]# [EUI-64 wadah yang dicari alamatnya]#<CR> Contoh: at+ucast:000D6F0002382C14=ZZ001#address-FS-gula#000ABC0001232321<CR> Jika, EUI-64 yang dicari tidak terdaftar di *database*, maka nilai tersebut akan diisi dengan “0” (tanpa tanda petik)
Meminta alamat EUI-64 *device* lain	address-[Tipe *Device*]	Format: at+ucast:[EUI-64 *device* A]=[*Device* ID *gateway*]#address-[Tipe *Device*]# [EUI-64 *device* yang dicari alamatnya]<CR>

		Contoh: at+ucast:000D6F0002382C14=ZZ 001#address- AA#000ABC0E01232CBA<CR> Jika, EUI-64 yang dicari tidak terdaftar di *database*, maka nilai tersebut akan diisi dengan “0” (tanpa tanda petik)

d. *Device* A menerima update_packet dari *device* B

e. *Device* A membalas dengan ACK ke *device* B

f. *Device* B menerima pesan ACK

3.3.3.5. Komputer *Gateway* Mengganti Pengaturan Wadah

a. *Gateway* mengirimkan setting_packet ke *device*.

i. Wadah

Tabel 3.8 Format *Setting_Packet Zigbee* Wadah

Fungsi	Data_Type	Format dan Contoh Setting_Packet
Mengganti *setting* frekuensi cek dan	freq-percent	Format: at+ucast:[EUI-64

update (jika ada perubahan nilai) persentase stok tersisa. **Satuan: menit (*default*: 5 menit)**		device]=SETTING#freq-percent#[frekuensi dalam menit]#<CR> Contoh: at+ucast:000D6F00023832D3=SETTING#freq-percent#10#<CR>
Mengganti *setting* frekuensi update umur. **Satuan: hari (*default*: 1 hari)**	freq-age	Format: at+ucast:[EUI-64 device]=SETTING#freq-age#[frekuensi dalam hari]#<CR> Contoh: at+ucast:000D6F00023832D3=SETTING#freq-age#2#<CR>

b. *Device* menerima setting_packet dari *gateway*

Format data yang diterima oleh *device*:

<CR><LF>UCAST:[EUI-64 *gateway*],[DataLength]=SETTING#[Data_Type]#[value]#<CR><LF>

Contoh:

<CR><LF>UCAST: 000D6F0002382C14,13=SETTING#freq-age#2#<CR><LF>

c. *Device* mengatur *setting* sesuai dengan permintaan *gateway* lalu memberikan ACK.

Format ACK yang diberikan *device*:

at+ucast:[EUI-64 *gateway*]=ACK-SETTING<CR>

Contoh:

at+ucast:000D6F0002382C14=ACK-SETTING<CR>

d. *Gateway* menerima ACK dari *device* lalu mengganti nilai di *database*

Format ACK yang diterima:

<CR><LF>UCAST:[EUI-64 device],[DataLength]= ACK-SETTING<CR><LF>

Contoh:

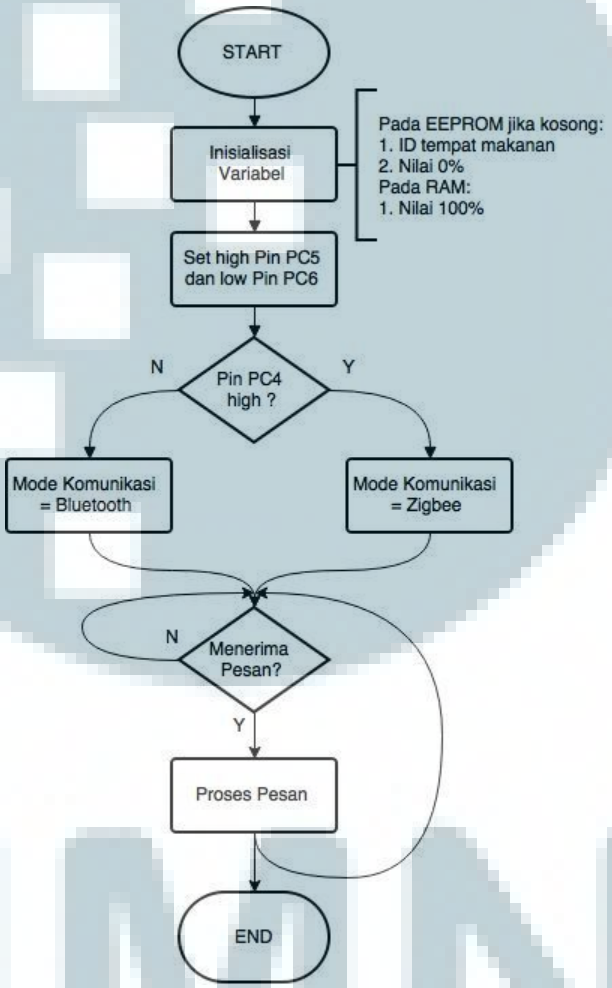
<CR><LF>UCAST: 000D6F00023832D3,0B=ACK-SETTING<CR><LF>

3.4. Perancangan *Software* Wadah

Program dibuat dengan bahasa C. Terdapat enam *interrupt* pada program, yaitu *external interrupt 0, external interrupt 1, external interrupt 2,*

timer 1 interrupt, timer 2 interrupt, dan *serial interrupt* (*receive interrupt*). Komunikasi dengan modul *bluetooth* atau modul *zigbee* menggunakan serial dengan *baud rate* 9600 *bps*.

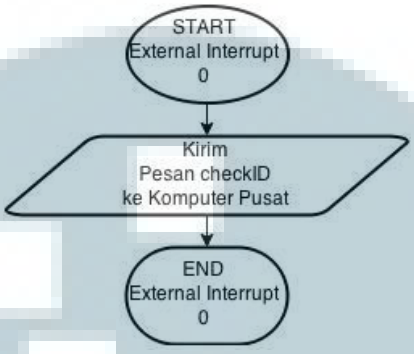
Flowchart program utama dapat dilihat pada Gambar 3.3. Program utama melakukan inisialisasi program, lalu melihat apakah menggunakan



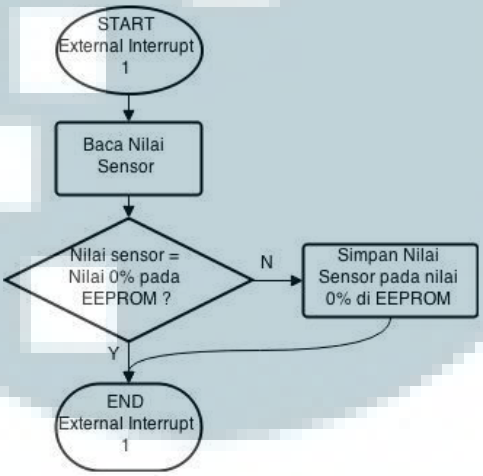
Gambar 3.3 *Flowchart Main Program* Wadah

mode *Bluetooth* atau *Zigbee* dengan melihat bacaan pin PC4, dan kemudian masuk ke dalam *loop* untuk selalu siap menerima pesan kecuali

pada saat *interrupt* dijalankan. Pesan yang diterima diproses sesuai dengan protokol komunikasi lokal pada subbab 3.3.

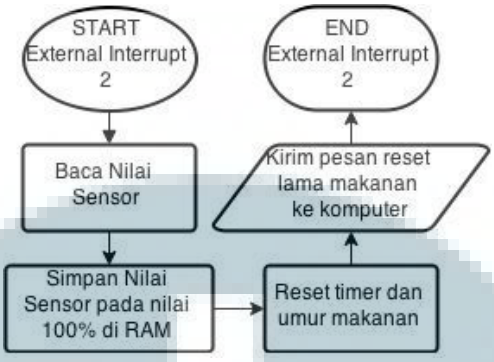


Gambar 3.4 *Flowchart External Interrupt 0* Wadah



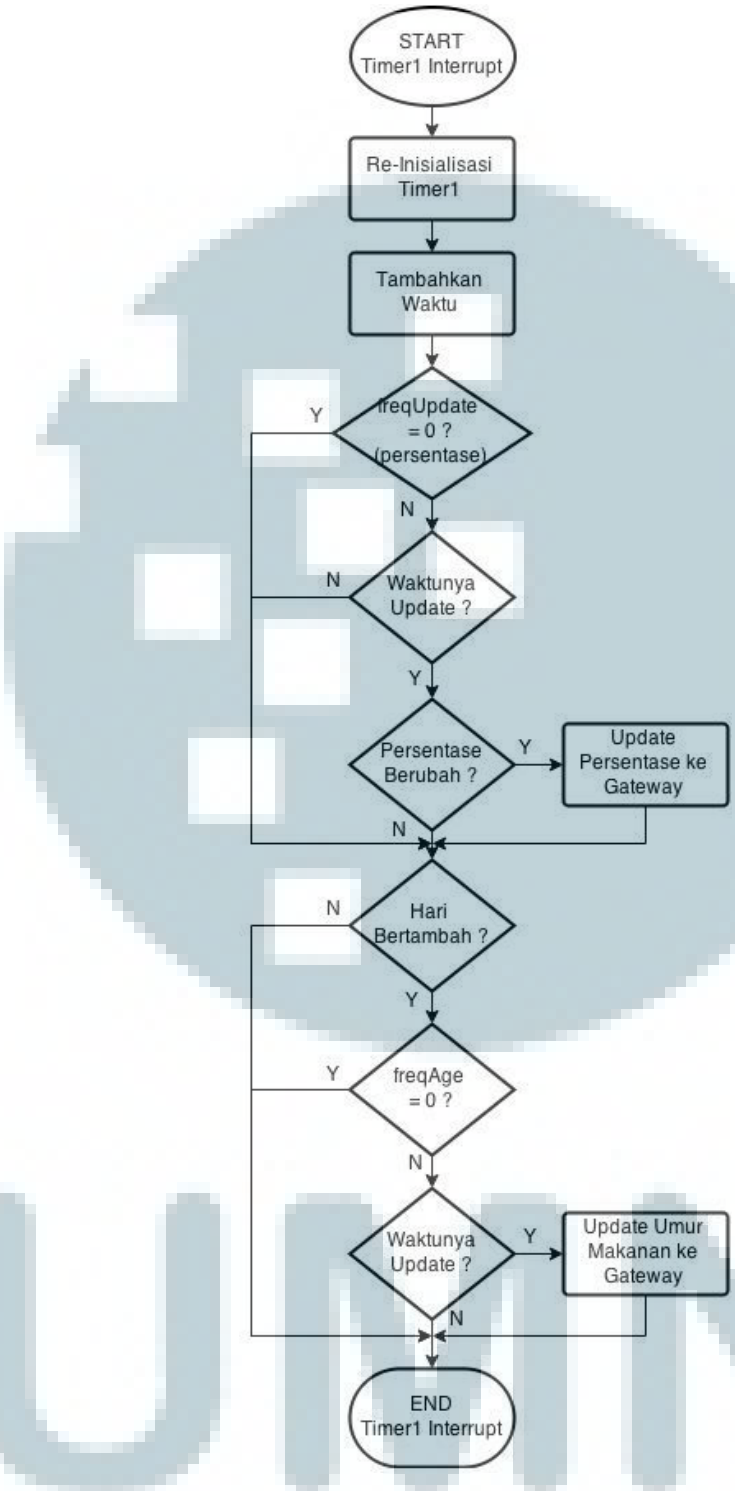
Gambar 3.5 *Flowchart External Interrupt 1* Wadah

External interrupt 0 dipakai untuk tombol *CheckID* memiliki *flowchart* yang dapat dilihat pada Gambar 3.4. *External interrupt* 1 dipakai untuk tombol 0% memiliki *flowchart* yang dapat dilihat pada Gambar 3.5. *External interrupt* 2 dipakai untuk tombol 100% memiliki *flowchart* yang dapat dilihat pada Gambar 3.6. Proses *update* ke *gateway* menyesuaikan dengan protokol komunikasi (*zigbee/bluetooth*) yang digunakan dan juga protokol komunikasi lokal pada subbab 3.3.



