BAB 4 ANALISIS KEBUTUHAN DAN PERANCANGAN

4.1 Deskripsi Umum Sistem

Sistem yang dibuat merupakan pengembangan dari sistem yang sudah ada. Tujuan sistem ini dibuat adalah untuk meningkatkan kualitas hidup manusia dari segi kesehatan sesuai dengan latar belakang penelitin ini. Tujuan berikutnya yaitu memudahkan pengguna dalam memperoleh arti dari data yang didapatkan oleh sensor sesuai dengan tujuan penelitian yang sudah disebutkan pada bab sebelumnya.

Pada sistem ini pemilihan antarmuka pengguna berdasarkan perangkat yang paling banyak digunakan, yaitu ponsel cerdas yang menggunakan sistem operasi Android. Antarmuka pengguna juga dibuat untuk pengguna yang ingin melihat detail data sensor menggunakan aplikasi Google Chrome pada komputer

4.2 Kebutuhan Sistem

Pada bagian ini akan dijelaskan kebutuhan sistem yang diawali dengan kebutuhan fungsional, kebutuhan non-fungsional, kebutuhan perangkat lunak, dan kebutuhan perangkat keras

4.2.1 Kebutuhan Fungsional

Kebutuhan fungsional merupakan daftar kebutuhan yang harus dipenuhi dan proses-proses yang bisa dilakukan oleh sistem. Kebutuhan fungsional dalam penelitian ini akan dijelaskan pada Tabel 4.1 berikut ini:

Tabel 4.1 Kebutuhan fungsional sistem

No.	Kebutuhan Fungsional
1	Node sensor dapat terhubung dengan *middleware*
2	Node sensor dapat mengirim data ke *middleware* melalui protokol CoAP
3	Node sensor dapat mengirim data ke *middleware* melalui protokol MQTT
4	Data center dapat mengambil data terakhir yang ada pada *database* dan mengirimkannya ke perangkat pengguna
5	Data center dapat mengambil data setiap jam dalam satu hari dari *database* dan mengirimkannya ke perangkat pengguna
6	Data center dapat mengambil data setiap lima menit dalam satu jam dari *database* dan mengirimkannya ke perangkat pengguna
7	Pengguna dapat melihat status terakhir karbon monoksida dari aplikasi pada ponsel cerdas yang menggunakan sistem operasi Android

8	Pengguna dapat menerima pemberitahuan mengenai karbon monoksida pada ponsel cerdas yang menggunakan sistem operasi Android apabila nilai karbon monoksida melebihi batas aman
9	Pengguna dapat melihat detail data yang telah di olah oleh data center dari aplikasi pada ponsel cerdas yang menggunakan sistem operasi Android
10	Pengguna dapat melihat detail data sensor dari aplikasi Google Chrome pada komputer

4.2.2 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional adalah kebutuhan yang membahas perilaku dan batasan fungsi yang ada pada sistem. Pada bagian ini, kebutuhan non-fungsional terdiri atas kebutuhan perangkat lunak dan kebutuhan perangkat keras yang akan disebutkan pada tabel 4.2 dan 4.3 berikut.

Tabel 4.2 Kebutuhan perangkat lunak

Perangkat	Kebutuhan Perangkat Lunak
NodeMCU Flasher Master	Aplikasi untuk meletakkan *firmware* pada ESP8266
NodeMCU firmware	Firmware berbasis *Lua* untuk ESP8266
ESPLorer	IDE untuk memprogram ESPB266 menggunakan bahasa Lua
Python	Bahasa pemrograman yang digunakan untuk mengolah data pada data center
Nano	Aplikasi untuk mengubah file program python
MongoDB	Basis data No-SQL untuk menyimpan data dari sensor
Android Studio	IDE untuk membuat aplikasi untuk ponsel pintar berbasis sistem operasi android
Google Chrome	Aplikasi *browser* pada komputer berbasis sistem operasi Windows

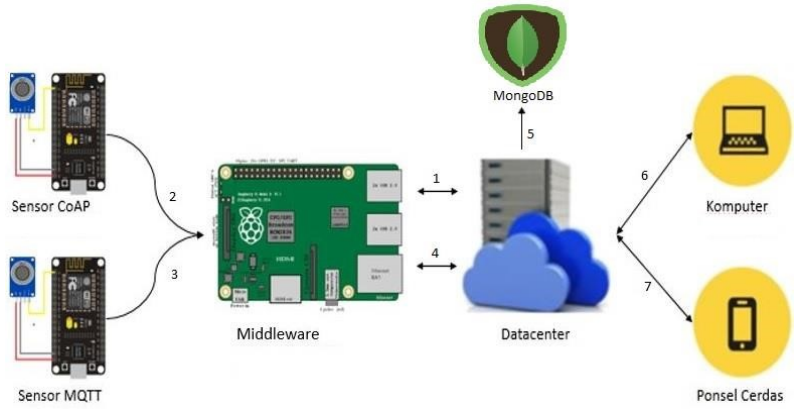
Tabel 4.3 Kebutuhan perangkat keras

Perangkat	Kebutuhan Perangkat Keras
Lenovo G400s-6485	Perangkat yang digunakan untuk melihat detail data sensor melalui aplikasi *browser*
LG Nexus 5	Perangkat yang digunakan untuk melihat kondisi udara berdasarkan pembacaan sensor, mendapatkan notifikasi terhadap kondisi udara,

