BAB 4 ANALISIS KEBUTUHAN DAN PERANCANGAN

Bab ini membahas persiapan testbed untuk menyiapkan kebutuhan yang digunakan dalam membangun lingkungan pengujian yang meliputi kebutuhan perangkat keras dan kebutuhan perangkat lunak. Selanjutnya dijelaskan perancangan sistem yang akan dibangun.

4.1 Kebutuhan Perangkat Keras

Perangkat keras yang dibutuhkan untuk membangun lingkungan pengujian dalam penelitian yaitu:

1. Laptop dengan spesifikasi:

   Merk : DELL-E6420

   Processor : Intel(R) Core™ i5-2520M CPU @2,50GHz

   RAM : 4 GB

   Sistem Operasi : Windows 10 Pro 64 bit & Ubuntu 17.04 64 bit

   Perangkat ini dibutuhkan untuk mengakses *middleware* secara remote (SSH) dan untuk mengakses web-app yang dapat menampilkan data dari *middleware*.

2. Raspberry Pi Zero

   Chipset : Broadcom BCM2835

   CPU : 1GHz ARM11

   RAM : 512MB LPDDR2 SDRAM

   Sistem Operasi : Raspbian Jessie Lite

   Perangkat ini digunakan sebagai *host* untuk *middleware*.

3. NodeMCU ESP8266

   Mikorkontroller ini nantinya dirangkai dengan modul DHT 11 / DHT 22 dan diprogram sehingga dapat mengirimkan data suhu dan kelembapan melalui protokol MQTT dan CoAP.

4. DHT 11 / DHT 22

   Modul sensor untuk membaca suhu dan kelembapan.

5. Server

   Perangkat ini digunakan untuk menjalankan aplikasi web sebagai *subscriber* yang membaca data dari *middleware* melalui protokol websocket.

6. *Wireless Adapter*

   Perangkat ini digunakan sebagai pendukung Raspberry Pi Zero agar dapat digunakan sebagai *access point*.

7. USB to LAN

   Karena pada Raspberry Pi Zero tidak memiliki port LAN maka dibutuhkan perangkat ini untuk mengkonversi port USB ke port LAN sehingga *middleware* dapat mengirim data ke server.

8. USB Hub

   Karena minimnya port USB pada Raspberry Pi Zero maka dibutuhkan tambahan USB port menggunakan USB Hub.

4.2 Kebutuhan Perangkat Lunak

Perangkat lunak yang dibutuhkan untuk mengimplementasikan sistem dan melakukan pengujian adalah sebagai berikut:

1. Wireshark

   Digunakan untuk menganalisis paket data yang telah direkam oleh *middleware* dan server.

2. Robomongo

   Untuk memonitoring data yang telah tersimpan dalam mongodb.

3. Tcpdump

   Program untuk merekam lalu lintas paket data baik di *middleware* maupun di server.

4. Nodejs v6

   Program yang berfungsi menjalankan program javascript dalam *middleware*.

5. Npm

   Adalah package manager yang digunakan untuk memasang package nodejs.

6. Firmware MCU

   Semacam sistem operasi yang berisi berbagai modul untuk nodeMCU. Modul – modul yang digunakan adalah bit, cjson, coap, mqtt, dht, file, gpio, net, node, rtc*time*, sntp, tmr, uart, dan wifi.

7. Mongodb

   Database yang digunakan untuk menyimpan data dari *middleware*.

8. ESplorer

   IDE yang digunakan untuk menulis program pada nodeMCU.

9. Redis

   Message broker yang digunakan dalam *middleware*.

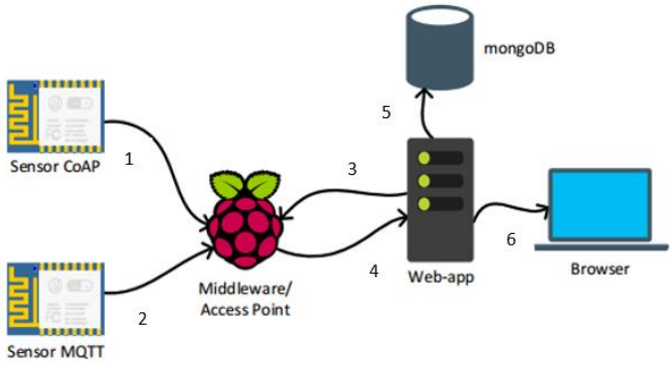
10. Nodemcu-flasher

    Aplikasi yang digunakan untuk flash firmware kedalam nodeMCU.

4.3 Aliran Data

Pertama node sensor harus terhubung dengan *middleware* supaya dapat mengirimkan data suhu dan kelembapan. Interval pengiriman data dilakukan setiap 30 detik sekali. Mekanisme pengiriman data sensor MQTT dengan melakukan publish ke *middleware* dengan topik home/barrack, sedangkan untuk data sensor CoAP dengan melakukan POST request ke *middleware* dengan topik home/kitchen.

