BAB 5 IMPLEMENTASI

Bab ini menjelaskan tentang implementasi sistem pengembangan sistem karbon monoksida berbasis *Internet of Things.* Impementasi tersebut meliputi impementasi Node Sensor, implementasi data center, dan implementasi aplikasi telepon pintar berbasis sistem operasi Android

5.1 Implementasi Node Sensor

Pada implementasi node sensor, akan dibagi 2 bagian, yaitu implementasi Program init, implementasi program node sensor MQTT, dan implementasi program node sensor MQTT. implementasi program bertujuan memerintahkan NodeMCU untuk dapat membaca data dari sensor dan mengonversi data yang didapat kedalam bentuk angka yang mudah dimengerti.

5.1.1 Implementasi Program *Init*

Setiap saat NodeMCU dihidupkan, NodeMCU akan mencari dan menjalankan *file* init.lua. Program yang dibuat bertujuan memerintahkan NodeMCU untuk melakukan ekseskusi terhadap file yang berada di dalam memori NodeMCU. Di dalam program ini juga terdapat *timer* selama 5 detik sebelum memulai eksekusi *file* selanjutnya. Berikut ditampilkan *pseudo-code* dari program init.lua pada tabel 5.1 berikut ini.

Tabel 5.1 *Pseudo-code* program init.lua

```
DEFINE function startup()
    DO run file('MQ7_MQTT.lua')
    END
```

5.1.2 Implementasi Program Node Sensor MQTT

Program yang dibuat bertujuan memerintahkan NodeMCU untuk membaca data dari sensor dan mengirimkan data tersebut ke *middleware*. Di dalam program ini terdapat logika untuk merubah data raw dari sensor menjadi data PPM (*Part per Million*). Setelah mendapatkan raw data, dimulai perhitungan dengan persamaan matematis yang sudah dijelaskan pada bab sebelumnya. Sehingga untuk mendaptakan nilai PPM dari data raw sensor diberlakukan rumus sebagai berikut:

$$data_{ppm} = (38.028 \times \frac{R_s}{R_o})^{-0.796}$$

Data yang dikirim dari NodeMCU adalah tipe protokol, waktu pengiriman yang didapat dari *middleware*, topik, tipe sensor, dan data CO yang didapat dari sensor. Data dikirim dalam format JSON sesuai dengan semantik data yang sudah dijelaskan pada bab sebelumnya. Berikut dijelaskan pada tabel 5.2 mengenai *pseudo*-code program untuk mengirim data sensor ke *middleware*.

Tabel 5.2 *Pseudo-code* program untuk mengirim data sensor MQTT

```
DEFINE function connect()
  DO connect to mqtt gateway using define host and port
END

DEFINE function get_sensor_data()
  SET sensorValue = read carbon monixide data from sensor
  SET CO = convert RAW data readed from sensor
END

DEFINE function send_sensor_data()
  SET data = read carbon monixide data from sensor
  SET data = format data to JSON
  IF sensor value > 0 THEN
    DO send data over mqtt
  ELSE
    DO print error message
  END
END
```

5.1.3 Implementasi Program Node Sensor CoAP

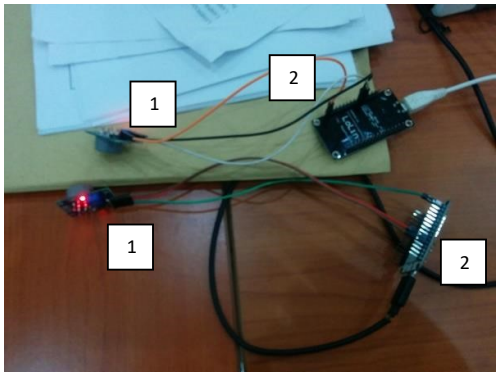
Program yang dibuat bertujuan memerintahkan NodeMCU untuk membaca data dari sensor dan mengirimkan data tersebut ke *middleware*. Di dalam program ini juga terdapat inisialisasi URL protokol CoAP, inisialisasi koneksi wifi *middleware*, perintah pengambilan waktu dari *middleware*, serta logika untuk merubah data raw dari sensor menjadi data PPM (*Part per Million*). Perbedaan kode program antara MQTT dan CoAP ada pada cara pengiriman datanya seperti yang sudah dijelaskan pada bab perancangan. Pada Node sensor CoAP, ketika NodeMCU berhasil mendapatkan nilai CO dari sensor, maka data akan langsung dikirimkan ke middleware. Berikut ditampilkan *pseudo*-code program untuk mengirim data sensor ke *middleware* pada tabel 5.3 dibawah ini.