	dan melihat ringkasan data yang diolah oleh data center
ESP8266	Perangkat ini sebagai mikrokontroler bagi sensor karbon monoksida.
MQ-7	Modul sensor untuk membaca nilai karbon monoksida pada lingkungan
Server Cloud (Ubuntu *instance*)	Perangkat ini digunakan untuk menjalankan aplikasi web dan bertindak sebagai *subscriber* yang kemudian menyimpan data sensor. Data yang disimpan kemudian diolah oleh perangkat ini sebelum disajikan kepada pengguna

4.3 Perancangan Alur Komunikasi Sistem

Pada sistem ini ada tiga bagian utama berdasarkan tugasnya, yaitu node sensor CO menjadi *publisher*, Raspberry Pi 2 model B akan menjadi *middleware*, dan komputer beserta telepon pintar akan menjadi *subscriber*. Model dan ukuran data serta *Quality of Service* yang digunakan ditentukan oleh peneliti. Perancangan alur komunikasi sistem dapat dilihat pada gambar 4.1 berikut ini.



Gambar 4.1 Alur komunikasi sistem

Berikut keterangan angka yang ada pada gambar:

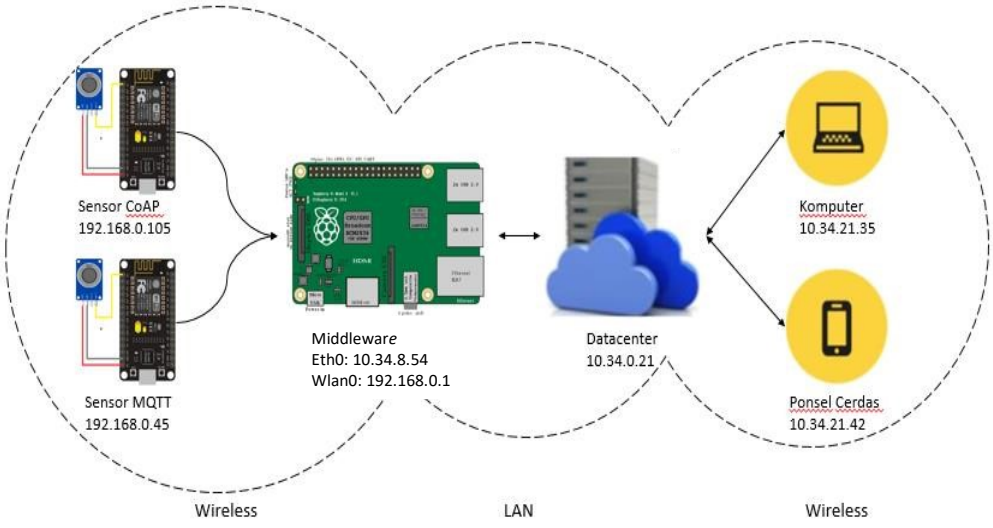
1. Aplikasi web pada data center subscribe topik `home/CO` dan `home/udara` ke *middleware*
2. Sensor CoAP mengirimkan data karbon monoksida dengan topik `home/udara` setiap 30 detik sekali
3. Sensor MQTT mengirimkan data karbon monoksida dengan topik `home/CO` setiap 30 detik sekali
4. Aplikasi web pada data center menerima data karbon monoksida untuk topik `home/CO` dan `home/udara` yang dikirm oleh sensor melalui data center

5. Data center menyimpan data sensor yang didapatkan pada basis data mongoDB
6. Aplikasi web pada data center diakses menggunakan *browser* pada sebuah laptop
7. Program pada data center diakses menggunakan aplikasi yang terpasang pada sebuah telepon pintar

Sensor CoAP akan mengirimkan data karbon monoksida ke *middleware* dengan topik `home/udara` dengan mengirimkan POST request ke midleware. Sedangkan sensor MQTT akan mengirimkan data karbon monoksida dengan topik `home/CO` dengan melakukan publish ke *middleware*. Di sisi lain, setiap kali sensor mengirimkan data, *middleware* akan mengirimkan data tersebut ke aplikasi web secara real-time menggunakan protokol Websocket. Sebelum aplikasi web dapat menerima data, aplikasi web harus terhubung terlebih dahulu dengan *middleware* melalui jaringan komputer dan subscribe pada topik yang dikirimkan oleh sensor. Setelah data diterima oleh aplikasi web, data tersebut selanjutnya disimpan dalam basis data MongoDB

4.4 Perancangan Topologi Jaringan

Topologi jaringan ini mencerminkan bagaimana setiap komponen sistem berinteraksi. Pada gambar 4.2 akan dijelaskan bagaimana sensor dapat terhubung dan mengirim data ke *middleware*, serta bagaimana *middleware* dapat terhubung dengan data center sehingga data center dapat menyimpan dan mengolah data dari *middleware*.