Setiap kali node sensor mengirimkan data, *middleware* akan mengirimkan data tersebut secara *realtime* ke web-app menggunakan protokol websocket. Sebelum web-app dapat menerima data dari *middleware*, web-app harus terhubung terlebuh dahulu dengan *middleware* dan melakukan subscribe pada topik yang dikirimkan oleh sensor. Setelah data diterima oleh web-app, data tersebut akan disimpan dalam MongoDB.



Gambar 4.1 Aliran Data

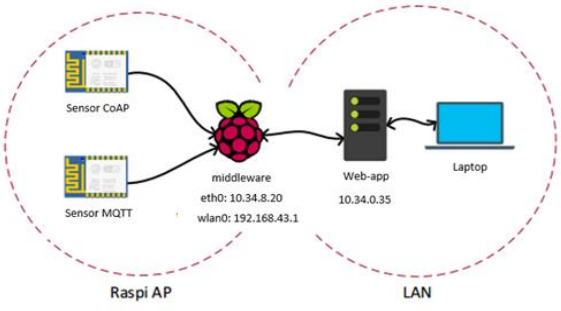
Keterangan pada Gambar 4.1 di atas adalah:

1. Sensor CoAP mengirimkan data suhu dan kelembapan ke *middleware* dengan topik home/kitchen setiap 30 detik sekali.
2. Sensor MQTT mengirimkan data suhu dan kelembapan ke *middleware* dengan topik home/barrack setiap 30 detik sekali.
3. Web-app subscribe topik home/kitchen dan home/barrack ke *middleware*.

4. Web-app menerima data suhu dan kelembapan untuk topik home/kitchen dan home/barrack.
5. Web-app menyimpan data kedalam database mongoDB.
6. Web-app diakses menggunakan browser pada laptop.

4.4 Topologi Jaringan

Agar sistem dapat berjalan sesuai dengan harapan, maka perlu dibuat topologi jaringan antar komponen dalam sistem. Topologi jaringan ini menggambarkan bagaimana sensor dapat terhubung dan mengirimkan data ke *middleware* kemudian diteruskan ke web-app. Topologi jaringan dapat dilihat pada Gambar 4.2.



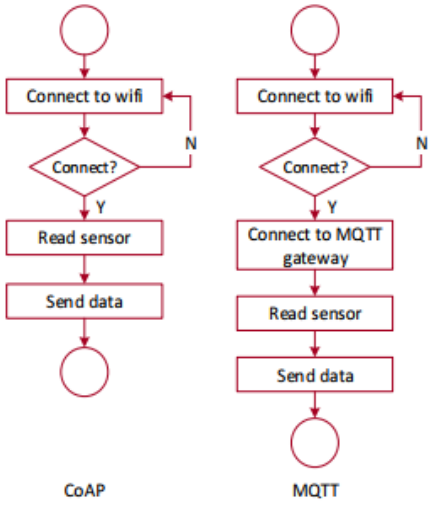
Gambar 4.2 Topologi Jaringan

Raspberry pi zero berperan sebagai *middleware* antara sensor dan web-app dan juga sebagai *access point*. Diperlukan *wireless adapter* yang difungsikan sebagi *access point* dan juga USB to LAN sehingga raspi zero memiliki 2 buah *interface* yaitu wlan0 dengan IP 192.168.43.1 dan eth0 yang terhubung dengan jaringan LAN dengan IP 10.34.8.20. Semua sensor akan berkomunikasi dengan *middleware* melalui *interface* wlan0. *Middleware* yang dikembangkan menggunakan tiga port: 3000 untuk websocket, 5683 untuk CoAP, dan 1883 untuk MQTT.

Subscriber yang digunakan dalam penelitian ini yaitu sebuah aplikasi web. Aplikasi ini dijalankan pada server dengan IP 10.34.0.35 yang terhubung dengan *interface* eth0 pada *middleware*. Kemudian aplikasi web dapat diakses oleh *browser* dengan alamat http://10.34.0.35:8000. Untuk pengiriman data, sensor MQTT mengirimkan data dengan topik home/barrack dengan melakukan *publish* ke alamat *middleware* mqtt://192.168.43.1:1883. Sedangkan sensor CoAP mengirimkan data dengan topik home/kitchen dengan melakukan POST *request* ke alamat coap://192.168.42.1:5683/r/home/kitchen. Aplikasi web sebagai *subscriber* menerima data dari *middleware* melalui protokol websocket dengan alamat ws://192.168.43.1:3000.