Gambar 3.6 *Flowchart External Interrupt 2* Wadah

Wadah juga menghitung umur isi wadah dalam unit hari menggunakan fungsi *timer*. Setiap penambahan jumlah hari, wadah akan mengecek apakah waktunya meng-*update* umur ke komputer *gateway* dilihat dari *freqAge*. Jika memang sudah waktunya, maka wadah akan meng-*update* umur ke komputer *gateway*. Umur yang tersimpan dan *timer 1* akan di-*reset* saat pengguna menekan tombol 100% seperti pada Gambar 3.6. Saat di-*reset*, wadah juga memberi tahu ke komputer *gateway* untuk me-reset umur di server. Sambil menghitung umur, setiap beberapa menit (tergantung pengaturan dari pengguna, default: 5 menit), wadah akan memeriksa nilai persentase berat isi wadah, apakah berubah dari nilai sebelumnya. Jika berubah, akan mengirimkan *update* ke komputer *gateway*. Perhitungan persentase stok yang dikirimkan menggunakan rumus selisih nilai ADC saat itu dengan nilai ADC saat 0% dibagi dengan selisih nilai ADC saat 100% dengan saat 0% kemudian dikalikan dengan 100%. Semua ini dilakukan dengan interrupt timer 1 seperti pada Gambar 3.7.



Gambar 3.7 *Flowchart Timer 1 Interrupt* Wadah

Penggunaan timer yang lain yaitu *timer* 2. *Timer* 2 hanya dipakai oleh fungsi menerima data dengan *timeout*. Lamanya *timeout* dapat lebih

akurat dengan penggunaan *timer*. Akan tetapi, *timeout* tidak dapat dipakai saat *external interrupt* karena *external interrupt* memiliki prioritas yang lebih tinggi dibandingkan dengan *timer interrupt*. *Serial interrupt* dipakai juga bersamaan dengan *timer 2* untuk penerimaan dengan *timeout* ini. Pada ATMega8535, prioritas *interrupt* tidak dapat diubah. Perhitungan nilai *timer* dapat dilihat pada program di LAMPIRAN 1. Program Wadah Pintar.

3.5. Perancangan *Database*

Database menggunakan *mySQL* yang ada di *cloud* dengan nama *database k3842690_umn-iot-taskforceDB*. Berikut adalah proses normalisasi *database* yang menyimpan informasi mengenai *username*, *device* secara umum, dan wadah (tempat makanan).

a. UNF (*Un-Normalized Form)*

   username + password + { deviceID + tipeDevice + EUI64 + statusAktif + namaMakanan + tglKadaluarsa + notifikasiStok + notifikasiKadaluarsa + updateUmur + persenSisaAkhir (mengambil nilai persentaseSisa yang paling baru) + umurMakanan + batasBawah + { waktu + persentaseSisa }}

b. 1NF (*1st Normalized Form*)

   username (kandidat *PK*/*Primary Key*) + password + deviceID (kandidat *PK*) + tipeDevice + EUI64 + statusAktif + namaMakanan +

tglKadaluarsa + notifikasiStok + notifikasiKadaluarsa + updateUmur + umurMakanan + batasBawah + waktu + persentaseSisa

c. 2NF (*2nd Normalized Form*)

User = username (redundan) + password (redundan) + deviceID + tipeDevice + EUI64 + statusAktif

TempatMakanan = deviceID (redundan) + namaMakanan (redundan) + tglKadaluarsa (redundan) + notifikasiStok (redundan) + notifikasiKadaluarsa (redundan) + updateUmur (redundan) + umurMakanan (redundan) + batasBawah (redundan) + waktu + persentaseSisa

d. 3NF (*3rd Normalized Form*)

User = username + password

Device = deviceID + userID + tipeDevice + EUI64 + statusAktif

TempatMakananHeader = deviceID + namaMakanan + tglKadaluarsa + notifikasiStok + notifikasiKadaluarsa + updateUmur + umurMakanan + batasBawah

TempatMakananDetail = waktu + persentaseSisa + deviceID

Tabel *User* dapat dilihat pada Tabel 3.9. Atribut *username* harus unik dan merupakan *primary key* dari tabel *User*. *Password* yang disimpan berupa *password* yang sudah di-*hash* dengan *SHA3-256.*

Tabel 3.9 Tabel *User*

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**username**	varchar(20)	latin1_swedish_ci		No	*None*	
2	**password**	varchar(64)	latin1_swedish_ci		Yes	*NULL*	

Tabel Device dapat dilihat pada Tabel 3.10. Atribut *deviceID* harus unik dan merupakan *primary key* dari tabel ini. Atribut *username* merupakan *foreign key* yang me-*refer username* pada tabel *User*.

Tabel 3.10 Tabel *Device*

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**deviceID**	varchar(6)	latin1_swedish_ci		No	*None*	
2	**username**	varchar(20)	latin1_swedish_ci		Yes	*NULL*	
3	**tipeDevice**	varchar(30)	latin1_swedish_ci		Yes	*NULL*	
4	**EUI64**	varchar(16)	latin1_swedish_ci		Yes	*NULL*	
5	**statusAktif**	tinyint(1)			No	1	

Tabel *TempatMakananHeader* dapat dilihat pada Tabel 3.11. Atribut *deviceID* merupakan *foreign key* yang me-*refer* ke *deviceID* pada tabel *Device*. Atribut *namaMakanan* dapat diisi dengan nama bahan lain yang tidak harus makanan. Atribut *tanggalKadaluarsa* menyimpan tanggal kadaluarsa dalam bentuk *epoch/UNIX time* dengan satuan *second*. Atribut notifikasiStok merupakan frekuensi pemeriksaan stok yang dilakukan oleh wadah dalam satuan menit. Jika persentase berubah, wadah akan *update* ke *database* melalui perantaraan *gateway* dan *server*. Selain *update*, persentase juga akan dibandingkan dengan *batasBawah*. Jika persentase lebih rendah, maka akan dilakukan notifikasi ke perangkat *Android* atau

perangkat lainnya yang bersangkutan. Notifikasi stok jika tidak diaktifkan akan diberi nilai 0. Atribut notifikasiKadaluarsa merupakan jumlah berapa hari sebelum kadaluarsa perangkat *Android* atau perangkat lainnya akan mendapatkan notifikasi. Jika notifikasi dimatikan, maka akan diberi nilai -1. Atribut updateUmur merupakan frekuensi wadah meng-*update* umur ke *database* dalam satuan hari. Jika notifikasi dimatikan, nilai atribut tersebut diberi nilai 0.

Tabel 3.11 Tabel *TempatMakananHeader*

#	Name	Type	Collation	Attributes	Null	Default
1	**deviceID**	varchar(6)	latin1_swedish_ci		No	*None*
2	**notifikasiStok**	int(11)			No	5
3	**notifikasiKadaluarsa**	int(11)			No	2
4	**namaMakanan**	varchar(30)	latin1_swedish_ci		No	*None*
5	**tanggalKadaluarsa**	bigint(20)			Yes	*NULL*
6	**umurMakanan**	int(11)			No	0
7	**updateUmur**	int(11)			No	1
8	**batasBawah**	int(11)			No	20

Tabel *TempatMakananDetail* berisi *logger* persentase isi wadah dari waktu ke waktu. Waktu disimpan dalam bentuk *epoch/UNIX* time dengan satuan *second*. Atribut deviceID merupakan *foreign key* terhadap *deviceID* di tabel *Device*. Atribut dapat dilihat pada Tabel 3.12.

Tabel 3.12 Tabel *TempatMakananDetail*

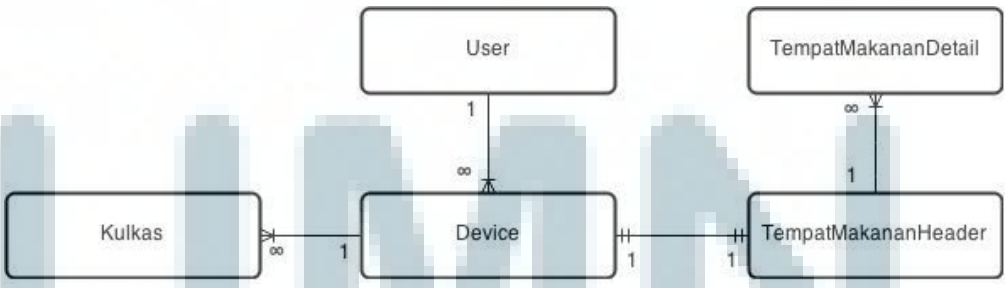
#	Name	Type	Collation	Attributes	Null	Default
1	**deviceID**	varchar(6)	latin1_swedish_ci		Yes	*NULL*
2	**waktu**	bigint(20)			Yes	*NULL*
3	**persentaseStok**	int(11)			Yes	100

Selain tabel untuk wadah (tempat makanan), untuk saat ini juga ada tabel untuk kulkas yaitu tabel *Kulkas* pada Tabel 3.13 [**31**]. Waktu yang disimpan juga berupa *epoch/UNIX time* dengan satuan *second*. Tabel Kulkas tidak dibahas lebih lanjut.

Tabel 3.13 Tabel *Kulkas* [31]

#	Name	Type	Collation	Attributes	Null	Default
1	**deviceID**	varchar(6)	latin1_swedish_ci		Yes	*NULL*
2	**waktu**	bigint(20)			Yes	*NULL*
3	**stok**	int(11)			Yes	*NULL*
4	**suhu**	int(11)			Yes	*NULL*

Saat ini, *database* memiliki *Entity Relationship Model* (*ER Model*) seperti pada Gambar 3.8.



Gambar 3.8 *ER Model Database*

Terdapat juga *buffer* untuk menampung *command* dari *Android device* ke *gateway*. *Buffer* disimpan dalam tabel tersendiri yang tidak berhubungan dengan tabel yang lain yaitu tabel *CommandBuffer*. Tabel

CommandBuffer pada Tabel 3.14 memiliki atribut *heartbeatID* yang dimulai dari angka 1 dan terus naik per *gatewayID* sehingga nilai *heartbeatID* tidak unik, *gatewayID* yang merupakan *deviceID* dari *gateway* yang dituju, *command*, dan *parameter* yang antar parameternya dipisahkan oleh pagar. Setelah *command* diambil oleh *gateway* akan dihapus dari *database*.