Gambar 4.2 Perancangan topologi jaringan

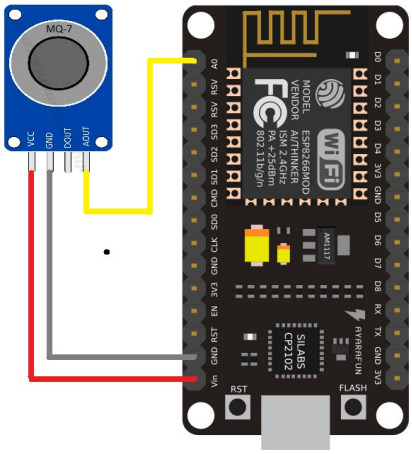
Raspberry Pi 2 model B berperan sebagai *middleware* antara sensor dan aplikasi serta berperan juga sebagai *access point*. Untuk menunjang hal ini dibutuhkan juga perangkat keras lain, yaitu USB Adapter EW-7811Un – Edimax yang berfungsi sebagai *wireless adapter*, microSD 8GB sebagai tempat

penyimpanan semua perangkat lunak yang dibutuhkan dan sistem operasi Raspbian Jessie. Pada Raspberry Pi 2 model B terdapat 2 buah *network interface* yaitu *interface* wlan0 yang terhubung ke sensor dengan IP static 192.168.0.1 serta eth0 yang terhubung ke jaringan LAN dengan IP static 10.34.8.54. Ketika *middleware* dijalankan, *middleware* menggunakan tiga port yaitu 5683 untuk CoAP, 1883 untuk MQTT, serta 3000 untuk Websocket. Selanjutnya sensor terhubung ke Raspberrry Pi dengan IP 192.168.0.45 untuk sensor MQTT dan 192.168.0.105 untuk sensor CoAP.

Di sisi lain, *subscriber* yang digunakan pada penelitian ini yaitu sebuah aplikasi web. Aplikasi ini dijalankan pada sebuah server yang terhubung dengan Raspberrry Pi melalui jaringan LAN dengan ip 10.34.0.21. Aplikasi ini akan berkomunikasi dengan *middleware* melalui interface eth0 pada Raspberrry Pi. Berikutnya aplikasi ini dapat diakses oleh browser pada laptop dengan alamat http://10.34.0.21:8001.

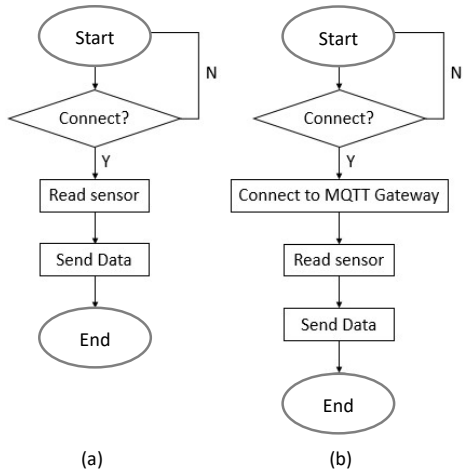
4.5 Perancangan Node Sensor

Perancangan node sensor menjelaskan bagaimana sensor di hubungkan dengan *microcontroller*, dalam hal ini adalah NodeMCU. Untuk membuat sebuah node sensor, alat yang dibutuhkan adalah NodeMCU, 3 buah kabel jumper, dan sensor MQ-7 (sensor karbon monoksida). Sensor memiliki 4 pin, yaitu VCC, GND, AOUT, dan DOUT. Pin yang digunakan hanya 3, yaitu VCC, GND, dan AOUT. VCC merupakan pin yang menunjukkan tegangan listrik positif (+). Pin VCC pada sensor dihubungkan dengan pin VCC pada NodeMCU yang memberikan listrik positif (+) bertegangan 5 volt. GND merupakan pin yang menunjukkan tegangan listrik negatif (-) 0 volt. Pin GND pada sensor dihubungkan dengan pin GND pada NodeMCU sebagai koneksi tegangan listrik negative (-). AOUT merupakan pin untuk mengirimkan data analog. Pin AOUT pada sensor dihubungkan dengan pin A0 pada NodeMCU yang dapat menerima data analog dari sensor seperti yang akan dijelaskan pada gambar 4.3 dibawah ini.