4.5 Node Sensor

Sebelum nodeMCU dapat digunakan sebagai node sensor pertama nodeMCU harus diinstal firmware dengan modul bit, cjson, coap, mqtt, dht, file, gpio, net, node, rtctime, sntp, tmr, uart, dan wifi menggunakan program Nodemcu-flasher. Kemudian ditulis program pada nodeMCU dengan menggunakan ESplorer. Node sensor yang digunakan adalah 5 nodeMCU dengan protokol MQTT dan 5 nodeMCU dengan protokol CoAP. NodeMCU berisi program yang dapat membaca nilai suhu dan kelembapan pada modul DHT setiap 30 detik sekali kemudian mengirimkan data tersebut ke *middleware* melalui protokol MQTT dan CoAP. Karena keterbatasan modul DHT, maka pada penelitian ini nilai dari suhu dan kelembapan diisi secara manual (*dummy*). Sebelum mengirimkan data suhu dan kelembapan, nodeMCU terlebih dahulu melakukan sinkronisasi waktu dengan *middleware* sehingga didapatkan *timestamp* untuk perhitungan *delay* nantinya. Aliran dari kode program yang dibuat digambarkan pada Gambar 4.3 berikut.



Gambar 4.3 Aliran Program NodeMCU

Berikut adalah kode program untuk nodeMCU dengan menggunakan protokol CoAP.

```
1   coap_url = 'coap://192.168.43.1:5683/r/home/kitchen'
2   coap_id = node.chipid()
3   cc = coap.Client()
4   temperature = 35
5   humidity = 90
6   wifi_SSID = "Pizero"
7   wifi_password = "raspizero"
8   wifi.setmode(wifi.STATION)
9   wifi.sta.config(wifi_SSID, wifi_password)
10  wifi.sta.connect()
11  time_between_sensor_readings = 30000
12  srv=net.createServer(net.TCP)
13  srv:listen(80,function(conn)
14      conn:on("receive", function(client,request)
```

```
        local    _,    _,    method,    path,    vars    =
string.find(request, "([A-Z]+) (.+)?(.+) HTTP");
        if(method == nil)then
            _, _, method, path = string.find(request, "([A-
Z]+) (.+) HTTP");
        end
        local _GET = {}
        if (vars ~= nil)then
            for k, v in string.gmatch(vars, "(%w+)=(%w+)&*")
do
                _GET[k] = v
                if ( v == "disconnect" ) then
                    tmr.stop(1)
                elseif (v == "restart") then
                    node.restart()
                end
            end
        end
        client:close()
        collectgarbage()
    end)
end)
function send_sensor_Data()
conn=net.createConnection(net.TCP, 0)
conn:on("connection",function(conn, payload)
    conn:send("HEAD / HTTP/1.1\r\n".."\r\n\r\n")
    end)
conn:on("receive", function(conn, payload)
    --time=string.sub(payload,string.find(payload,"Date:
")+6,string.find(payload,"Date: ")+34)
    --get_sensor_Data()
    cc:post(coap.CON, coap_url, cjson.encode({
       protocol = "coap",
        timestamp = {
            sec = sec,
            micro = usec,
            },
        topic = "home/kitchen",
        sensor = {
            tipe = "esp8266",
            index = coap_id,
            ip = ipaddr,
            module = "dht22"
        },
        humidity = {
            value = humidity,
            unit = "%"
        },
        temperature = {
            value = temperature,
            unit = "celcius"
        }
    }))

```

```
69      print("Going to deep sleep for
70  "..(time_between_sensor_readings/1000).." seconds")
71      end)
72      t = tmr.now()
73      conn:connect(8080,'192.168.43.1')
74  end
75  function loop()
76      if wifi.sta.status() == 5 then
77          ipaddr = wifi.sta.getip()
78          -- Stop the loop
79          tmr.stop(0)
80          print("Connected to CoAP")
81          print("  IP:".. ipaddr)
82          print("  URL: ".. coap_url)
83          print("  Client ID: ".. coap_id)
84          -- Get sensor data
85          tmr.alarm(1,    time_between_sensor_readings,    1,
86  function()
87            sntp.sync(('192.168.43.1'),
88                      function(sec, usec, server, info)
89                      print('timestamp', sec,usec, server)
90                      _G.sec = sec
91                      _G.usec= usec
92                      end)
93            send_sensor_Data()
94          end)
95      else
96          print("Connecting...")
97      end
98  end
    tmr.alarm(0, 1000, 1, function() loop() end)
```

Kode Program 4.1 NodeMCU CoAP

Penjelasan:

1. Baris 1 adalah deklarasi variabel coap_url untuk terhubung ke alamat coap pada *middleware*
2. Baris 2 adalah deklarasi variabel coap_id yang membaca ID dari chip nodeMCU
3. Baris 3 deklarasi variabel cc sebagai coap client
4. Baris 4-5 pengisian nilai suhu dan kelembapan secara manual
5. Baris 6-10 untuk terhubung ke *access point middleware*
6. Baris 11 adalah jeda pengiriman data sensor yaitu 30 detik
7. Baris 12-36 adalah proses pembuatan server
8. Baris 37-73 adalah untuk mekanisme pengiriman data dalam bentuk json
9. Baris 74-98 adalah proses looping pengiriman data sensor, pada baris 86 dilakukan sinkronisasi waktu menggunakan modul sntp sehingga diperoleh *time*stamp pengiriman data untuk menghitung nilai dari *delay* nantinya.