Tabel 3.14 Tabel *CommandBuffer*

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**heartbeatID**	int(11)			No	*None*	
2	**gatewayID**	varchar(6)	latin1_swedish_ci		No	*None*	
3	**command**	varchar(20)	latin1_swedish_ci		No	*None*	
4	**parameter**	varchar(100)	latin1_swedish_ci		No	*None*	

3.6. Perancangan Komunikasi antara *Gateway* dan *Server*

Gateway dan *server* berkomunikasi dengan protokol *XML-RPC.* Untuk fungsi-fungsi yang khusus untuk satu jenis device, terdapat tambahan di depan nama fungsi yang dipisahkan dengan tanda titik yang menandakan tipe *device* seperti pada subbab 3.3.1. Berikut adalah fungsi-fungsi di *server* yang dapat dipanggil oleh *gateway*.

1. **addUsername** : fungsi untuk menambah *username* baru pada *database*.

   Gateway mengirimkan *XML-RPC* dengan *methodName* = *addUsername* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.15.

Tabel 3.15 Struktur *Struct* Fungsi *addUsername Request*

Key	**Tipe Data**	**Keterangan**
USERNAME	String 20	*Username* baru yang akan ditambahkan ke *database* yang sebelumnya dicek oleh *gateway* apakah *username* tersebut sudah ada atau belum dengan findUsername.
PASSWORD	String 64	*Password* dari *username* yang ditambahkan yang telah di-*hash* dengan *SHA3-256*.

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah struct dengan nilai seperti pada Tabel 3.16.

Tabel 3.16 Struktur *Struct* Fungsi *addUsername Response*

Key	**Tipe Data**	**Keterangan**
RESPONSECODE	String	'00' : *username* baru berhasil ditambahkan. '01' : gagal.
MESSAGE	String	Berisi "*Succeed*" jika berhasil. Berisi pesan *error* jika gagal.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>addUsername</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
```

```
                <name>USERNAME</name>
                <value><string>tralala</string></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>RESPONSECODE</name>
                <value><string>00</string></value>
        </member>
        <member>
                <name>MESSAGE</name>
                <value><string>Succeed</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

2. **findUsername** : fungsi untuk mengetahui apakah *username* yang bersangkutan ada di dalam *database*. Fungsi mengembalikan *username* dan *password*-nya jika ada di *database*.

Gateway mengirimkan *XML-RPC* dengan *methodName* = *findUsername* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.17.

Tabel 3.17 Struktur *Struct* Fungsi *findUsername Request*

Key	**Tipe Data**	**Keterangan**
USERNAME	String 20	*Username* yang ingin diketahui keberadaannya atau ingin diketahui *password*-nya jika ada di *database*.

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah struct dengan nilai seperti pada Tabel 3.18.

Tabel 3.18 Struktur *Struct* Fungsi *findUsername Response*

Key	**Tipe Data**	**Keterangan**
RESPONSECODE	String	‘00’ : *username* tidak ada di *database*. ’01’ : *username* ada di *database*.
MESSAGE	String	Berisi *password* jika *username* ada di *database*. Berisi “*username not exist*” jika *username* tidak ada di *database*. *Password* yang dikirimkan berupa hasil *hash* dengan *SHA3-256.*

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>findUsername</methodName>
```

```
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>USERNAME</name>
                <value><string>tralala</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>RESPONSECODE</name>
                <value><string>01</string></value>
        </member>
        <member>
                <name>MESSAGE</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

3. **addDevice** : fungsi untuk menambah *device* baru pada tabel *Device*.

Gateway mengirimkan *XML-RPC* dengan *methodName = addDevice* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.19.

Tabel 3.19 Struktur *Struct* Fungsi *addDevice Request*

Key	**Tipe Data**	**Keterangan**
DEVICEID	String 6	Nomor *deviceID* baru yang ingin ditambahkan ke *database*.
USERNAME	String 20	*Username* yang menambahkan *device* baru.
TIPEDEVICE	String 30	Tipe *device* yang ditambahkan. Contoh: "kulkas"
EUI64	String 16	Nilai EUI64 modul *zigbee device* baru. Jika menggunakan *bluetooth*, nilai ini berisi *COM Port*, contoh: "COM21".
PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah struct dengan nilai seperti pada Tabel 3.20.

Tabel 3.20 Struktur *Struct* Fungsi *addDevice Response*

Key	**Tipe Data**	**Keterangan**
RESPONSECODE	String	'00' : *device* baru berhasil ditambahkan. '01' : gagal.
MESSAGE	String	Berisi "*Succeed*" jika berhasil. Berisi pesan *error* jika gagal.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
```

```
<methodName>addDevice</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>DEVICEID</name>
                <value><string>FS 001</string></value>
        </member>
        <member>
                <name>USERNAME</name>
                <value><string>tralala</string></value>
        </member>
        <member>
                <name>TIPEDEVICE</name>
                <value><string>tempat makanan</string></value>
        </member>
        <member>
                <name>EUI64</name>
                <value><string>000D6F00023832D3</string></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>RESPONSECODE</name>
                <value><string>00</string></value>
        </member>
        <member>
                <name>MESSAGE</name>
                <value><string>Succeed</string></value>
```

```
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

4. **fs.getListOfFoodStorageDevice** : fungsi untuk mengambil nama makanan atau bahan lainnya dan deviceID dari semua wadah yang terdaftar di *database* atas nama *username* yang dikirimkan oleh *gateway*.

Gateway mengirimkan *XML-RPC* dengan *methodName* = *fs.getListOfFoodStorageDevice* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.21.

Tabel 3.21 Struktur *Struct* Fungsi fs.*getListOfFoodStorageDevice Request*

Key	**Tipe Data**	**Keterangan**
USERNAME	String 20	*Username* yang menambahkan *device* baru.
PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah *array of struct* dengan nilai seperti pada Tabel 3.22.

Tabel 3.22 Struktur *Struct* (dalam *Array*) Fungsi fs.*getListOfFoodStorageDevice Response*

Key	**Tipe Data**	**Keterangan**
NAMAMAKANAN	String 30	Nama makanan atau bahan lainnya.
DEVICEID	String 20	*Device ID* yang berisi nama makanan

		atau bahan lainnya di atas.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>fs.getListOfFoodStorageDevice</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>USERNAME</name>
                <value><string>tralala</string></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
        <array>
        <data>
        <value>
                <struct>
                <member>
                        <name>NAMAMAKANAN</name>
                        <value><string>Cokelat</string></value>
                </member>
                <member>
                        <name>DEVICEID</name>
                        <value><string>FS 001</string></value>
                </member>
```