Gambar 4.3 Rancangan node sensor

ESP8266 harus diprogram yang berfungsi memerintahkan NodeMCU untuk dapat membaca data dari sensor dan mengonversi data yang didapat kedalam bentuk angka yang mudah dimengerti. Fungsi program lainnya yaitu untuk mengirimkan data tersebut ke *middleware* setiap 30 detik sekali. Berikut akan dijelaskan alur program node sensor pada gambar 4.4 dibawah ini.



Gambar 4.4 Diagram alur program node sensor (a) CoAP dan (b) MQTT

Program yang dibuat juga memerintahkan NodeMCU untuk mengambil data waktu terlebih dahulu dari server lokal, dalam hal ini adalah *middleware*. Data waktu dari server digunakan oleh NodeMCU untuk menginisialisasi waktu pengiriman data sensor berdasarkan waktu pada *middleware*. Berikut ditunjukkan pada tabel 4.4 mengenai semantik data yang dikirimkan NodeMCU ke *middleware.*

Tabel 4.4 Semantik data dari program node sensor

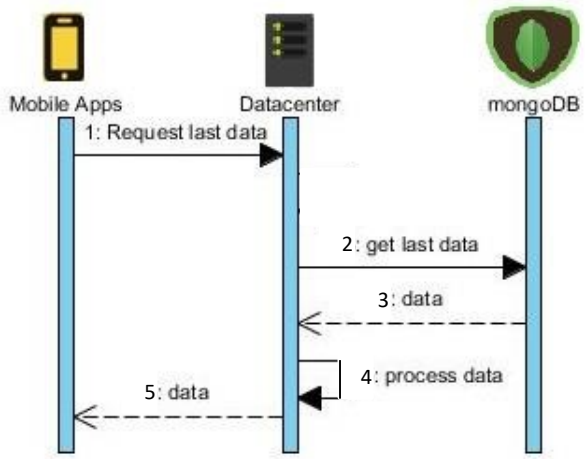
```
  {
    protocol = string,  // protokol yang digunakan
    timestamp = string, // diambil dari server lokal
    topic = string,
    sensor = {
      tipe = string,    // tipe node sudah ditetapkan
      index = string,   // index sensor
      ip = string,      // alamat IP node
      module = string   // modul sensor sudah ditetapkan MQ7
    },
    CarbonMonoxide = {
      value = number,   // nilai CO dari sensor
      unit = string     // dalam ppm
    }
}
```

4.6 Perancangan Data center

Pada data center akan dilakukan pengolahan data yang dikirimkan oleh *middleware*. Data center akan melakukan *subscribe* ke topik `home/CO` dan `home/udara`, dan mendapatkan data dari *middleware*. Data dari *middleware* akan disimpan dalam basis data mongoDB. Data tersebut kemudian di olah agar dapat mudah dimengerti oleh pengguna. Pengolahan data ini menggunakan sebuah program berbahasa Python bernama logic.py. Berikut akan dijabarkan 3 buah fungsi utama pada program Python yang akan dibuat.

4.6.1 Perancangan Pengambilan Data Terakhir

Pengambilan data terakhir ditujukan agar pengguna dapat mendapatkan data yang terakhir kali masuk ke data center. Ketika aplikasi pengguna melakukan *request* kepada data center agar dikirimkan data terakhir, maka data center akan melakukan fungsi ini. Hal pertama yang dilakukan fungsi ini adalah membuat koneksi dengan *database* pada *localhost* dengan *port* standar mongoDB 27017. Kedua, fungsi ini mengambil satu data terakhir yang ada pada *database* dan dikembalikan dengan nilai *cursor*. Hal selanjutnya adalah memasukkan *cursor* yang didapatkan ke dalam *list* untuk disaring sehingga data yang akan keluar hanya data `CarbonMonoxide` saja. Fungsi ini selanjutnya mengembalikan nilai `CarbonMonoxide` dalam bentuk string dan siap dikirimkan ke aplikasi pengguna. Berikut ditunjukkan pada gambar 4.5 mengenai *sequence* diagram dari fungsi pengambilan data terakhir.

