PENGENDALI LAMPU RUMAH BERBASIS ESP8266 DENGAN PROTOKOL MQTT

Rizki Priya Pratama [1],
Program Studi Teknik Mekatronika Politeknik Kota Malang
Email: rizkipriyap@gmail.com

ABSTRACTS : *The smart home is a system that provides comfort, safety, and energy savings, automatically. This system can be used to control electrical equipment such as lights, water pumps, air conditioners, etc. By replacing any conventional switch into a wireless switch, users can control this device from inside and outside the home automatically. The design of this lamp controller uses the ESP8266 module, relay driver, Tenstar Robot SMPS power supply, and TTP223 touch sensor. This control device is made as small as possible so that it can fit into inbow dus switch. The ESP8266 programming platform is Arduino by implementing the PubSubClient library for communication on the MQTT protocol. With the MQTT protocol, the system is created by giving orders via the topic "/order/(ruang)" and see the status of the lights with the topic "/ status/(ruang)". The command and status method is also applied to the Android application "Kontrol Rumah MQTT" so users can more easily. Besides, this control device is equipped with a schedule that can be set through the Android application. Based on the test results, the system has worked according to plan and can control all lights with the Android application, Kontrol Rumah MQTT. Messages "/order/(ruang)" from the Android application to the lamp controller can be received and sent back via "/status/(ruang)" properly. By using the MQTT broker address: broker.hivemq.com, the publish-subscribe process has 100% success. Meanwhile, testing the lamp schedule on the lamp control device can work correctly according to the schedule given.*
Keyword*: ESP8266, WIFI , Android, Smart Home, MQTT*

ABSTRAK: Rumah pintar adalah sebuah sistem yang memberikan kenyamanan, keamanan dan penghematan energi, secara otomatis pada sebuah rumah. Sistem ini dapat digunakan untuk mengendalikan peralatan-peralatan listrik seperti lampu, pompa air, pendingin ruangan dll. Dengan mengganti setiap saklar konvensional menjadi saklar *wireless*, pengguna dapat mengontrol perangkat ini dari dalam maupun dari luar rumah secara otomatis. Perancangan pengendali lampu ini menggunakan modul ESP8266, *driver relay, power supply* SMPS Tenstar Robot dan sensor sentuh TTP223. Perangkat pengendali ini dibuat sekecil mungkin sehingga dapat masuk pada *inbow dus* saklar. *Platform* pemrograman ESP8266 adalah Arduino dengan menerapkan *library* PubSubClient untuk komunikasi pada protokol MQTT. Dengan protokol MQTT, sistem dibuat dengan memberikan perintah melalui topik "/order/(ruang)" dan melihat status lampu dengan topik "/status/(ruang)". Metode perintah dan status tersebut juga diterapkan pada aplikasi Android "Kontrol Rumah MQTT" sehingga pengguna dapat lebih mudah. Selain itu, perangkat pengendali ini dilengkapi dengan jadwal yang dapat diatur melalui aplikasi Android. Berdasarkan hasil pengujian, sistem telah bekerja sesuai dengan perencanaan dan mampu mengendalikan lampu seluruhnya dengan aplikasi Android, Kontrol Rumah MQTT. Pesan "/order/(ruang)" dari aplikasi Android ke pengendali lampu ESP8266 dapat diterima dan dikirim kembali melalui "/status/(ruang)" dengan baik. Dengan menggunakan alamat MQTT *broker* : broker.hivemq.com, proses *publish - subscribe* mempunyai keberhasilan 100%. Sedangkan, pengujian jadwal lampu pada perangkat pengendali lampu dapat bekerja dengan benar sesuai jadwal yang diberikan.
Kata Kunci: ESP8266, WIFI , Android, Rumah pintar, MQTT

PENDAHULUAN

Sistem rumah pintar bertujuan untuk membuat sistem pemantauan, pengendalian, otomasi yang murah, aman dan berguna dalam penghematan energi [1]. Fungsi pemantauan pada rumah pintar digunakan untuk mengetahui apakah peralatan listrik rumah dalam kondisi terpakai atau tidak. Fungsi pengendalian digunakan untuk mengendalikan perangkat listrik / lampu hidup atau mati.