Tabel 5.3 *Pseudo-code* program untuk mengirim data sensor CoAP

```
DEFINE function get_sensor_data()
  SET sensorValue = read carbon monixide data from sensor
  SET CO = convert RAW data readed from sensor
END
```

```
DEFINE function send_sensor_data()
  SET data = read carbon monixide data from sensor
  SET data = format data to JSON
  IF sensor value > 0.1 THEN
    DO send data over mqtt
  ELSE
    DO print error message
  END
END
```

Implementasi node sensor dapat dilihat dari penampakan fisik sensor setelah terpasang. Tampilan fisik node sensor ketika sudah berjalan dapat diperhatikan pada gambar 5.1 berikut ini.



Gambar 5.1 Implementasi node sensor

Keterangan pada gambar 5.1 :

1. Sensor MQ-7
2. Mikrokontroller NodeMCU

5.2 Implementasi Data center

Data center berfungsi untuk mengolah data yang berasal dari *middleware* dan menyimpannya dalam *database*. Dalam melakukan proses pengolahan data, peneliti membuat sebuah program menggunakan bahasa pemrograman Python untuk melakukan pengambilan data dari *database* dan mengolahnya menjadi status kondisi udara saat ini.

Pada direktori *home*, peneliti membuat sebuah program menggunakan nano sebagai editor dan diberi nama logic.py. *Library* PyMongo dimasukkan kedalam program agar dapat membaca data dari MongoDB menggunakan *query*. Berikut merupakan bagian *program* logic.py yang berfungsi membuat koneksi ke MongoDB, mengambil data menggunakan *query*, dan menjalankan logika untuk mengambil nilai sensor yang tersimpan dalam *database*. Pada program yang dibuat, terdapat 3 fungsi utama, yaitu fungsi untuk mengembalikan data terakhir

yang ada pada *database,* fungsi untuk mengembalikan data dalam satu hari yang dibagi berdasarkan jumlah jam setiap hari, dan fungsi untuk mengembalikan data dalam satu jam yang dibagi setiap lima menit.

5.2.1 Implementasi Pengambilan Data Terakhir

Salah satu fitur program yang dijalankan pada data center adalah mengambil data terakhir yang ada pada *database*. Data yang ada dalam database memiliki beberapa atribut pendukung seperti ID, topik, dan lain-lain. Program yang dibuat akan mengambil data berdasarkan atribut yang ada pada database, kemudian menyaring data tersebut hingga didapatkan nilai sensor yang terakhir masuk. Hal tersebut dilakukan dikarenakan data yang akan dikirim kepada *client* hanya data sensor saja. Berikut ditunjukkan pada tabel 5.4 mengenai *pseudo-code* potongan program logic.py yang bertugas untuk mengambil data terakhir.

Tabel 5.4 *Pseudo-code* pengambilan data terakhir

```
DEFINE function getLastData()
  DO connect to mongoDB
  DO get one data based on topic "home/CO" and sort descending
  FOR data from query
    DO find "CarbonMonoxide"
  END
  RETURN CO
```

5.2.2 Implementasi Pengambilan Data Setiap Jam

Fitur selanjutnya adalah mengambil data setiap jam dalam satu hari yang ada pada *database*. Program yang dibuat akan mengambil data berdasarkan waktu yang ada pada database, kemudian menyaring data tersebut hingga didapatkan semua nilai sensor dalam satu hari. Selanjutnya data yang disaring akan dihitung rata – rata nilainya dan dibagi berdasarkan jumlah jam dalam satu hari. Namun apabila tidak ada data yang masuk dalam jangka waktu satu jam, maka data yang kosong tersebut tidak akan dikirim kepada *client*. Hal tersebut dilakukan untuk meminimalisir tampilan yang tidak diperlukan pada aplikasi telepon pintar di sisi pengguna. Berikut ini disediakan pada tabel 5.5 tentang *pseudo-code* potongan program logic.py yang bertugas untuk mengambil data setiap jam.

Tabel 5.5 *Pseudo-code* pengambilan data setiap jam

```
DEFINE function getTableData(date)
  DO connect to mongoDB
  FOR hours in day
    DO get data based on time
    FOR data from query
      DO find "CarbonMonoxide"
    END
```

```
    IF data NOT empty
      DO get average CO data hourly
      SET avg_CO = average CO data
      SET time24 = time for every average CO data
    END
  END
  RETURN zip list avg_CO and time24
END
```