```
                </struct>
        </value>
        <value>
                <struct>
                <member>
                        <name>NAMAMAKANAN</name>
                        <value><string>Beras</string></value>
                </member>
                <member>
                        <name>DEVICEID</name>
                        <value><string>FS 002</string></value>
                </member>
                </struct>
        </value>
        </data>
        </array>
</value>
        </param>
        </params>
</methodResponse>
```

5. **changePassword** : fungsi untuk mengganti *password* dari *username* yang sudah ada di *database*.

   Gateway mengirimkan *XML-RPC* dengan *methodName* = *changePassword* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.23.

Tabel 3.23 Struktur *Struct* Fungsi *changePassword Request*

Key	**Tipe Data**	**Keterangan**
USERNAME	String 20	*Username* yang akan diganti *password*-nya.
PASSWORD	String 64	*Password* baru yang sudah di-*hash* dengan *SHA3-256*.
OLDPASS	String 64	Password lama yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah struct dengan nilai seperti pada Tabel 3.24.

Tabel 3.24 Struktur *Struct* Fungsi *changePassword Response*

Key	**Tipe Data**	**Keterangan**
RESPONSECODE	String	'00' : *password* berhasil diganti. '01' : gagal.
MESSAGE	String	Berisi "*Succeed*" jika berhasil. Berisi pesan *error* jika gagal.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>changePassword</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>USERNAME</name>
                <value><string>tralala</string></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>b0cd784942edf7b37f3f6ac1a94278aabff022123f9e
a1899bf7a3813ff93a7c</string></value>
        </member>
        <member>
                <name>OLDPASS</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>RESPONSECODE</name>
                <value><string>00</string></value>
        </member>
        <member>
                <name>MESSAGE</name>
                <value><string>Succeed</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

6. **deleteDevice** : fungsi untuk menghapus *device* dengan *device ID* yang dikirim dari *gateway* dari tabel *Device*.

   Gateway mengirimkan *XML-RPC* dengan *methodName* = *deleteDevice* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.25.

Tabel 3.25 Struktur *Struct* Fungsi *deleteDevice Request*

Key	**Tipe Data**	**Keterangan**
DEVICEID	String 6	*Device ID* beserta atribut-atribut lainnya yang ingin dihapus dari tabel *Device*.
PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah struct dengan nilai seperti pada Tabel 3.26.

Tabel 3.26 Struktur *Struct* Fungsi *deleteDevice Response*

Key	**Tipe Data**	**Keterangan**
RESPONSECODE	String	'00' : berhasil menghapus *device*. '01' : gagal menghapus.
MESSAGE	String	Berisi "*Succeed*" jika berhasil. Berisi pesan *error* jika gagal.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>deleteDevice</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>DEVICEID</name>
                <value><string>FS 001</string></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
```

```
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>RESPONSECODE</name>
                <value><string>00</string></value>
        </member>
        <member>
                <name>MESSAGE</name>
                <value><string>Succeed</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

7. **fs.deleteFoodStorageDevice** : fungsi untuk menghapus *device* wadah dari tabel *TempatMakananDetail* dan *TempatMakananHeader*.

   Gateway mengirimkan *XML-RPC* dengan *methodName* = *deleteFoodStorageDevice* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.27.

Tabel 3.27 Struktur *Struct* Fungsi *fs.deleteFoodStorageDevice Request*

Key	**Tipe Data**	**Keterangan**
DEVICEID	String 6	*Device ID* wadah beserta atribut-atribut lainnya yang ingin dihapus dari tabel *TempatMakananDetail* dan *TempatMakananHeader*.
PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-*

		256 untuk autentikasi

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah struct dengan nilai seperti pada Tabel 3.28.

Tabel 3.28 Struktur *Struct* Fungsi *fs.deleteFoodStorageDevice Response*

Key	**Tipe Data**	**Keterangan**
RESPONSECODE	String	'00' : berhasil menghapus *device*. '01' : gagal menghapus.
MESSAGE	String	Berisi "*Succeed*" jika berhasil. Berisi pesan *error* jika gagal.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>fs.deleteFoodStorageDevice</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>DEVICEID</name>
                <value><string>FS 001</string></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>RESPONSECODE</name>
                <value><string>00</string></value>
        </member>
        <member>
                <name>MESSAGE</name>
                <value><string>Succeed</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

8. **fs.addTempatMakanan** : fungsi menambahkan wadah baru ke tabel *TempatMakananHeader* dan *TempatMakananDetail*.

   Gateway mengirimkan *XML-RPC* dengan *methodName* = *fs.addTempatMakanan* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.29.

Tabel 3.29 Struktur *Struct* Fungsi *fs.addTempatMakanan Request*

Key	**Tipe Data**	**Keterangan**
DEVICEID	String 6	*Device ID* wadah baru.
NAMAMAKANAN	String 30	Nama makanan atau bahan lainnya yang disimpan.
KADALUARSA	String	Tanggal kadaluarsa. Dapat tidak diisi jika

		tidak perlu tanggal kadaluarsa. Merupakan *long epoch time seconds.*
PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah struct dengan nilai seperti pada Tabel 3.30.

Tabel 3.30 Struktur *Struct* Fungsi *fs.addTempatMakanan Response*

Key	**Tipe Data**	**Keterangan**
RESPONSECODE	String	‘00’ : berhasil menambah wadah baru. ’01’ : gagal menghapus.
MESSAGE	String	Berisi “*Succeed*” jika berhasil. Berisi pesan *error* jika gagal.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>fs.addTempatMakanan</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>DEVICEID</name>
                <value><string>FS 001</string></value>
        </member>
        <member>
                <name>NAMAMAKANAN</name>
                <value><string>Cokelat</string></value>
        </member>
        <member>
                <name>KADALUARSA</name>
                <value><string> 1432565117649</string></value>
```

```
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>RESPONSECODE</name>
                <value><string>00</string></value>
        </member>
        <member>
                <name>MESSAGE</name>
                <value><string>Succeed</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

9. **fs.changeFoodName** : fungsi untuk mengganti nama makanan atau bahan lainnya yang ada di tabel *TempatMakananHeader*.

   Gateway mengirimkan *XML-RPC* dengan *methodName* = *fs.changeFoodName* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.31.

Tabel 3.31 Struktur *Struct* Fungsi *fs.changeFoodName Request*

Key	**Tipe Data**	**Keterangan**
DEVICEID	String 6	*Device ID* wadah yang ingin diubah nama isi wadahnya.
NAMAMAKANAN	String 30	Nama isi wadah baru.
PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah struct dengan nilai seperti pada Tabel 3.32.

Tabel 3.32 Struktur *Struct* Fungsi *fs.changeFoodName Response*

Key	**Tipe Data**	**Keterangan**
RESPONSECODE	String	‘00’ : berhasil mengganti nama isi wadah. ’01’ : gagal mengganti.
MESSAGE	String	Berisi “*Succeed*” jika berhasil. Berisi pesan *error* jika gagal.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>fs.changeFoodName</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>DEVICEID</name>
                <value><string>FS 001</string></value>
```

```
        </member>
        <member>
                <name>NAMAMAKANAN</name>
                <value><string>Keripik</string></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>RESPONSECODE</name>
                <value><string>00</string></value>
        </member>
        <member>
                <name>MESSAGE</name>
                <value><string>Succeed</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

10. **fs.changeExpiredNotification** : fungsi untuk mengganti berapa hari sebelum tanggal kadaluarsa notifikasi dikirimkan oleh *server* pada tabel *TempatMakananHeader*.

Gateway mengirimkan *XML-RPC* dengan *methodName* = *fs.changeExpiredNotification* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.33.

Tabel 3.33 Struktur *Struct* Fungsi *fs.changeExpiredNotification Request*

Key	**Tipe Data**	**Keterangan**
DEVICEID	String 6	*Device ID* wadah yang ingin diubah notifikasi kadaluarsanya.
NOTIF	Int	Jumlah hari berapa lama sebelum kadaluarsa notifikasi dikirimkan.
PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah struct dengan nilai seperti pada Tabel 3.34.

Tabel 3.34 Struktur *Struct* Fungsi *fs.changeExpiredNotification Response*

Key	**Tipe Data**	**Keterangan**
RESPONSECODE	String	‘00’ : berhasil mengganti notifikasi kadaluarsa. ’01’ : gagal mengganti.
MESSAGE	String	Berisi “*Succeed*” jika berhasil. Berisi pesan *error* jika gagal.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
```

```
<methodCall>
<methodName>fs.changeExpiredNotification</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>DEVICEID</name>
                <value><string>FS 001</string></value>
        </member>
        <member>
                <name>NOTIF</name>
                <value><int>4</int></value>
        </member>
                <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>RESPONSECODE</name>
                <value><string>00</string></value>
        </member>
        <member>
                <name>MESSAGE</name>
                <value><string>Succeed</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

11. **fs.changeStockNotification** : fungsi untuk mengganti frekuensi pengecekan nilai stok (dalam menit), *update* nilai stok, dan pengecekan dengan batas bawah untuk notifikasi ke *device* pada tabel *TempatMakananHeader*.

Gateway mengirimkan *XML-RPC* dengan *methodName* = *fs.changeStockNotification* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.35.

Tabel 3.35 Struktur *Struct* Fungsi *fs.changeStockNotification Request*

Key	**Tipe Data**	**Keterangan**
DEVICEID	String 6	*Device ID* wadah yang ingin diubah notifikasi stoknya.
NOTIF	Int	Frekuensi baru notifikasi stok. Diisi 0 jika ingin mematikan notifkasi.
PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah struct dengan nilai seperti pada Tabel 3.36.

Tabel 3.36 Struktur *Struct* Fungsi *fs.changeStockNotification Response*

Key	**Tipe Data**	**Keterangan**
RESPONSECODE	String	‘00’ : berhasil mengganti notifikasi stok. ’01’ : gagal mengganti.
MESSAGE	String	Berisi “*Succeed*” jika berhasil. Berisi

		pesan *error* jika gagal.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>fs.changeStockNotification</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>DEVICEID</name>
                <value><string>FS 001</string></value>
        </member>
        <member>
                <name>NOTIF</name>
                <value><int>10</int></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>RESPONSECODE</name>
                <value><string>00</string></value>
        </member>
        <member>
                <name>MESSAGE</name>
                <value><string>Succeed</string></value>
```

```
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

12. **fs.changeAgeUpdate** : fungsi untuk mengganti frekuensi *update* umur dari wadah ke *gateway* yang ada di tabel *TempatMakananHeader* dalam satuan hari.

Gateway mengirimkan *XML-RPC* dengan *methodName* = *fs.changeAgeUpdate* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.37.

Tabel 3.37 Struktur *Struct* Fungsi fs.changeAgeUpdate *Request*

Key	**Tipe Data**	**Keterangan**
DEVICEID	String 6	*Device ID* wadah yang ingin diubah frekuensi *update* umurnya.
FREK	Int	Frekuensi baru *update umur*.
PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah struct dengan nilai seperti pada Tabel 3.38.

Tabel 3.38 Struktur *Struct* Fungsi *fs.changeAgeUpdate Response*

Key	**Tipe Data**	**Keterangan**

RESPONSECODE	String	'00' : berhasil mengganti frekuensi umur. '01' : gagal mengganti.
MESSAGE	String	Berisi "*Succeed*" jika berhasil. Berisi pesan *error* jika gagal.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>fs.changeAgeUpdate</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>DEVICEID</name>
                <value><string>FS 001</string></value>
        </member>
        <member>
                <name>FREK</name>
                <value><int>2</int></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
```

```
        <struct>
        <member>
                <name>RESPONSECODE</name>
                <value><string>00</string></value>
        </member>
        <member>
                <name>MESSAGE</name>
                <value><string>Succeed</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

13. **fs.changeLowerBound** : fungsi untuk mengganti batas bawah dalam persentase untuk notifikasi *low stock* ke *device* yang ada di tabel *TempatMakananHeader*.

Gateway mengirimkan *XML-RPC* dengan *methodName* = *fs.changeLowerBound* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.39.

Tabel 3.39 Struktur *Struct* Fungsi *fs.changeLowerBound Request*

Key	**Tipe Data**	**Keterangan**
DEVICEID	String 6	*Device ID* wadah yang ingin diubah batas bawahnya.
BOUND	Int	Batas bawah persentase stok baru.
PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah struct dengan nilai seperti pada Tabel 3.40.

Tabel 3.40 Struktur *Struct* Fungsi *fs.changeLowerBound Response*

Key	**Tipe Data**	**Keterangan**
RESPONSECODE	String	'00' : berhasil mengganti batas bawah. '01' : gagal mengganti.
MESSAGE	String	Berisi "*Succeed*" jika berhasil. Berisi pesan *error* jika gagal.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>fs.changeLowerBound</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>DEVICEID</name>
                <value><string>FS 001</string></value>
        </member>
        <member>
                <name>BOUND</name>
                <value><int>10</int></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
```

```
        <param>
<value>
        <struct>
        <member>
                <name>RESPONSECODE</name>
                <value><string>00</string></value>
        </member>
        <member>
                <name>MESSAGE</name>
                <value><string>Succeed</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

14. **fs.changeExpiredDate** : fungsi untuk mengganti tanggal kadaluarsa pada wadah di tabel *TempatMakananHeader*.