Penelitian mengenai sistem rumah pintar telah banyak dilakukan, antara lain (1) aplikasi rumah pintar dengan menggunakan modul ZigBee. Modul ZigBee ini digunakan untuk berkomunikasi antara komputer dengan mikrokontroller secara *wireless*. Mikrokontroller bertugas mengendalikan beberapa perangkat listrik. Monitoring dan pengendalian perangkat listrik dilakukan dengan media SMS dan *E-mail* [2]. (2) Pada penelitian [3], setiap perangkat listrik dan lampu dikendalikan melalui kontrol utama yaitu komputer *server*. (3) Sistem rumah pintar menggunakan IC ENC28J60, metode yang digunakan adalah IC ENC28J60 dijadikan mini *web server* untuk mengendalikan beberapa lampu [4]. Semua lampu terhubung secara langsung ke kontrol utama.

Pada artikel ini diusulkan modul ESP8266 untuk aplikasi rumah pintar dengan protokol MQTT. *Message Queue Telemetry Transport* (MQTT) adalah sebuah protokol yang dapat digunakan untuk menerapkan konsep rumah pintar. Protokol ini mempunyai sifat *light weighted message*, dapat bekerja dengan *bandwidth* rendah, *latency* yang tinggi dan berjalan pada layer aplikasi. Mekanisme protokol MQTT adalah *publish/subscribe* dimana pengiriman dan penerimaan pesan protokol MQTT

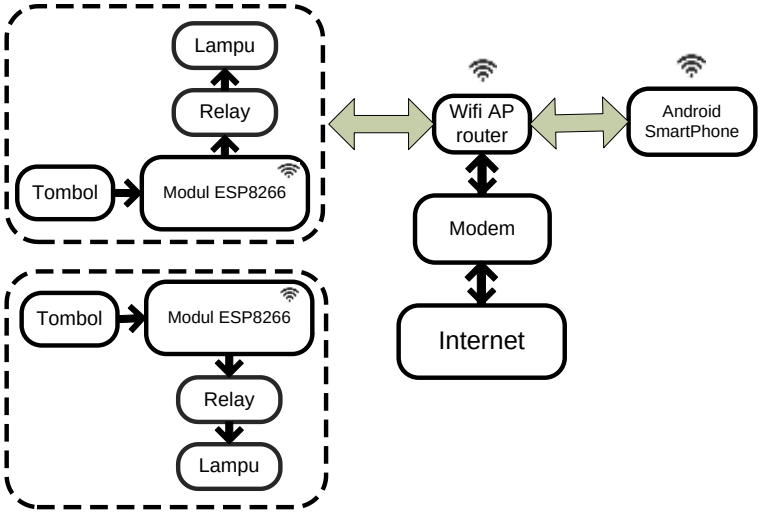
[1] Politeknik Kota Malang

berdasarkan pada topik – topik yang telah ditentukan. Dengan modul ESP8266 dan protokol MQTT, peralaan listrik dan lampu dapat terhubung secara *wireless* dari pengendali utama. Saklar-saklar listrik dapat diganti langsung dengan alat ini sebagai pengganti saklar konvensional.

PERANGKAT PENGENDALI LAMPU

Pengendali lampu merupakan perangkat yang dapat mengendalikan lampu dari jarak jauh dan dekat dengan aplikasi Android. Perangkat pengendali ini juga terdapat tombol atau sensor sentuh untuk mengendalikan lampu dari dekat. Selain itu, perangkat ini dapat bekerja secara otomatis sesuai dengan jadwal yang disimpan. Lampu dapat diatur kapan menyala dan mati sesuai dengan jadwal.