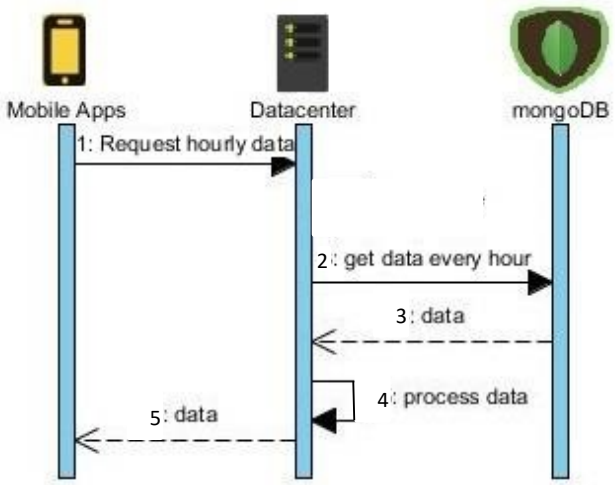


Gambar 4.5 *Sequence* diagram fungsi pengambilan data terakhir

4.6.2 Perancangan Pengambilan Data Setiap Jam

Pengambilan data setiap jam ditujukan agar pengguna dapat mendapatkan histori data yang lebih detail. Ketika aplikasi pengguna melakukan *request* kepada data center agar dikirimkan data pada tanggal tertentu, maka data center akan melakukan fungsi ini. Hal pertama yang dilakukan fungsi ini adalah membuat koneksi dengan *database* pada *localhost*. Kedua, fungsi ini mengambil semua data pada satu hari dalam *database* yang dibagi dengan jumlah jam dalam satu hari dan

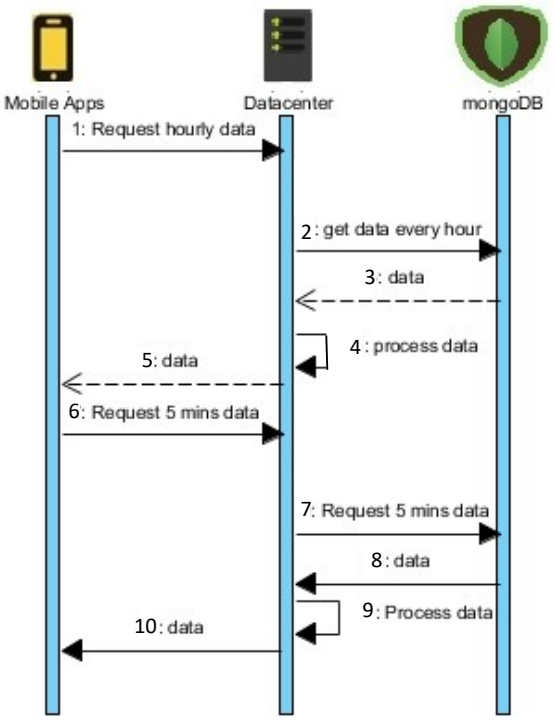
dikembalikan dengan nilai *cursor*. Hal selanjutnya adalah memasukkan *cursor* yang didapatkan ke dalam *list* untuk disaring sehingga data yang akan keluar hanya data `CarbonMonoxide` saja. Fungsi ini selanjutnya mengembalikan nilai `CarbonMonoxide` dalam bentuk *list string* dan siap dikirimkan ke aplikasi pengguna. Berikut dijelaskan pada gambar 4.6 mengenai *sequence* diagram dari fungsi pengambilan data setiap jam.



Gambar 4.6 *Sequence* diagram fungsi pengambilan data setiap jam

4.6.3 Perancangan Pengambilan Data Setiap Lima Menit

Pengambilan data setiap lima menit dalam 1 jam ditujukan agar pengguna dapat mendapatkan data yang lebih detail. Saat aplikasi pengguna melakukan *request* kepada data center agar dikirimkan data pada tanggal tertentu, kemudian fungsi pengambilan data setiap jam dijalankan, maka fungsi ini akan dapat dipanggil. Hal pertama yang dilakukan fungsi ini adalah membuat koneksi dengan *database* pada *localhost*. Kedua, fungsi ini mengambil semua data pada satu jam dalam *database* yang dibagi dengan lima menit dalam satu jam, kemudian dikembalikan dengan nilai *cursor*. Hal selanjutnya adalah memasukkan *cursor* yang didapatkan ke dalam *list* untuk disaring sehingga data yang akan keluar hanya data `CarbonMonoxide` saja. Fungsi ini selanjutnya mengembalikan nilai `CarbonMonoxide` dalam bentuk *list string* dan siap dikirimkan ke aplikasi pengguna. Berikut disediakan pada gambar 4.7 tentang *sequence* diagram dari fungsi pengambilan data setiap lima menit.