Gateway mengirimkan *XML-RPC* dengan *methodName* = *fs.changeExpiredDate* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.41.

Tabel 3.41 Struktur *Struct* Fungsi *fs.changeExpiredDate Request*

Key	**Tipe Data**	**Keterangan**
DEVICEID	String 6	*Device ID* wadah yang ingin diubah tanggal kadaluarsanya.
KADALUARSA	String	Tanggal kadaluarsa baru. Berupa *long epoch time seconds.*
PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah struct dengan nilai seperti pada Tabel 3.42.

Tabel 3.42 Struktur *Struct* Fungsi *fs.changeExpiredDate Response*

Key	**Tipe Data**	**Keterangan**
RESPONSECODE	String	'00' : berhasil mengganti tanggal kadaluarsa. '01' : gagal mengganti.
MESSAGE	String	Berisi "*Succeed*" jika berhasil. Berisi pesan *error* jika gagal.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>fs.changeExpiredDate</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>DEVICEID</name>
                <value><string>FS 001</string></value>
        </member>
        <member>
                <name>KADALUARSA</name>
                <value><string>1432565117649</string></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
```

```
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>RESPONSECODE</name>
                <value><string>00</string></value>
        </member>
        <member>
                <name>MESSAGE</name>
                <value><string>Succeed</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

15. **fs.getDetailFoodStorage** : mengetahui detail dari wadah dengan *device ID* yang ditentukan, termasuk isi persentase terakhir.

Gateway mengirimkan *XML-RPC* dengan *methodName* = *fs.getDetailFoodStorage* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.43.

Tabel 3.43 Struktur *Struct* Fungsi *fs.getDetailFoodStorage Request*

Key	**Tipe Data**	**Keterangan**
DEVICEID	String 6	*Device ID* wadah yang ingin diketahui detailnya.

PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah struct dengan nilai seperti pada Tabel 3.44.

Tabel 3.44 Struktur *Struct* Fungsi *fs.getDetailFoodStorage Response*

Key	**Tipe Data**	**Keterangan**
NOTIFKADAL	Int	Jumlah berapa hari sebelum kadaluarsa notifikasi dijalankan.
NOTIFSTOK	Int	Frekuensi *update* stok dalam menit dari wadah sekaligus pengecekan dengan batas bawah untuk notifikasi.
UPDATEUMUR	Int	Frekuensi *update* umur dalam hari.
UMUR	Int	Umur isi wadah dalam hari.
BATASBAWAH	Int	Batas bawah persentase stok untuk pesan *low stock*.
TGLKADAL	String	Tanggal kadaluarsa dalam bentuk *long epoch time seconds*.
PERSENTASE	Int	Persentase terakhir wadah.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>fs.getDetailFoodStorage</methodName>
        <params>
        <param>
```

```
<value>
        <struct>
        <member>
                <name>DEVICEID</name>
                <value><string>FS 001</string></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>NOTIFKADAL</name>
                <value><int>2</int></value>
        </member>
        <member>
                <name>NOTIFSTOK</name>
                <value><int>5</int></value>
        </member>
        <member>
                <name>UPDATEUMUR</name>
                <value><int>1</int></value>
        </member>
        <member>
                <name>UMUR</name>
                <value><int>25</int></value>
        </member>
        <member>
                <name>BATASBAWAH</name>
                <value><int>20</int></value>
        </member>
        <member>
                <name>TGLKADAL</name>
```

```
                <value><string>1432565117649</string></value>
        </member>
        <member>
                <name>PERSENTASE</name>
                <value><int>50</int></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

16. **getComPortFromDeviceID** : fungsi untuk mengetahui COM Port wadah (atau *device* lain yang menggunakan *bluetooth*) berdasarkan *device ID* yang diberikan.

Gateway mengirimkan *XML-RPC* dengan *methodName* = *getComPortFromDeviceID* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.45.

Tabel 3.45 Struktur *Struct* Fungsi *getComPortFromDeviceID Request*

Key	**Tipe Data**	**Keterangan**
DEVICEID	String 6	*Device ID* yang ingin diketahui nilai *COM port*-nya.
PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah struct dengan nilai seperti pada Tabel 3.46.

Tabel 3.46 Struktur *Struct* Fungsi *getComPortFromDeviceID Response*

Key	**Tipe Data**	**Keterangan**
RESPONSECODE	String	'00' : *COM Port* tidak ada. '01' : *COM Port* ada.
MESSAGE	String	Berisi *COM Port* jika ada di *database*. Berisi "*COM Port not exist*" jika *COM Port* tidak ada di *database*.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>getComPortFromDeviceID</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>DEVICEID</name>
                <value><string>FS 001</string></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
```

```
        <struct>
        <member>
                <name>RESPONSECODE</name>
                <value><string>01</string></value>
        </member>
        <member>
                <name>MESSAGE</name>
                <value><string>COM21</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

17. **getEUI64FromDeviceID** : fungsi untuk mendapatkan EUI64 dari *device* yang menggunakan protokol *zigbee*.

    Gateway mengirimkan *XML-RPC* dengan *methodName* = *getEUI64FromDevice* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.47.

Tabel 3.47 Struktur *Struct* Fungsi *getEUI64FromDeviceID Request*

Key	**Tipe Data**	**Keterangan**
DEVICEID	String 6	*Device ID* yang ingin diketahui nilai *EUI64*-nya.
PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah struct dengan nilai seperti pada Tabel 3.48.

Tabel 3.48 Struktur *Struct* Fungsi *getEUI64FromDeviceID Response*

Key	**Tipe Data**	**Keterangan**
RESPONSECODE	String	‘00’ : *EUI64* tidak ada. ’01’ : *EUI64* ada.
MESSAGE	String	Berisi *EUI64* jika ada di *database*. Berisi “*EUI64 not exist*” jika *EUI64* tidak ada di *database*.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>getEUI64FromDeviceID</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>DEVICEID</name>
                <value><string>FS 001</string></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
```

```
<methodResponse>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>RESPONSECODE</name>
                <value><string>01</string></value>
        </member>
        <member>
                <name>MESSAGE</name>
                <value><string>000D6F00023832D3</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

18. **getListComPortBluetooth** : fungsi untuk mendapatkan semua nilai *COM Port* (mode *Bluetooth*) dari *username* yang diberikan.

Gateway mengirimkan *XML-RPC* dengan *methodName* = *getListComPortBluetooth* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.49.

Tabel 3.49 Struktur *Struct* Fungsi *getListComPortBluetooth Request*

Key	**Tipe Data**	**Keterangan**
USERNAME	String 20	*Username* yang ingin diketahui *list COM Port*.
PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah struct dengan nilai seperti pada Tabel 3.50.

Tabel 3.50 Struktur *Struct* Fungsi *getListComPortBluetooth Response*

Key	**Tipe Data**	**Keterangan**
COMPORT	Array of String	Berisi *list COM Port* yang bertipe data String.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>getListComPortBluetooth</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>USERNAME</name>
                <value><string>tralala</string></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>COMPORT</name>
                <value>
                        <array><data>
                        <value><string>COM21</string></value>
```

```
                    <value><string>COM22</string></value>
                    <value><string>COM23</string></value>
                    </data></array>
                </value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

19. **fs.updatePercent** : fungsi untuk menambahkan *row* di *TempatMakananDetail* untuk *update* persentase stok isi wadah. Waktu dari server.

Gateway mengirimkan *XML-RPC* dengan *methodName* = *fs.updatePercent* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.51.

Tabel 3.51 Struktur *Struct* Fungsi *fs.updatePercent Request*

Key	**Tipe Data**	**Keterangan**
DEVICEID	String 6	*Device ID* wadah yang meng-*update* persentase.
PERSENTASE	Int	Persentase stok terbaru.
WAKTU	String	Waktu *update* dalam *long epoch time seconds*.
PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah struct dengan nilai seperti pada Tabel 3.52.

Tabel 3.52 Struktur *Struct* Fungsi *fs.updatePercent Response*

Key	**Tipe Data**	**Keterangan**
RESPONSECODE	String	'00' : berhasil *update* persentase. '01' : gagal *update*.
MESSAGE	String	Berisi "*Succeed*" jika berhasil. Berisi pesan *error* jika gagal.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>fs.updatePercent</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>DEVICEID</name>
                <value><string>FS 001</string></value>
        </member>
        <member>
                <name>PERSENTASE</name>
                <value><int>40</int></value>
        </member>
        <member>
                <name>WAKTU</name>
                <value><string>1432821473</string></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>RESPONSECODE</name>
                <value><string>00</string></value>
        </member>
        <member>
                <name>MESSAGE</name>
                <value><string>Succeed</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

20. **fs.updateAge** : fungsi *update* umur pada wadah pada tabel *TempatMakananHeader*.

    Gateway mengirimkan *XML-RPC* dengan *methodName* = *fs.updateAge* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.53.

Tabel 3.53 Struktur *Struct* Fungsi *fs.updateAge Request*

Key	**Tipe Data**	**Keterangan**
DEVICEID	String 6	*Device ID* wadah yang meng-*update* umur.
AGE	Int	Umur isi wadah terbaru (dalam hari).
PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah struct dengan nilai seperti pada Tabel 3.54.

Tabel 3.54 Struktur *Struct* Fungsi *fs.updateAge Response*

Key	**Tipe Data**	**Keterangan**
RESPONSECODE	String	'00' : berhasil *update* umur. '01' : gagal *update*.
MESSAGE	String	Berisi "*Succeed*" jika berhasil. Berisi pesan *error* jika gagal.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>fs.updateAge</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>DEVICEID</name>
                <value><string>FS 001</string></value>
        </member>
        <member>
                <name>AGE</name>
                <value><int>23</int></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>RESPONSECODE</name>
                <value><string>00</string></value>
        </member>
        <member>
                <name>MESSAGE</name>
                <value><string>Succeed</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

21. **rf.updateData** : fungsi *update* jumlah telur dan suhu kulkas. [**31**]

    Gateway mengirimkan *XML-RPC* dengan *methodName* = *rf.updateData* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.53.

Tabel 3.55 Struktur *Struct* Fungsi *rf.updateData Request*

Key	**Tipe Data**	**Keterangan**
DEVICEID	String 6	*Device ID* kulkas yang meng-*update* data.
JMLTELUR	Int	Jumlah telur terbaru.
SUHU	Int	Suhu kulkas dalam derajat Celcius.
WAKTU	String	Waktu *update* dalam *long epoch time seconds*.

PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah struct dengan nilai seperti pada Tabel 3.54.

Tabel 3.56 Struktur *Struct* Fungsi *rf.updateData Response*

Key	**Tipe Data**	**Keterangan**
RESPONSECODE	String	'00' : berhasil *update* umur. '01' : gagal *update*.
MESSAGE	String	Berisi "*Succeed*" jika berhasil. Berisi pesan *error* jika gagal.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>rf.updateData</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>DEVICEID</name>
                <value><string>FS 001</string></value>
        </member>
        <member>
                <name>JMLTELUR</name>
                <value><int>3</int></value>
        </member>
        <member>
                <name>SUHU</name>
                <value><int>3</int></value>
        </member>
        <member>
                <name>WAKTU</name>
                <value><string>1432537435210</string></value>
```

```
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>RESPONSECODE</name>
                <value><string>00</string></value>
        </member>
        <member>
                <name>MESSAGE</name>
                <value><string>Succeed</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

22. **rf.getCompleteRefrigerator** : fungsi ini dipakai untuk mengambil *list* kulkas beserta detailnya. [**31**]

    Gateway mengirimkan *XML-RPC* dengan *methodName* = *rf.getCompleteRefrigerator* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.61.

Tabel 3.57 Struktur *Struct* Fungsi *rf.getCompleteRefrigerator Request*

Key	**Tipe Data**	**Keterangan**
USERNAME	String 20	*Username* yang akan mengambil detail kulkas.
PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah *array of struct* dengan nilai *struct* seperti pada Tabel 3.62.

Tabel 3.58 Struktur *Struct* (dalam Array) Fungsi *rf.getCompleteRefrigerator Response*

Key	**Tipe Data**	**Keterangan**
DEVICEID	String	Device ID kulkas.
JMLTELUR	Int	Jumlah telur yang tersisa di kulkas yang bersangkutan.
SUHU	Int	Suhu kulkas yang terakhir di-*update*. Dalam derajat Celcius.
LASTUPDATE	String	Waktu terakhir kali *update* dalam *long epoch time*.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>rf.getCompleteRefrigerator</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
```

```
                <name>USERNAME</name>
                <value><string>tralala</string></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
        <array>
        <data>
                <value>
        <struct>
        <member>
                <name>DEVICEID</name>
                <value><string>RF 001</string></value>
        </member>
        <member>
                <name>JMLTELUR</name>
                <value><int>5</int></value>
        </member>
        <member>
                <name>SUHU</name>
                <value><int>4</int></value>
        </member>
        <member>
                <name>LASTUPDATE</name>
                <value><string>1432537435210</string></value>
        </member>
        </struct>
                </value>
        </data>
        </array>
</value>
        </param>
```

```
        </params>
</methodResponse>
```