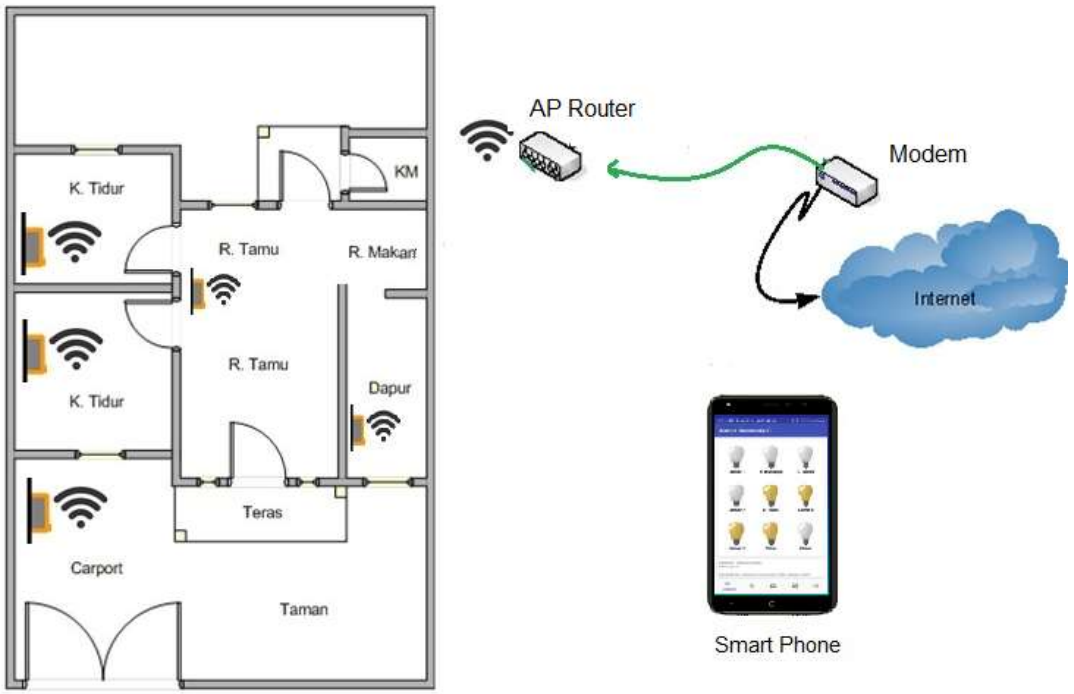
Blok diagram sistem dapat dilihat pada Gambar 1. Pengendali lampu pada sistem ini berjumlah 9 buah, dimana setiap perangkat terdapat sebuah mikrokontroller dan sebuah lampu. Pengendali lampu ditandai dengan garis putus-putus. Untuk menghubungkan semua perangkat pengendali lampu dan *SmartPhone* Android dibutuhkan sebuah AP router yang terhubung internet melalui modem. Setiap perangkat dapat terkoneksi ke *wifi*, mempunyai IP *address* dan mempunyai identitas sendiri-sendiri. Pengendali lampu harus terhubung jaringan internet, karena MQTT broker yang digunakan bersifat *online* yaitu broker.hivemq.com.



■ **Gambar 1**. Blok Diagram Sistem

Mikrokontroller yang dapat terhubung *wifi*, salah satunya adalah ESP8266. ESP8266 menawarkan solusi jaringan *wifi* yang cukup lengkap, yang dapat bekerja secara *stand-alone*, tanpa bantuan mikrokontroler lain. *Prosessor* dan *memory* pada modul ESP8266 memungkinkan untuk diintegrasikan dengan perangkat sensor, aktuator dan aplikasi lainnya melalui pin GPIO. Modul ini mempunyai fitur standar IEEE 802.11 b/g/n, WiFi *direct* (P2P), dan *AccesPoint soft-AP*. Kapasitas ESP8266 mempunyai RAM 81 Mb dan Flash memory 1Mb, kecepatan hingga 160 MHz, serta daya keluaran sebesar 19.5 dBm. [5]