Gambar 4.7 *Sequence* diagram fungsi pengambilan data setiap lima menit

4.7 Perancangan Aplikasi Android

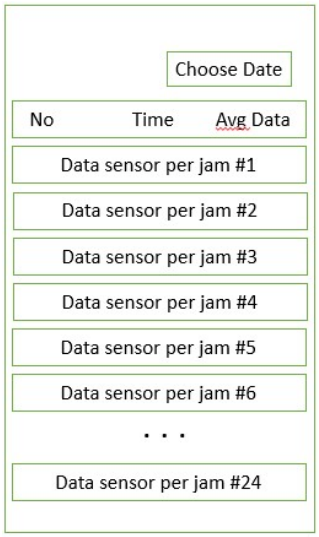
Aplikasi untuk telepon pintar berbasis sistem operasi Android digunakan untuk menampilkan data dari sensor yang digunakan. Data yang ditampilkan pada aplikasi pengguna didapatkan dari data center, dimana data tersebut akan diolah terlebih dahulu oleh data center sebelum dikirimkan ke aplikasi pengguna. Pengguna juga akan mendapatkan pemberitahuan berupa *heads-up notification*, suara, dan getaran apabila sensor mengirimkan nilai yang berada diatas standar aman karbon monoksida dalam ruangan. Aplikasi untuk telepon pintar berbasis sistem operasi Android ini terdiri dari 2 tampilan utama dan 1 tampilan tambahan.

4.7.1 Perancangan Tampilan

Tampilan pertama menampilkan data terakhir yang diterima oleh data center beserta penjelasan mengenai data sensor yang diterima pada aplikasi pengguna. Tampilan pertama memiliki 1 tombol, yaitu “Details” yang memiliki fungsi untuk menampilkan tampilan kedua pada aplikasi. Berikut akan ditampilkan perancangan tampilan utama pada gambar 4.8 dibawah ini.

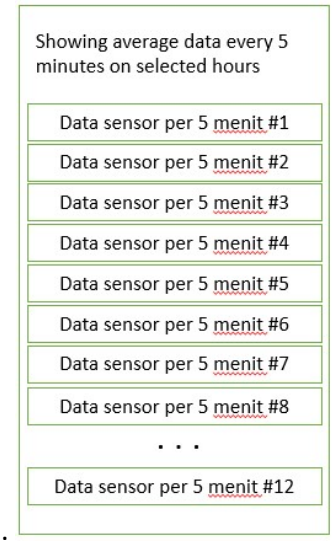
Gambar 4.8 Perancangan tampilan utama aplikasi android

Tampilan kedua menampilkan data sensor pada tanggal yang dipilih pengguna dan mengembalikan daftar berupa rata – rata data selama satu jam dalam satu hari. Data per jam yang ditampilkan didapatkan dari hasil pengolahan data pada data center. Pada setiap baris data yang ditampilkan, terdapat *listener* yang akan membaca baris ke berapa yang dipilih oleh pengguna. Berikut akan ditampilkan perancangan tampilan utama pada gambar 4.9 dibawah ini.



Gambar 4.9 Perancangan tampilan kedua aplikasi android

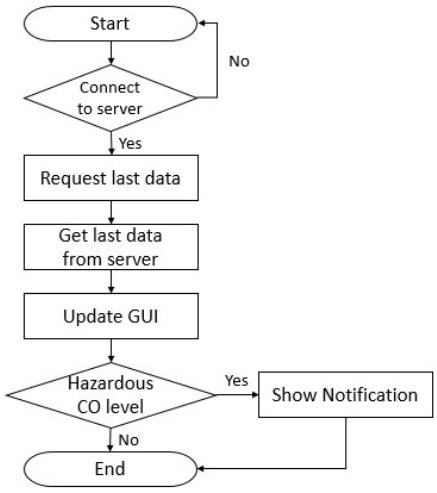
Tampilan tambahan ini tampil ketika pengguna memilih baris data pada tampilan kedua. Tampilan tambahan menampilkan data yang lebih rinci lagi yaitu data setiap lima menit berdasarkan pilihan jam dari pengguna. Data setiap lima menit yang tampil didapatkan dari hasil pengolahan data pada data center. Berikut akan ditampilkan perancangan tampilan utama pada gambar 4.10 dibawah ini.



Gambar 4.10 Perancangan tampilan tambahan aplikasi Android

4.7.2 Perancangan Program

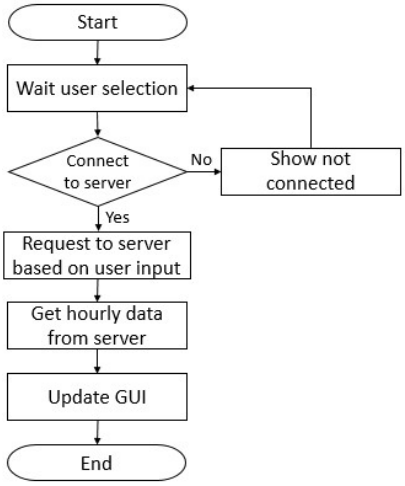
Program dibuat menggunakan aplikasi Android Studio. Program pada tampilan utama dibuat untuk mengambil data dari data center secara berkala, mengubah tampilan utama secara berkala, dan memberikan pemberitahuan kepada pengguna apabila nilai yang didapatkan dari data center melebihi batas aman. Pemberitahuan berupa *heads-up notification*, getar, dan suara dering. Berikut akan ditampilkan pada gambar 4.11 mengenai perancangan program untuk tampilan utama aplikasi telepon pintar.