23. **rf.deleteRefrigeratorDevice** : fungsi untuk menghapus kulkas dari tabel Kulkas. [**31**]

Gateway mengirimkan *XML-RPC* dengan *methodName* = *rf.deleteRefrigeratorDevice* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.59.

Tabel 3.59 Struktur *Struct* Fungsi *rf.deleteRefrigeratorDevice Request*

Key	**Tipe Data**	**Keterangan**
DEVICEID	String 6	*Device ID* kulkas yang ingin dihapus
PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke *gateway* dengan sebuah struct dengan nilai seperti pada Tabel 3.60.

Tabel 3.60 Struktur *Struct* Fungsi *rf.deleteRefrigeratorDevice Response*

Key	**Tipe Data**	**Keterangan**
RESPONSECODE	String	‘00’ : berhasil menghapus kulkas. ’01’ : gagal menghapus.
MESSAGE	String	Berisi “*Succeed*” jika berhasil. Berisi pesan *error* jika gagal.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *gateway* ke *server*.

```
<?xml version="1.0"?>
```

```
<methodCall>
<methodName>rf.deleteRefrigeratorDevice</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>DEVICEID</name>
                <value><string>RF 001</string></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada *gateway.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>RESPONSECODE</name>
                <value><string>00</string></value>
        </member>
        <member>
                <name>MESSAGE</name>
                <value><string>Succeed</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

Selain itu, *gateway* secara periodik mengirimkan *heartbeat* untuk meminta *command* yang dikirimkan oleh *Android client* atau *client* lainnya.

Jika ada *command* yang harus dijalankan, *heartbeat* sementara dihentikan kemudian *command* dijalankan sampai selesai, baru ambil *heartbeat* lagi. *Heartbeat* berisi *gatewayID* (*device ID gateway*), *heartbeatID* (dalam *int*) yang selalu *increment* setelah mendapatkan *command* baru, dan *password* untuk keperluan autentikasi. Jika belum mendapatkan *command* baru, *heartbeatID* akan tetap sama untuk periode berikutnya. Variabel tersebut dikirimkan dalam *struct.* Berikut adalah format *heartbeat* dalam *XML-RPC* yang dikirimkan oleh *gateway*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>heartbeat</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>GATEWAYID</name>
                <value><string>ZZ 001</string></value>
        </member>
        <member>
                <name>HEARTBEATID</name>
                <value><int>1</int></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Respons dari server berisi *command* dan *parameter* untuk menjalankan *command*. Parameter merupakan string yang berisi parameter-parameter yang dipisahkan dengan tanda pagar. Jika belum

ada *command* untuk *gateway*, maka *server* akan mengirimkan *struct* kosong.

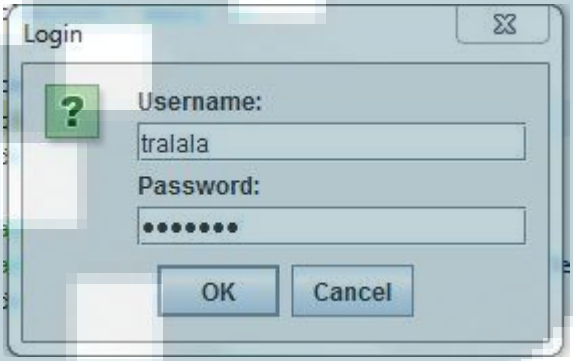
```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>COMMAND</name>
                <value><string>INI COMMAND</string></value>
        </member>
        <member>
                <name>PARAMETER</name>
 <value><string>parameter1#parameter2#parameter3</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

3.7. Perancangan *Software* Komputer *Gateway*

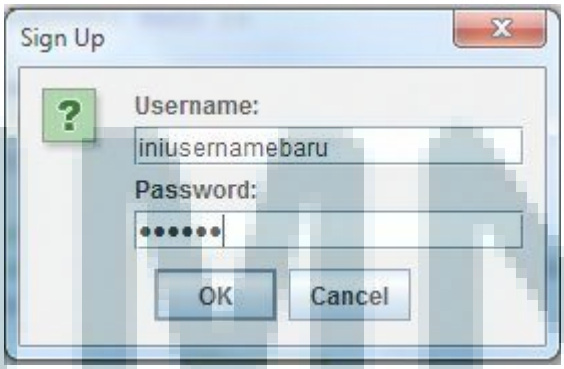
Komputer *gateway* dibangun dengan *Java* karena *Java* dapat digunakan di banyak *platform*. Bentuk antarmuka berupa *tab* sehingga memudahkan pengguna untuk berpindah antara *device* yang satu dengan yang lain. Saat ini terdapat tiga buah *tab* yaitu *tab General, tab Food Storage,* dan *tab Refrigerator*. Aplikasi diawali dengan rangkaian *message popup* untuk *login* atau *sign up*. Gambar 3.9 merupakan *popup* pertama untuk mengetahui apakah pengguna sudah mendaftar atau belum. Jika sudah akan masuk ke *popup* untuk *login* seperti pada Gambar 3.10. Jika belum akan masuk ke *popup* untuk *sign up* seperti pada Gambar 3.11.

Setelah melakukan *login* atau *sign up*, akan muncul *popup* seperti pada Gambar 3.12 dan Gambar 3.13 jika berhasil atau seperti pada Gambar 3.14, Gambar 3.15, dan Gambar 3.16 jika terdapat kesalahan.

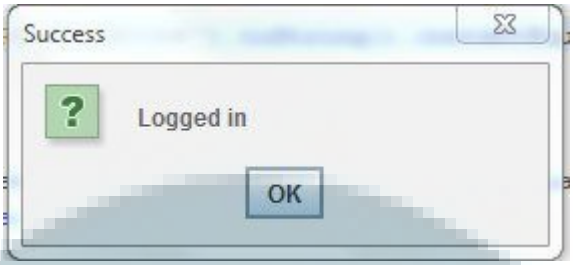
Gambar 3.9 *Popup Message* Awal



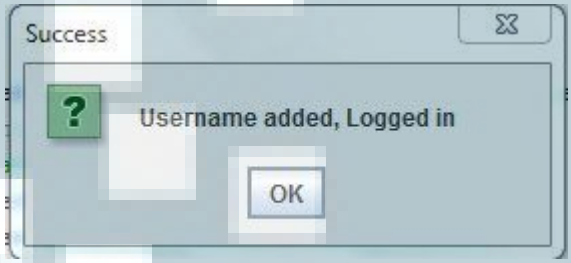
Gambar 3.10 *Popup Message Login*



Gambar 3.11 *Popup Message Sign Up*



Gambar 3.12 *Popup Message Login* Berhasil



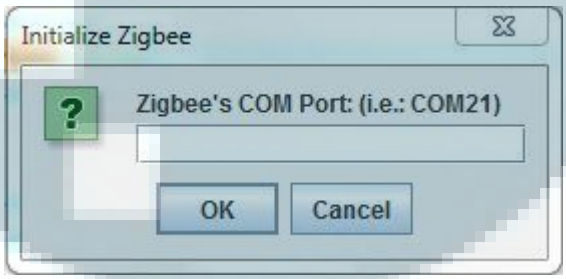
Gambar 3.13 *Popup Message Sign Up* Berhasil

Gambar 3.14 *Popup Message Login* Gagal (Salah *Password*)

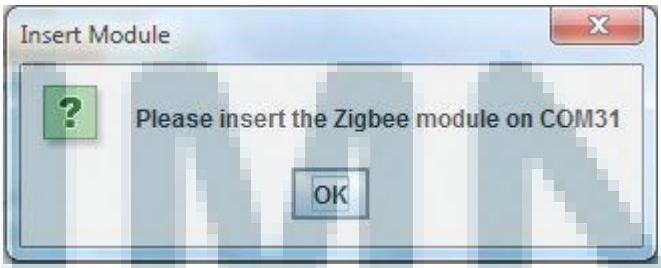
Gambar 3.15 *Popup Message Login* Gagal (*Username* Tidak Ada)

Gambar 3.16 *Popup Message Sign Up* Gagal (*Username* Dikosongkan)

Setelah berhasil *login* atau *sign up*, program akan mengeluarkan *popup* message seperti pada Gambar 3.17 lalu setelah di-OK Gambar 3.18 untuk inisialisasi *port* serial yang akan digunakan untuk *zigbee*. Setelah *popup* tersebut, program akan membuka *port* komunikasi yang digunakan.



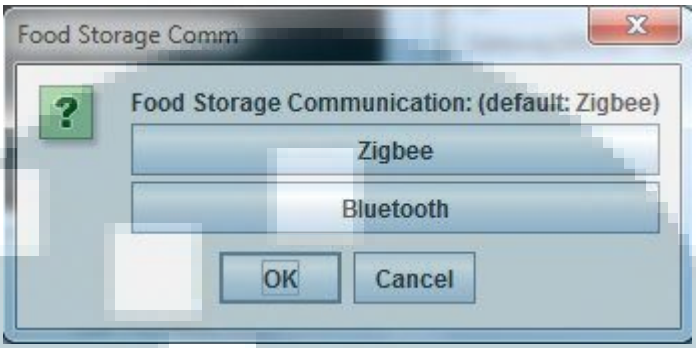
Gambar 3.17 *Popup Message* Inisialisasi *Zigbee*



Gambar 3.18 *Popup Message* untuk Pengingat Memasukkan *Zigbee*

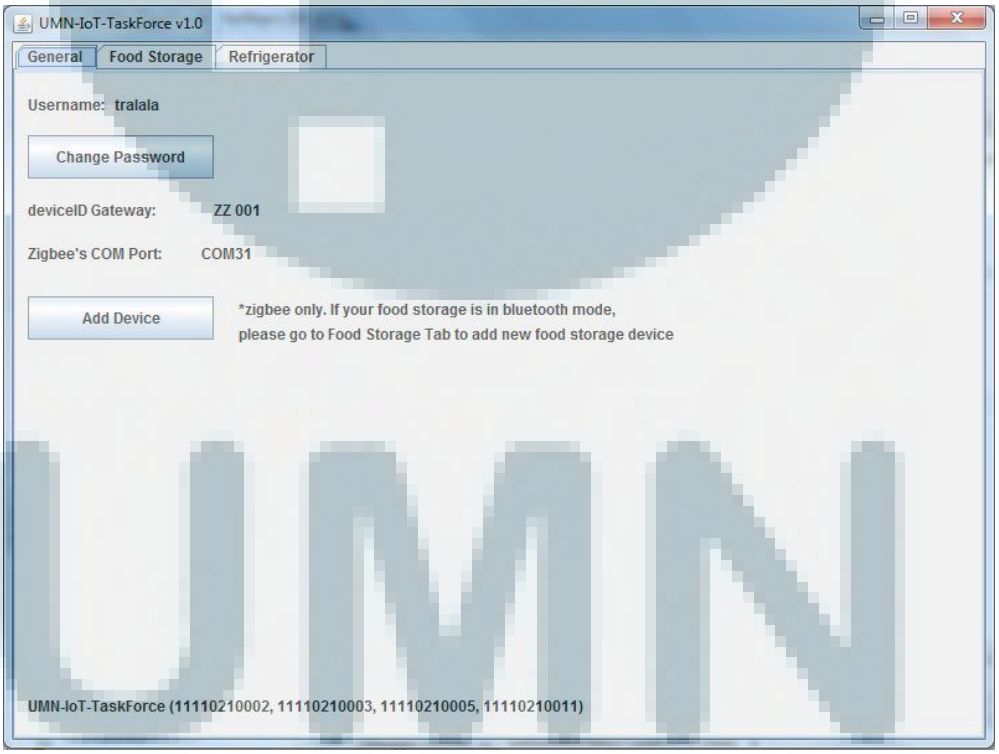
Setelah menentukan *port* komunikasi untuk *Zigbee*, pengguna perlu memilih mode komunikasi untuk wadah. Terdapat pilihan *Bluetooth* atau *Zigbee* seperti pada Gambar 3.19. Setelah memilih, pengguna dapat

menekan OK. Jika pengguna tidak memilih, secara *default* akan masuk ke mode komunikasi *Zigbee*.