Pengendali lampu ESP8266 dapat mengendalikan lampu dari tombol / sensor yang disentuh, dari perintah *SmartPhone* Android dan dari jadwal waktu. Jika tombol disentuh, maka lampu yang tadinya mati akan hidup dan sebaliknya, kemudian modul ESP8266 akan memberi status pada *SmartPhone* agar tampilan berubah sesuai kondisi lampu. *SmartPhone* yang diinstal Aplikasi dapat mengendalikan lampu pada ESP8266, men-*setting* konfigurasi ESP8266 dan menampilkan jadwal waktu yang dapat diatur oleh pengguna. Denah rumah untuk implementasi sistem pengendali lampu dapat dilihat pada Gambar 2.



■ **Gambar 2**. Denah Rumah pada Sistem

Perancangan *Cassing* Perangkat Pengendali

Cassing pengendali ESP8266 ini dibagi menjadi *cassing* luar dan dalam. *Cassing* luar terdiri atas penutup saklar Panasonic type WEJ78019W (*single*) serta chip kosong WEJ3020. Gambar *cassing* rangkaian ESP8266 dapat dilihat pada Gambar 3. *Cassing* ini digunakan untuk menahan dan melindungi rangkaian ESP8266.

■ **Gambar 3**. *Cassing* rangkaian pengendali ESP8266

Desain Komunikasi Perangkat Pengendali dengan SmartPhone

MQTT, atau *Message QueuingTelemetry Transport*, adalah protokol perpesanan yang memungkinkan beberapa perangkat saling berkomunikasi satu sama lain [6]. MQTT ditemukan pada tahun 1999. Saat ini, protokol ini populer untuk keperluan *Internet of Things*. MQTT berdasarkan idenya adalah setiap perangkat dapat *publish* atau *subscribe* topik [7]. Salah satu kelebihannya adalah data yang ditransmisikan pada jaringan *wifi* dengan menggunakan protokol MQTT, datanya tidak mudah rusak.

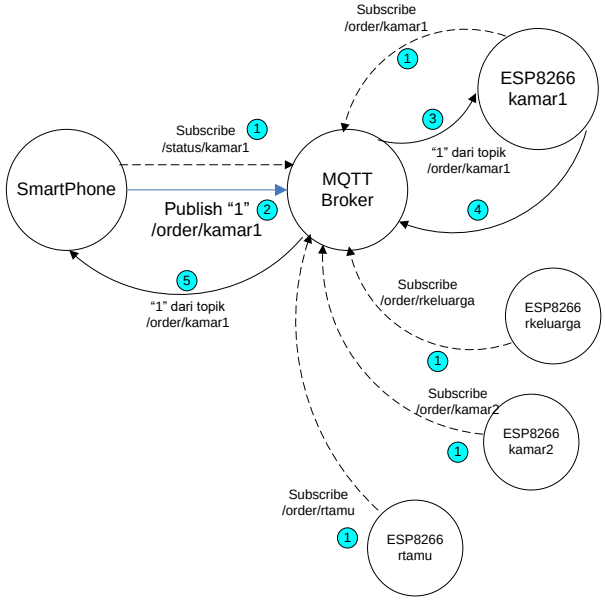
Dalam melaksanakan tugas dan fungsinya sebagai *client* MQTT, maka jalur komunikasi antar perangkat harus diatur agar tidak saling berbenturan. Topik yang digunakan untuk perintah adalah ”/order/(ruang)” sedangkan topik untuk mengetahui kondisi lampu atau respon dari ESP8266 adalah ”/status/(ruang)”. Berikut ini adalah gambaran urutan komunikasi MQTT untuk aplikasi ini, antara lain:

1. Proses ke 1:

Semua modul ESP8266 melakukan *subscribe* dengan topik ”/order/(ruang)”. Ruang ini merupakan nama masing-masing modul yang mewakili ruangan pada rumah. Contoh kamar1, kamar2, rtamu dan sebagainya. Sedangkan SmartPhone juga akan melakukan *subscribe* dengan topik ”/status/(ruang)” sebanyak jumlah ruang yang ada. Pada proses ini, semua *client* MQTT melakukan *subscribe* ke MQTT *broker*

2. Proses ke 2:
   Sebagai contoh ruang kamar 1 dalam keadaan padam. Pada saat Gambar Lampu kamar 1 ditekan pada *SmartPhone*, maka *SmartPhone* akan mengirimkan data atau *payload* ”1” pada topik ”/order/kamar1” ke MQTT *broker*.
3. Proses ke 3:
   MQTT *broker* mengirimkan *payload* ”1” ke perangkat yang sudah men-*subscribe* topik ”/order/kamar1”.
4. Proses ke 4:
   Modul ESP8266 yang telah men-*subscribe* ”/order/kamar1” akan menyalakan lampu dan mengirimkan *payload* ”1” dengan topik ”/status/kamar1” ke MQTT *broker*.
5. Proses ke 5:
   MQTT *broker* mengirimkan *payload* ”1” ke perangkat yang sudah men-*subscribe* topik ”/status/kamar1”. Dimana SmartPhone Android telah men- *subscribe*

Urutan dan proses komunikasi MQTT untuk aplikasi ini dapat dilihat pada Gambar 4.



■ **Gambar 4**. Desain Komunikasi MQTT.

Perancangan program ESP8266

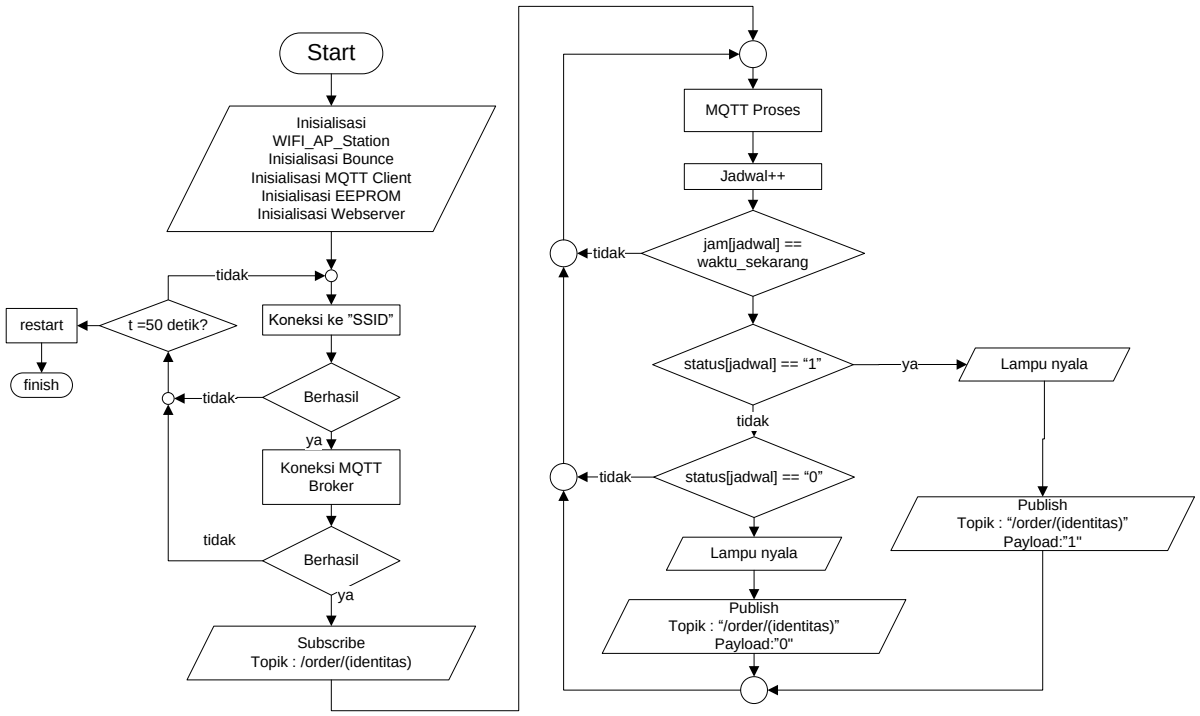
Perancangan program ESP8266 ini membutuhkan *library* Bounce2, ESP8266WiFi, PubSubClient, EEPROM dan ESP8266WebServer. Penjelasan fungsi dari *library* tersebut dapat dilihat pada Tabel 1.