Gambar 4.11 Diagram *flow* perancangan program tampilan utama

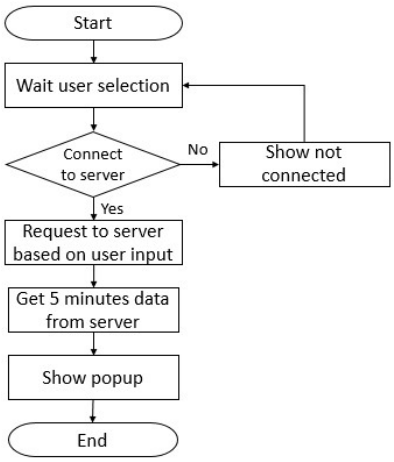
Ketika pengguna menekan tombol `details` pada tampilan utama, maka tampilan kedua akan muncul. Program untuk tampilan kedua berfungsi untuk menampilkan rata-rata data CO setiap satu jam dalam satu hari yang dipilih oleh

pengguna. Berikut akan ditampilkan pada gambar 4.12 mengenai perancangan program untuk tampilan utama aplikasi telepon pintar.



Gambar 4.12 Diagram *flow* perancangan program tampilan kedua

Ketika pengguna menekan salah satu grid pada tampilan kedua, maka tampilan tambahan akan muncul berupa *popup window*. Program untuk tampilan tambahan berfungsi untuk menampilkan rata-rata data CO setiap lima menit dalam satu jam yang dipilih oleh pengguna. Berikut akan ditampilkan pada gambar 4.13 mengenai perancangan program untuk tampilan utama aplikasi telepon pintar.



Gambar 4.13 Diagram *flow* perancangan program tampilan tambahan

4.8 Perancangan Pengujian

Perancangan pengujian dibutuhkan untuk mengetahui batasan minimal yang harus dipenuhi dalam pengujian. Dalam perancangan pengujian ini juga akan dibuat skenario dalam pengujian. Pada penelitian kali ini, akan dilakukan pengujian integritas, pengujian aplikasi, dan pengujian interoperabilitas.

4.8.1 Perancangan Pengujian Integritas

Pengujian integritas merupakan pengujian yang dilakukan untuk mengetahui interaksi antara dua atau lebih komponen untuk menjalankan satu fungsi tertentu. Pengujian ini dilakukan sebelum sistem digunakan untuk mengetahui apakah semua fungsi sistem yang pada kebutuhan fungsional sudah terpenuhi. Dalam penelitian ini pengujian integritas dilakukan secara otomatis menggunakan tools Behave. Behave merupakan salah satu program *Behavior Driven Development* (BDD) yang dapat menjadi alat pengujian integritas. Sama seperti *tools* Cucumber.js yang digunakan peneliti sebelumnya dalam pengujian penelitiannya, Behave juga menggunakan bahasa sehari-hari sebagai skenario pengujian. Untuk menjalankan Behave setidaknya diperlukan dua buah file yakni skenario pengujian dan kode pengujian. Apabila dua file tersebut sudah ada maka pengujian dapat dilakukan. Berikut dijelaskan skenario pengujian integritas pada tabel 4.5.

Tabel 4.5 Skenario pengujian integritas

Kode	Kebutuhan Fungsional	Skenario
P_01	Node sensor dapat terhubung dengan *middleware*	• Ketika node sensor melakukan koneksi ke *middleware* menggunakan Wi-Fi • Maka *middleware* menampilkan pemberitahuan bahwa *client* terhubung
P_02	Node sensor dapat mengirim data ke *middleware* melalui protokol CoAP	• Ketika data CoAP diterima oleh data center • Maka dapat terlihat data CoAP yang diterima
P_03	Node sensor dapat mengirim data ke *middleware* melalui protokol MQTT	• Ketika data MQTT diterima oleh data center • Maka dapat terlihat data MQTT yang diterima
P_04	Data center dapat mengambil data terakhir yang ada dari database dan mengirimkannya ke perangkat pengguna	• Ketika aplikasi meminta data terakhir pada database • Maka dapat terlihat data sensor terakhir yang diambil
P_05	Data center dapat mengambil data setiap jam dalam satu hari dari database dan mengirimkannya ke perangkat pengguna	• Ketika aplikasi meminta data pada tanggal 21-12-2017 • Maka dapat terlihat data setiap jam pada tanggal 21-12-2017