Gambar 3.19 *Popup Message* untuk Pemilihan Mode Komunikasi Wadah

Tab General seperti pada Gambar 3.20 merupakan *tab* umum yang dapat digunakan oleh semua jenis *device* yang menggunakan *zigbee* dan



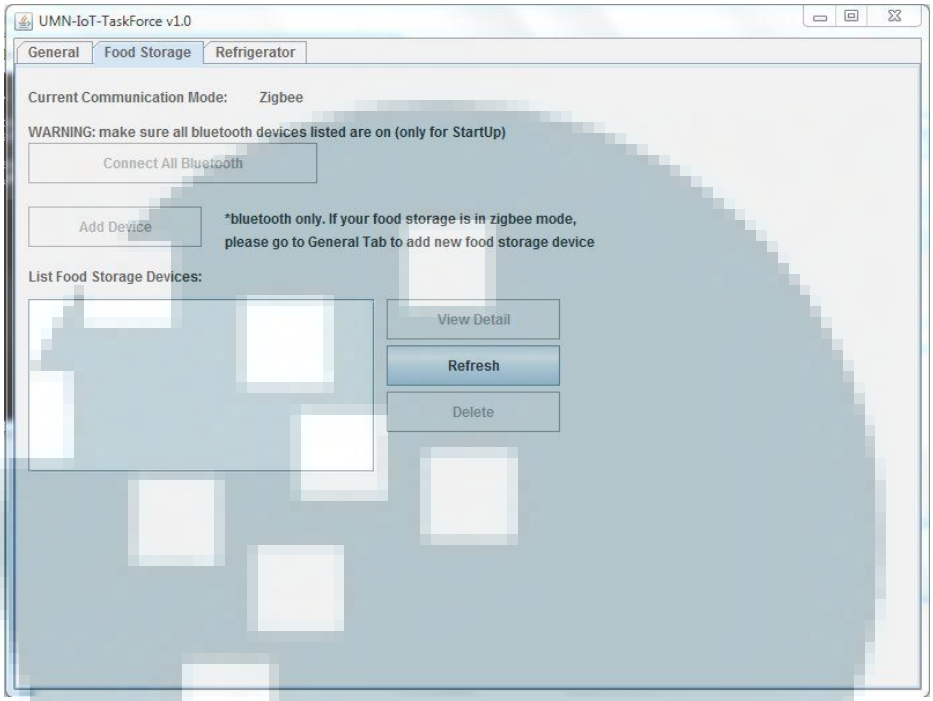
Gambar 3.20 *Tab General*

memuat informasi umum seperti *username, port* komunikasi yang digunakan untuk terhubung ke modul *zigbee,* dan *deviceID gateway*. Terdapat fungsi ganti *password* dengan menekan tombol *Change Password* sehingga akan muncul *popup message* seperti pada Gambar 3.21. Jika ingin menambahkan *device* yang menggunakan *zigbee*, dapat menekan *Add Device*.

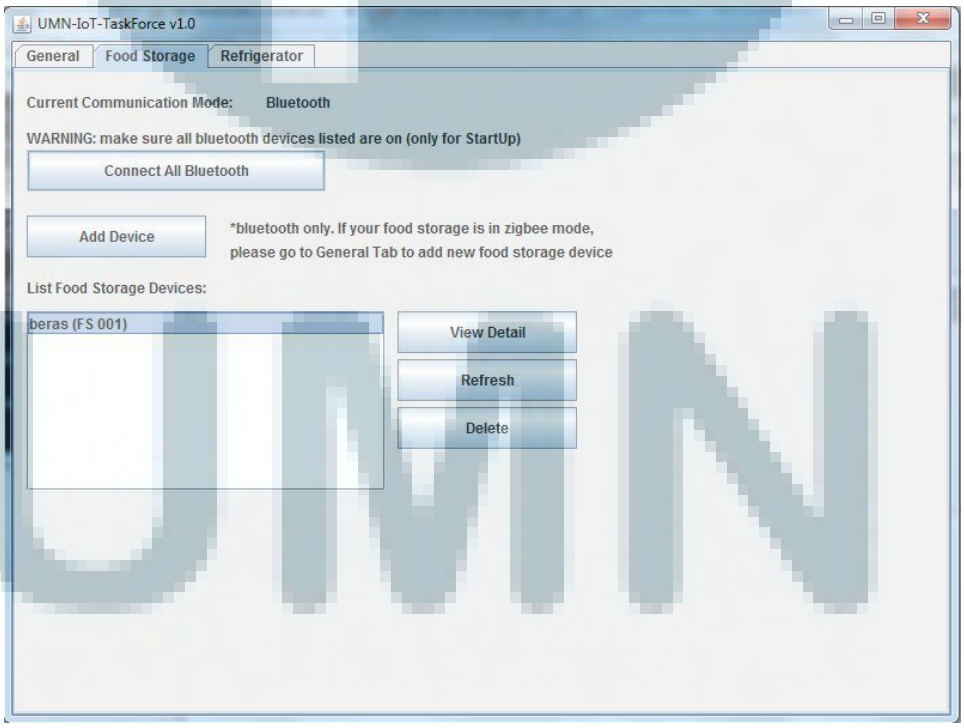
Gambar 3.21 *Popup Message Change Password*

Tab Food Storage pada Gambar 3.22 berisi informasi mengenai wadah yang terdaftar di *database* atas nama *username* yang bersangkutan saat menggunakan mode komunikasi *Zigbee*. Jika komunikasi adalah *Bluetooth*, beberapa tombol menjadi aktif seperti pada Gambar 3.23. Tombol *Add Device* dipakai untuk menambahkan wadah yang menggunakan *Bluetooth*. Tombol *Connect All Bluetooth* dipakai untuk mengoneksikan semua wadah yang menggunakan protokol *Bluetooth* terutama setelah *gateway* dimatikan. Tombol *View Detail* dipakai untuk memunculkan *popup* berisi detail wadah yang terpilih pada *list* dan dapat dipakai untuk mengganti *setting* seperti yang dapat dilihat pada Gambar

3.24. Tombol *Refresh* dipakai untuk mengambil ulang *list*

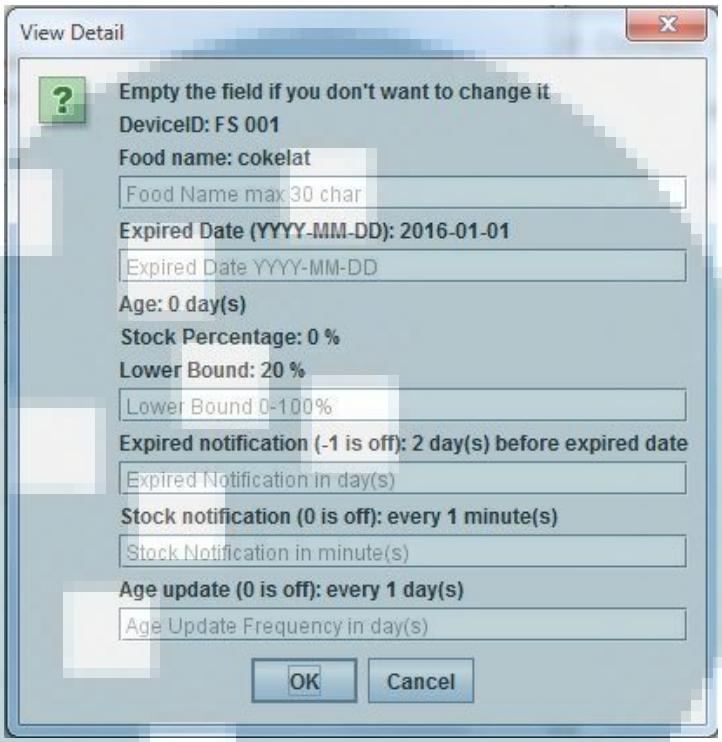


Gambar 3.22 *Tab Food Storage* dengan Mode *Zigbee*



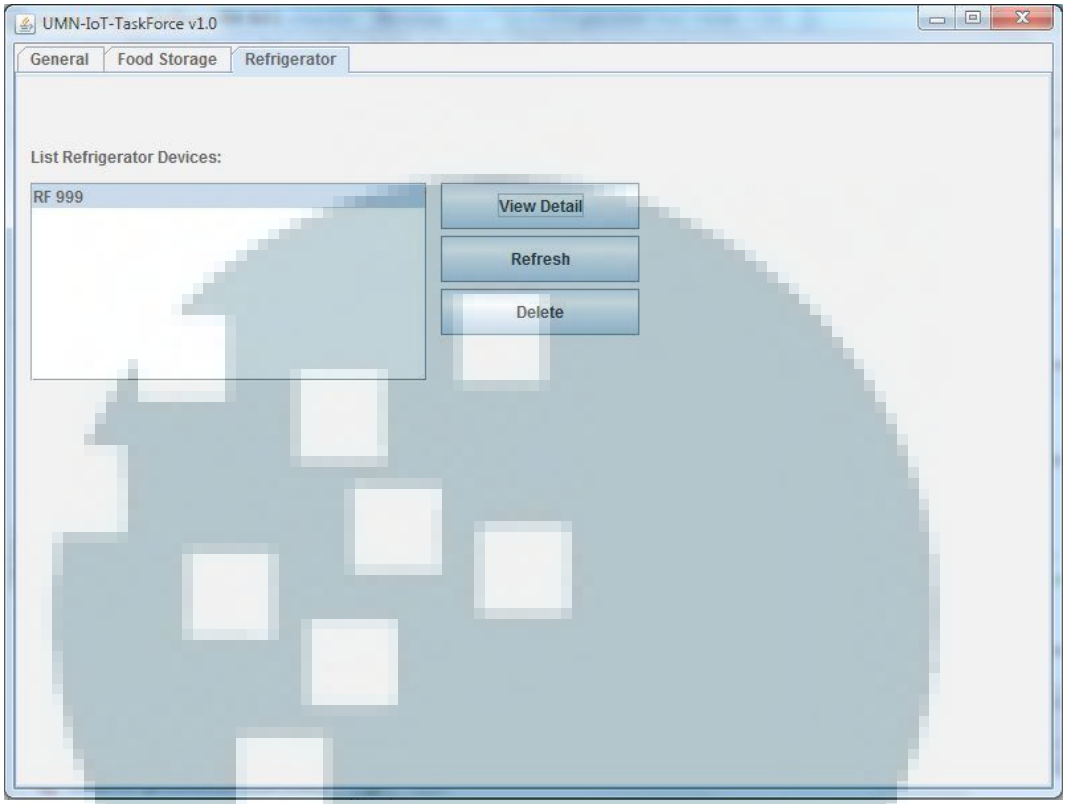
Gambar 3.23 *Tab Food Storage* dengan Mode *Bluetooth*

wadah dari *database* untuk dapat ditampilkan kembali. Tombol *delete* dipakai untuk menghapus wadah yang terpilih dari *list* dan *database*.