■ **Tabel 1.** Keterangan Fungsi *Library* yang digunakan ESP2866

No	Paket Library	Fungsi
1	ESP8266WiFi	mengaktifkan Wifi baik untuk menjadi mode accespoint maupun mode station.
2	NTPClient	Mengambil data waktu NTP
3	PubSubClient	Mengkonfigurasi protokol MQTT
4	ESP8266WebServer	Melayani HTTP Get dari SmartPhone
5	Bounce2	Menghilangkan getaran saklar
6	EEPROM	Menyimpan data di eeprom

Pertama kali, ESP8266 bekerja pada 2 mode yaitu *accespoint* (AP) dan *station*. Mode *accespoint* ini memancarkan SSID bernama ”smarthome”, berfungsi untuk menghubungkan *SmartPhone* secara langsung ke ESP8266 untuk melakukan pengaturan nama dan password SSID yang akan dituju, nama ruang dan alamat MQTT broker. Sedangkan, mode *station* digunakan supaya

ESP8266 dapat terhubung ke *router* yang terhubung Internet. Ketika ESP8266 tidak menemukan *router* dengan SSID yang telah di setting, maka program akan kembali ke awal dan setelah 50 detik ESP8266 akan *restart*. *Flowchart* program tersebut dapat dilihat pada Gambar 5.



■ **Gambar 5**. *Flowcart* program ESP8266

```
while (WiFi.status() != WL_CONNECTED)
  {
    digitalWrite(ledPin1, !digitalRead(ledPin1));
    server.handleClient();
    cek_tombol();
    if (WiFi.softAPgetStationNum() == 0)
    {
      n++;
      if (n > 100)
      {
        ESP.restart();
      }
      delay(500);
      Serial.print("-");
      cek_tombol();
      tombol_reset_eeprom();
    }
    digitalWrite(ledPin1, HIGH);
  }
  WiFi.mode(WIFI_STA);
  Serial.print("\nIP address: ");
  Serial.println(WiFi.localIP());
```

■ **Gambar 6**. Potongan Program ESP8266 untuk mengatur mode WIFI

Apabila *router* dengan SSID yang telah di-*setting* ditemukan, maka mode *accespoint* akan dinonaktifkan, yang aktif hanya mode station saja. Gambar 6 merupakan potongan program untuk mengatur mode *accespoint* (AP) dan *station* yang telah dijelaskan diatas.

Mode *station* terjadi proses komunikasi antara ESP8266 dengan MQTT *broker*. ESP8266 men-*subscribe* topik "/order/(ruang)". Jika ada kiriman dengan topik "/order/(ruang)" dengan payload "0" atau "1", maka ESP8266 akan mem-*publish* topik "/status/(ruang)" dengan kondisi *payload* sesuai dengan kondisi lampu yang diperintah. Gambar 7 merupakan potongan program untuk menampung / menerima kiriman dengan topik " /order/(ruang)". Saat terjadi penekanan sensor / saklar, ESP8266 mem-*publish* topik "/order/(ruang)". Nilai 0 untuk mematikan lampu dan nilai 1 untuk menghidupkan lampu.

```
void ReceivedMessage(char *topic, byte *payload, unsigned int length)
if (strcmp(mqtt_topic_subs, topic) == 0)
  {
    if ((char)payload[0] == '0')
    {
      digitalWrite(ledPin, LOW);
      logic = false;
      client.publish(mqtt_topic_pubs, "0", true);
    }
    if ((char)payload[0] == '1')
    {
      digitalWrite(ledPin, HIGH);
      logic = true;
      client.publish(mqtt_topic_pubs, "1", true);
    }
  }
```