5.2.3 Implementasi Pengambilan Data Setiap Lima Menit

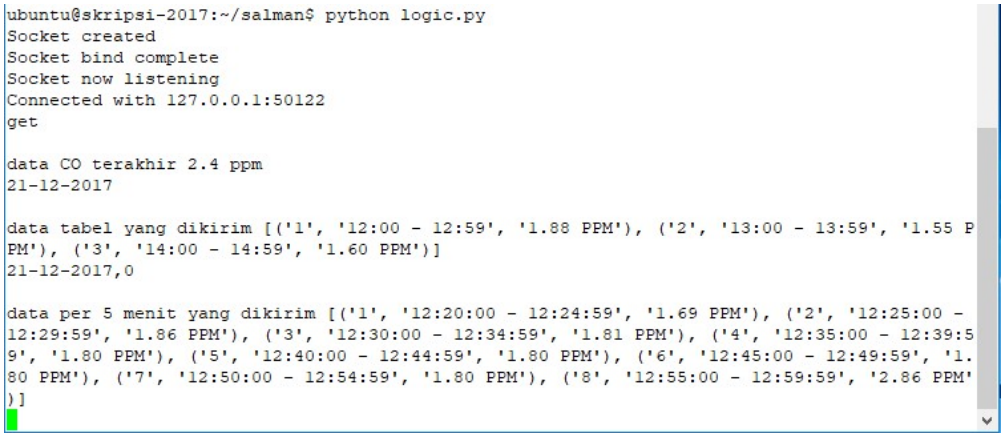
Fitur terakhir dalam program adalah mengambil data setiap lima menit dalam satu jam yang ada pada *database*. Program yang dibuat akan mengambil data berdasarkan waktu yang ada pada database, kemudian menyaring data tersebut hingga didapatkan semua nilai sensor dalam satu jam. Selanjutnya data yang disaring akan dihitung rata – rata nilainya dan dibagi lima menit. Namun apabila tidak ada data yang masuk dalam jangka waktu lima menit, maka data yang kosong tersebut tidak akan dikirim kepada *client*. Hal tersebut dilakukan untuk meminimalisir tampilan yang tidak diperlukan pada aplikasi telepon pintar di sisi pengguna. Berikut ini ditunjukkan *pseudo-code* potongan program logic.py yang bertugas untuk mengambil data setiap lima menit pada tabel 5.6 dibawah ini.

Tabel 5.6 *Pseudo-code* pengambilan data setiap lima menit

```
DEFINE function getDataMinutes(date_min)
  DO connect to mongoDB
  FOR 5 minutes in hour
    DO get data based on time
    FOR data from query
      DO find "CarbonMonoxide"
    END
    IF data NOT null
      DO get average CO data every 5 minutes
      SET avg_CO = average CO data
      SET time24 = time for every average CO data
    END
  END
  RETURN zip list avg_CO and time24
END
```

Implementasi data center dilakukan dengan melihat keluaran dari *program* yang dijalankan. Keluaran dari program dapat dilihat menggunakan aplikasi putty yang berfungsi untuk melakukan koneksi SSH ke data center. Ketika program dijalankan, pada jendela terminal menampilkan data yang akan dikirim oleh data

center kepada *client.* Tampilan keluaran program yang dijalankan dapat diperhatikan pada gambar 5.2 berikut ini.



Gambar 5.2 Implementasi data center

5.3 Implementasi Aplikasi Telepon Pintar

Aplikasi telepon pintar yang dirancang berdasarkan perangkat yang sering digunakan oleh pengguna, yaitu telepon pintar berbasis sistem operasi Android. Peneliti menggunakan Android Studio untuk membuat aplikasi ini. Seperti yang sudah dijelaskan pada bab perancangan, bahwa aplikasi ini memiliki 2 tampilan utama dan 1 tampilan tambahan. Aplikasi yang dirancang juga memiliki kemampuan untuk memberikan pemberitahuan berupa pemberitahuan *heads-up*, suara pemberitahuan *default*, dan getaran kepada perangkat pengguna apabila nilai sensor melebihi batas normal yang ditentukan.

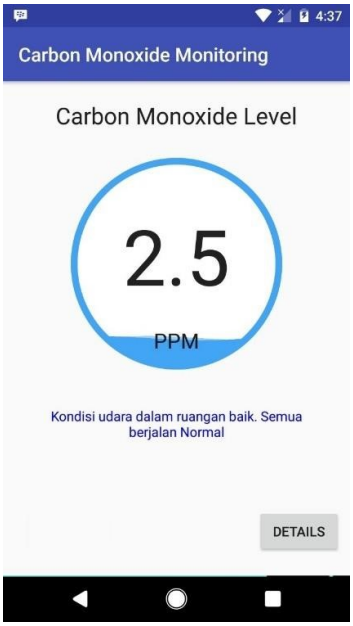
5.3.1 Implementasi Tampilan Utama

Tampilan utama yang pertama berfungsi untuk menampilkan data terakhir yang dikirimkan ke perangkat. Pada tampilan ini juga terdapat tulisan deskripsi yang warna dan tulisannya nya dapat berubah sesuai dengan nilai yang didapat dari data center. Pengguna disediakan tombol “Details” untuk menampilkan tampilan kedua. Berikut merupakan *pseudo-code* tampilan pertama yang akan ditunjukkan pada tabel 5.7 dibawah ini.