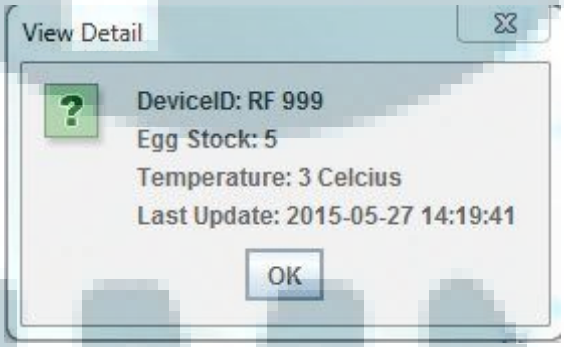


Gambar 3.24 *Popup Message View Detail* Wadah

Gambar 3.25 merupakan *tab Refrigerator* yang dipakai untuk menampilkan informasi lebih detail mengenai kulkas yang terdaftar di *database* atas nama *username* yang bersangkutan [**31**]. Untuk memunculkan *popup message* seperti pada Gambar 3.26 yang berisi detail mengenai kulkas yang terpilih pada *list* seperti stok telur dan suhu dapat menekan tombol *View Detail*. Tombol *Refresh* dipakai untuk mengambil ulang *list* kulkas dari *database* dan tombol *Delete* untuk menghapus kulkas yang terpilih pada *list*.



Gambar 3.25 *Tab Refrigerator*



Gambar 3.26 *Popup Message View Detail* Kulkas

Setiap membuka *port* komunikasi, sebuah *thread* dibuat untuk menerima dan mengirim serial pada *port* tersebut. Jika pada mode *bluetooth*, setiap *device* memiliki *thread*-nya masing-masing karena menggunakan *port* komunikasi yang berbeda. Jika menggunakan *zigbee*,

semua *device* berbagi di satu *port* yang sama karena *port* komunikasi yang dipakai hanya untuk berkomunikasi dengan modul *zigbee*.

Terdapat empat *class* yang dibuat selain *class-class* yang terdapat di *library*, yaitu *GatewayUMNIoTTaskForce, mySQL, TextPrompt, dan SerialReadWrite*. *Class GatewayUMNIoTTaskForce* merupakan *class* utama yang membangun *display* dan memanggil *class-class* lainnya. *Class mySQL* bertugas untuk terkoneksi dengan *server* dengan protokol *XML-RPC* yang detailnya dapat dilihat pada subbab 3.6. *Class TextPrompt* dipakai untuk *background text field* yang dipakai untuk memberikan petunjuk bagi pengguna seperti yang dipakai pada *View Detail* wadah Gambar 3.24. *Class SerialReadWrite* bertugas membuat *thread* kemudian terhubung ke *port* serial yang diminta yang kemudian dipakai untuk mengirim data dan menerima lalu mengolah pesan.

Untuk berkomunikasi dengan *port* serial yang ada di komputer, *Java Communication API (javax.comm)* digunakan. *BlueCove (javax.bluetooth)* dipakai untuk terhubung dengan *bluetooth*. *XmlRpc* terutama yang *org.apache.xmlrpc.client* dipakai untuk berhubungan dengan *server* melalui protokol *XML RPC*.

3.8. Perancangan Komunikasi antara *Server* dan *Android*

Server dan *Android* berkomunikasi dengan *XML-RPC*. Selain fungsi *addUsername* dan *findUsername* untuk *sign in* dan *sign up*, fungsi *rf.getCompleteRefrigerator* untuk mengambil *list* kulkas beserta detailnya

pada subbab 3.6, terdapat tambahan-tambahan fungsi seperti di bawah ini. Fungsi-fungsi ini dapat digunakan untuk pengambilan data dari *Android* atau pengecekan status untuk notifikasi dari *Android*. Selain itu, fungsi-fungsi ini juga dapat digunakan untuk komunikasi dengan yang lain.

1. **fs.getCompleteFoodStorage** : fungsi ini dipakai untuk mengambil *list* wadah beserta detailnya untuk ditampilkan pada aplikasi *Android.*

   Aplikasi *Android* mengirimkan *XML-RPC* dengan *methodName = fs.getCompleteFoodStorage* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.61.

Tabel 3.61 Struktur *Struct* Fungsi *fs.getCompleteFoodStorage Request*

Key	**Tipe Data**	**Keterangan**
USERNAME	String 20	*Username* yang akan mengambil detail wadah.
PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke *Android* dengan sebuah *array of struct* dengan nilai *struct* seperti pada Tabel 3.62.

Tabel 3.62 Struktur *Struct* (dalam *Array*) Fungsi *fs.getCompleteFoodStorage Response*

Key	**Tipe Data**	**Keterangan**
DEVICEID	String	Device ID wadah.
NAMAMAKANAN	String	Nama makanan atau bahan lainnya

		yang tersimpan di *database*.
PERSENTASE	Int	Persentase stok terakhir yang ada di *database*.
KADALUARSA	String	Tanggal kadaluarsa yang berupa angka *long epoch time seconds*.
UMUR	Int	Umur isi wadah saat itu dalam satuan hari.

Berikut adalah contoh format pesan *request* yang dikirimkan dari *Android* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>fs.getCompleteFoodStorage</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>USERNAME</name>
                <value><string>tralala</string></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada aplikasi *Android.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
```

```
        <param>
<value>
        <array>
        <data>
                <value>
        <struct>
        <member>
                <name>DEVICEID</name>
                <value><string>FS 001</string></value>
        </member>
        <member>
                <name>NAMA MAKANAN</name>
                <value><string>Kacang</string></value>
        </member>
        <member>
                <name>PERSENTASE</name>
                <value><int>30</int></value>
        </member>
        <member>
                <name>KADALUARSA</name>
                <value><string>1432537435210</string></value>
        </member>
        <member>
                <name>UMUR</name>
                <value><int>10</int></value>
        </member>
        </struct>
                </value>
                <value>
        <struct>
        <member>
                <name>DEVICEID</name>
                <value><string>FS 002</string></value>
        </member>
        <member>
                <name>NAMA MAKANAN</name>
                <value><string>Sereal</string></value>
        </member>
        <member>
                <name>PERSENTASE</name>
                <value><int>50</int></value>
        </member>
        <member>
                <name>KADALUARSA</name>
                <value><string>1432537435210</string></value>
        </member>
```

```
        <member>
                <name>UMUR</name>
                <value><int>5</int></value>
        </member>
        </struct>
                </value>
        </data>
        </array>
</value>
        </param>
        </params>
</methodResponse>
```

2. **fs.checkExpiredDateStatus** : fungsi ini dipakai untuk mengecek apakah ada makanan di wadah yang mendekati tanggal kadaluarsa.

   Aplikasi *Android* mengirimkan *XML-RPC* dengan *methodName = fs.checkExpiredDateStatus* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.63.

Tabel 3.63 Struktur *Struct* Fungsi *fs.checkExpiredDateStatus Request*

Key	**Tipe Data**	**Keterangan**
USERNAME	String 20	Username yang akan dicek apakah memiliki makanan di wadah yang akan mendekati kadaluarsa.
PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke aplikasi *Android* dengan sebuah struct dengan nilai seperti pada Tabel 3.64.

Tabel 3.64 Struktur *Struct* Fungsi *fs.checkExpiredDateStatus Response*

Key	**Tipe Data**	**Keterangan**
RESPONSECODE	String	'00' : tidak ada makanan yang dekat/sudah kadaluarsa '01' : ada makanan yang dekat/sudah kadaluarsa
NAMAMAKANAN	Array of String	Berisi list nama makanan yang akan atau sudah kadaluarsa.

Berikut adalah contoh format pesan *request* yang dikirimkan dari aplikasi *Android* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>fs.checkExpiredDateStatus</methodName>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>USERNAME</name>
                <value><string>tralala</string></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada aplikasi *Android.*

```
<?xml version="1.0"?>
```

```
<methodResponse>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>RESPONSECODE</name>
                <value><string>00</string></value>
        </member>
        <member>
                <name>NAMAMAKANAN</name>
                <value><array><data>
                <value><string>Keripik</string></value>
                <value><string>Sereal</string></value>
                <value><string>Susu</string></value>
                </data></array></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

3. **checkLowStockStatus** : fungsi ini dipakai untuk mengecek apakah ada makanan atau bahan lainnya di wadah dan/atau kulkas (telur) yang sudah berada pada status *low stock*. Telur akan dikatakan *low stock* jika jumlahnya kurang dari 4 butir.

Aplikasi *Android* mengirimkan *XML-RPC* dengan *methodName* = *checkLowStockStatus* dan parameter dalam sebuah *struct* dengan perincian seperti pada Tabel 3.63.

Tabel 3.65 Struktur *Struct* Fungsi *checkLowStockStatus Request*

Key	**Tipe Data**	**Keterangan**

USERNAME	String 20	Username yang akan dicek apakah memiliki makanan atau bahan lainnya di wadah/kulkas yang sudah *low stock*.
PASSWORD	String 64	Password yang telah di-*hash* dengan *SHA3-256* untuk autentikasi

Fungsi ini mengembalikan nilai ke aplikasi *Android* dengan sebuah struct dengan nilai seperti pada Tabel 3.64.

Tabel 3.66 Struktur *Struct* Fungsi *checkLowStockStatus Response*

Key	**Tipe Data**	**Keterangan**
RESPONSECODE	String	'00' : tidak ada makanan atau bahan lainnya yang *low stock.* '01' : ada makanan atau bahan lainnya yang *low stock.*
NAMAMAKANAN	Array of String	Berisi list nama makanan atau bahan lainnya yang *low stock* (untuk kulkas, nama makanannya Telur). Berisi *array* kosong jika tidak ada makanan atau bahan lainnya yang *low stock*.

Berikut adalah contoh format pesan *request* yang dikirimkan dari aplikasi *Android* ke *server*.

```
<?xml version="1.0"?>
<methodCall>
<methodName>checkLowStockStatus</methodName>
        <params>
        <param>
<value>
```

```
        <struct>
        <member>
                <name>USERNAME</name>
                <value><string>tralala</string></value>
        </member>
        <member>
                <name>PASSWORD</name>
        <value><string>086abe9e75baba382d84c1aeee6aacec9874aac77
99924e286ea000362d4db82</string></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodCall>
```

Berikut adalah contoh respon yang diberikan *server* kepada aplikasi *Android.*

```
<?xml version="1.0"?>
<methodResponse>
        <params>
        <param>
<value>
        <struct>
        <member>
                <name>RESPONSECODE</name>
                <value><string>01</string></value>
        </member>
        <member>
                <name>NAMAMAKANAN</name>
                <value><array><data>
                <value><string>Telur</string></value>
                <value><string>Cokelat</string></value>
                <value><string>Roti</string></value>
                </data></array></value>
        </member>
        </struct>
</value>
        </param>
        </params>
</methodResponse>
```

3.9. Perancangan Aplikasi *Server*

Server dibuat dengan bahasa *PHP*. *Server* bertugas untuk berkomunikasi dengan komputer *gateway* dengan protokol *XML-RPC*. *Server* juga berkomunikasi dengan *GCM Server* dengan protokol *XMPP (CCS)* untuk menerima dan mengirim data ke aplikasi *Android*.

Library yang digunakan untuk mempermudah penerimaan dan pembentukan pesan *XML-RPC* adalah *phpxmlrpc*. Terdapat tiga *file* yang harus di-*include* untuk menggunakan *library* ini, yaitu *"xmlrpc.inc", "xmlrpcs.inc"*, dan *"xmlrpc_wrappers.inc"*. Fungsi-fungsi yang dapat dipanggil melalui *XML-RPC* dapat dilihat di subbab 3.6 dan 3.8.

Server juga bertugas untuk mengubah dan mengambil isi *database mySQL* yang juga berada di *cloud*. *Query* dimasukkan tergantung dari fungsi yang dipanggil oleh *gateway* atau perangkat *Android* melalui *XML-RPC*.