Berikut adalah kode program untuk nodeMCU dengan menggunakan protokol MQTT.

```
mqtt_broker_ip = "192.168.43.1"
mqtt_broker_port = 1883
mqtt_client_id = node.chipid()
m = mqtt.Client(mqtt_client_id, 120)
temperature = 25
humidity = 80
wifi_SSID = "Pizero"
wifi_password = "raspizero"
wifi.setmode(wifi.STATION)
wifi.sta.config(wifi_SSID, wifi_password)
wifi.sta.connect()
time_between_sensor_readings = 30000
srv=net.createServer(net.TCP)
srv:listen(80,function(conn)
    conn:on("receive", function(client,request)
        local    _,    _,    method,    path,    vars    =
string.find(request, "([A-Z]+) (.+)?(.+) HTTP");
        if(method == nil)then
            _, _, method, path = string.find(request, "([A-
Z]+) (.+) HTTP");
        end
        local _GET = {}
        if (vars ~= nil)then
            for    k,    v    in    string.gmatch(vars,
"(%w+)=(%w+)&*") do
                _GET[k] = v
                if ( v == "disconnect" ) then
                    tmr.stop(1)
                    m:close()
                elseif (v == "restart") then
                    node.restart()
                end
            end
        end
        client:close()
        collectgarbage()
    end)
end)
function send_sensor_Data()
conn=net.createConnection(net.TCP, 0)
conn:on("connection",function(conn, payload)
    conn:send("HEAD / HTTP/1.1\r\n".."\r\n\r\n")
    end)
conn:on("receive", function(conn, payload)
    time=string.sub(payload,string.find(payload,"Date:
")+6,string.find(payload,"Date: ")+34)
    m:publish("home/barrack",
        cjson.encode({
       protocol = "mqtt",
        timestamp = {
            sec = sec,
```

```
            micro = usec,
            },
        topic = "home/barrack",
        sensor = {
            tipe = "esp8266",
            index = mqtt_client_id,
            ip = ipaddr,
            module = "dht11"
        },
        humidity = {
            value = humidity,
            unit = "%"
        },
        temperature = {
            value = temperature,
            unit = "celcius"
        }
    }), 1, 0, function(conn)
        print("Going to deep sleep for
"..(time_between_sensor_readings/1000).." seconds")
        time=nil
        end)
    end)
    t = tmr.now()
    conn:connect(8080,'192.168.43.1')
    conn=nil
end
function loop()
    if wifi.sta.status() == 5 then
        ipaddr = wifi.sta.getip()
        -- Stop the loop
        tmr.stop(0)
        m:connect( mqtt_broker_ip , mqtt_broker_port, 0,
function(conn)
            print("Connected to MQTT")
            print("  IP: ".. ipaddr)
            print("  Port: ".. mqtt_broker_port)
            print("  Client ID: ".. mqtt_client_id)
            -- Get sensor data
            tmr.alarm(1, time_between_sensor_readings, 1,
function()
            --get_sensor_Data()
                send_sensor_Data()
                sntp.sync(('192.168.43.1'),
                    function(sec, usec, server, info)
                    print('timestamp', sec,usec, server)
                    _G.sec = sec
                    _G.usec= usec
                    end)
                --tmr.alarm(1,
time_between_sensor_readings, 1)
            --end)
            end)
        end )
```

```
106     else
107         print("Connecting...")
108     end
109 end
110 tmr.alarm(0, 1000, 1, function() loop()end)
```

Kode Program 4.2 NodeMCU MQTT

Penjelasan:

1. Baris 1 deklarasi variabel mqtt broker
2. Baris 2 deklarasi port mqtt
3. Baris 3 deklarasi client_id berdasarkan pada id chip nodeMCU
4. Baris 4 deklarasi variabel m untuk mqtt client
5. Baris 5-6 pengisian nilai suhu dan kelembapan secara manual
6. Baris 7-11 untuk terhubung ke *access point middleware*
7. Baris 12 deklarasi jeda pengiriman data sensor yaitu 30 detik
8. Baris 13-38 proses pembuatan server
9. Baris 39-78 proses pengiriman data dalam bentuk json
10. Baris 79-110 adalah proses looping pengiriman data sensor, pada baris 95 dilakukan sinkronisasi waktu menggunakan modul sntp sehingga diperoleh *time*stamp pengiriman data untuk menghitung nilai dari *delay* nantinya.