Tabel 5.7 *Pseudo-code* tampilan pertama

```
DEFINE function onCreate()
  DO connect to server
  DO receive message every 30 seconds
  DO create notification in background
END
DEFINE runnable mStatusChecker()
  DO function every seconds
  IF data received
    DO refresh view
  END
END
```

Dari *pseudo-code* program tampilan pertama yang ditunjukkan pada tabel 5.7, maka dapat dilihat tampilan pertama aplikasi pada gambar 5.3 berikut ini.



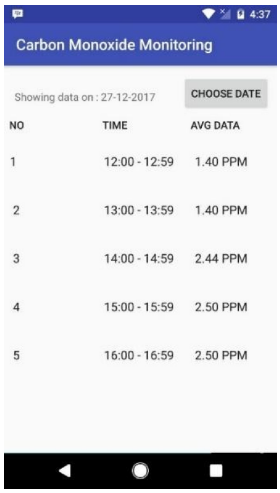
Gambar 5.3 Tampilan pertama aplikasi

Tampilan kedua memiliki fungsi untuk menampilkan detail data karbon monoksida yang terdapat dalam *database*. Pengguna dapat melihat detail data dalam satu hari yang akan disediakan dalam rangkuman rata – rata nilai setiap jam. Untuk mencapai hal ini, maka pengguna terlebih dahulu harus memilih tanggal yang ingin ditampilkan datanya. Pada tampilan kedua ini, awal mulanya hanya disediakan sebuah tombol "Choose Date" untuk menerima masukan dari pengguna berupa tanggal. Setelah pengguna selesai memilih tanggal, maka tampilan *grid* akan muncul pada kolom kosong dibawahnya. Tampilan grid akan menampilkan rata – rata data setiap jam yang didapatkan dalam satu hari. Berikut disediakan *pseudo-code* dari tampilan kedua pada tabel 5.8 dibawah ini.

Tabel 5.8 *Pseudo-code* tampilan kedua

```
DEFINE function onCreate()
  DO "Choose Date" set on click listener
  DO show DatePickerDialog
  IF date selected
    DO connect to server
    DO send request based input from user
    IF respond NOT null
      DO process respond
      DO update view
    ELSE
      DO update view
    END
    DO close connection
  END
END
```

Dari *pseudo-code* program tampilan kedua yang ditunjukkan pada tabel 5.8, maka dapat dilihat tampilan kedua aplikasi pada gambar 5.4 berikut ini.



Gambar 5.4 Tampilan kedua aplikasi

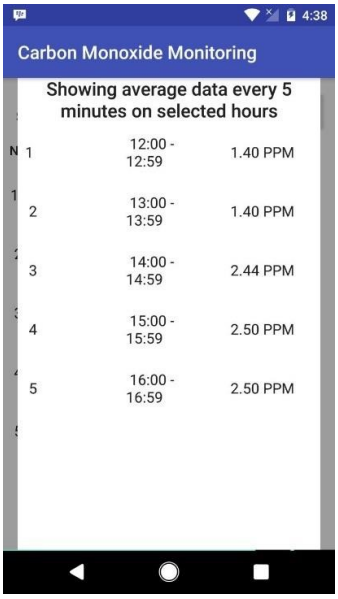
Tampilan tambahan merupakan tampilan yang berdiri diatas tampilan kedua. Fungsi dari tampilan tambahan ini adalah untuk menampilkan data yang lebih detail lagi, yaitu berupa nilai rata – rata dalam kurun waktu lima menit. Tampilan ini hanya akan tampil ketika pengguna telah memilih tanggal pada tampilan kedua sebelumnya. Hal ini dikarenaan untuk menampilkan tampilan tambahan ini pengguna harus memilih salah satu *grid* pada baris yang tersedia, sementara tampilan *grid* hanya akan tampil ketika pengguna sudah memilih tanggal. Tampilan *grid* pada tampilan kedua memiliki fungsi untuk menampilkan data

setiap jam dalam satu hari. Berikut merupakan *pseudo-code* dari tampilan tambahan yang akan disediakan pada tabel 5.9 dibawah ini.

Tabel 5.9 *Pseudo-code* tampilan tambahan pada aplikasi

```
IF date selected
  DO grid on item click listener
  DO connect to server
  DO send request based input from user
  IF respond NOT null
    DO process respond
    DO show popup
  ELSE
    DO show error popup
  END
  DO close connection
END
```

Dari *pseudo-code* program tampilan tambahan yang ditunjukkan pada tabel 5.9, maka dapat dilihat tampilan tambahan aplikasi pada gambar 5.5 berikut ini.



Gambar 5.5 Tampilan tambahan aplikasi