P_06	Data center dapat mengambil data setiap lima menit dalam satu jam dari database dan mengirimkannya ke perangkat pengguna	• Ketika aplikasi meminta data lima menit pada tanggal 21-12-2017 • Maka dapat terlihat data lima menit pada tanggal 21-12-2017
P_07	Pengguna dapat melihat status terakhir karbon monoksida dari aplikasi pada ponsel cerdas yang menggunakan sistem operasi Android	• Diketahui aplikasi sudah terpasang pada telepon pintar • Ketika aplikasi dijalankan • Maka pengguna dapat terlihat tingkat karbon monoksida
P_08	Pengguna dapat menerima pemberitahuan mengenai karbon monoksida pada ponsel cerdas yang menggunakan sistem operasi Android apabila nilai karbon monoksida melebihi batas aman	• Diketahui aplikasi terhubung dengan data center • Ketika status sensor tinggi • Maka pengguna mendapatkan pemberitahuan
P_09	Pengguna dapat melihat detail data yang telah di olah oleh data center dari aplikasi pada ponsel cerdas yang menggunakan sistem operasi Android	• Diketahui aplikasi sudah dibuka • Ketika pengguna menekan tombol `Details` • Maka pengguna dapat melihat detail data dari data center
P_10	Pengguna dapat melihat detail data sensor dari aplikasi Google Chrome pada komputer	• Diketahui aplikasi Google Chrome terpasang pada komputer pengguna • Ketika pengguna membuka aplikasi Google Chrome, memasukkan alamat URL `10.34.0.21:8001`, kemudian memasukkan topic `home/CO` atau `home/udara` • Maka dapat terlihat detail data sensor

4.8.2 Perancangan Pengujian Sistem

Pada pengujian sistem ini akan dilakukan pengujian apakah sistem yang dikembangkan berdasarkan kaidah komputasi berbasis konteks. Terdapat tiga kaidah utama dalam pengujian ini seperti yang disebutkan pada penjelasan bab sebelumnya. Berikut akan dijelaskan skenario pengujian sistem pada tabel 4.6 dibawah ini.

Table 4.6 Skenario pengujian sistem

Kode	Pengujian	Skenario
S_01	Sistem memenuhi fitur – fitur komputasi berbasis konteks	Pengujian akan dilakukan dengan mendefinisikan masing – masing fitur dengan sistem yang dibuat, kemudian dicocokkan dengan tiga buah fitur besar dalam komputasi berbasis konteks yang telah dijelaskan pada bab sebelumnya
S_02	Sistem dapat dibedakan berdasarkan tipe konteksnya dan dibuat skema pengkategoriannya	Langkah awal pengujian ini adalah dengan mendefinisikan dan mengklasifikasikan konteks primer dan sekunder. Langkah selanjutnya dengan membuat skema pengkategorian sesuai dengan konteks primer dan sekundernya.
S_03	Sistem dapat dibedakan berdasarkan tingkat konteks dan karakteristiknya	Pengujian akan dilakukan dengan mendefinisikan sistem sesuai dengan tingkat konteks dan karakteristiknya kemudian mengelompokkan sistem ke dalam tingkat konteks dan karakteristiknya sesuai dengan yang sudah disebutkan pada bab sebelumnya.

4.8.3 Perancangan Pengujian Quality of Service

Pengujian dilakukan dengan menghitung nilai *delay*, *throughput*, dan *jitter* pada pengiriman paket data dari sensor ke middleware. Pengujian mengunakan QoS tingkat 0 untuk protokol MQTT dan *Non-Confirmable* untuk protokol CoAP. Skenario pengujian nilai *delay*, *throughput*, dan *jitter*, akan dijelaskan pada tabel 4.7 dibawah ini.

Tabel 4.7 Skenario Pengujian *Quality of Service*

Kode	Pengujian	Skenario
T_01	Pengujian *throughput*	Pengujian dilakukan dengan memonitoring paket berisi data dari masing – masing sensor MQTT dan CoAP yang diterima *middleware* menggunakan bantuan *tools* tcpdump. Data paket yang diterima selanjutnya dibuka menggunakan *tools* Wireshark. Pengujian dilakukan selama satu jam dan di ulang sebanyak 3 kali untuk mendapatkan hasil yang lebih akurat. Pengujian

		dilakukan dengan mengambil nilai rata – rata *throughput* dalam satu kali skenario pengujian
T_02	Pengujian *delay*	Pengujian dilakukan dengan menggunakan data paket yang sama dengan pengujian *throughput.* Pengujian dilakukan dengan mengambil nilai rata – rata *delta time* dalam satu kali skenario pengujian, dimana *delta time* merupakan fitur pada Wireshark yang menghitung jarak waktu antara paket yang diterima sebelumnya dengan data paket yang diterima setelahnya.
T_03	Pengujian *jitter*	Pengujian dilakukan dengan menggunakan data paket dari pengujian *delay.* Pengujian dilakukan dengan mengurangi nilai *delta time* data setelahnya dengan nilai *delta time* data sebelumnya dalam satu kali skenario pengujian. Nilai tersebut kemudian dicari rata – ratanya.