■ **Gambar 7**. Potongan program untuk menampung / menerima kiriman topik

Gambar 8 merupakan potongan program untuk melaksanakan eksekusi jadwal lampu. Data jadwal lampu disimpan dalam bentuk *array* jam_jadwal dan status_jadwal. *Array* jam_jadwal merupakan data interger yang menampung waktu dalam satuan menit, sedangkan *array* status_jadwal merupakan data integer yang berisi logika 0 atau 1. Setiap periode, program akan melakukan pengecekan apakah nilai jam_jadwal sama dengan waktu sekarang. Waktu sekarang adalah waktu yang didapatkan dari NTP *Server*. Jika nilainya sama, maka program akan mengecek apakah status lampu menyala atau padam. Kemudian program akan mengirim status tersebut ke jaringan dan diterima oleh *SmartPhone*.

```
waktu_milis++;
if ((waktu_milis % 2000000) == 0)
{
  for (loop_jadwal = 1; loop_jadwal < jumlah_waktu_on_off; loop_jadwal++)
  {
    if ((jam_jadwal[loop_jadwal] == GetTimeNtpServer()))
       {
      if (status_jadwal[loop_jadwal] == 1)
      {
        digitalWrite(ledPin, HIGH);
        logic = true;
        client.publish(mqtt_topic_subs, "1", true);
      }
      else if (status_jadwal[loop_jadwal] == 0)
      {
        digitalWrite(ledPin, LOW);
        logic = false;
        client.publish(mqtt_topic_subs, "0", true);
      }
    }
  }
}
```

■ **Gambar 8**. Potongan program ESP8266 untuk jadwal lampu

Perancangan Aplikasi Android "Kontrol Rumah MQTT"

Perancangan program Android untuk mengendalikan ESP8266 dibuat dengan aplikasi Android Studio dengan bahasa java. Aplikasi Android ini dapat menampilkan status *on/off* lampu seluruh ruangan, status koneksi MQTT *broker*, setting *client* ESP8266, setting alamat MQTT Broker dan pengaturan jadwal lampu *on/off*. Saat aplikasi dibuka, status *on/off* lampu seluruh ruangan ditampilkan. Tampilan awal aplikasi Android untuk menampilkan status *on/off* lampu seluruh ruangan dapat dilihat pada Gambar 9a. Jika Gambar Lampu ditekan sebentar di salah satu ruangan maka *SmartPhone* ini akan mengirimkan perintah ke ESP8266. Jika Gambar Lampu ditekan lama maka akan tampil pengaturan jadwal lampu *on/off* yang dapat dilihat pada Gambar 9b.

Tampilan awal / pengendali lampu
a

Tampilan pengaturan jadwal lampu
b

■ **Gambar 9**. Tampilan aplikasi “Kontrol Rumah MQTT”

Data jadwal lampu *on/off* ini disimpan pada *database* SQLIte dengan isi tabel berupa nomer, lampu_waktu, lampu_ruang, dan lampu_kondisi. Saat penekanan tombol lampu pada aplikasi, misalkan ruang tamu, maka *database* hanya menampilkan data berdasarkan lampu_ruang yang diinginkan. Potongan program ini dapat dilihat pada Gambar 10. Sedangkan untuk menambah jadwal lampu, pengguna dapat menekan tombol Tambah pada Gambar 9b. Potongan program untuk menambah jadwal lampu dapat dilihat pada Gambar 11.

```
try {
      cursor = db.rawQuery("SELECT * FROM lampu_db WHERE lampu_ruang ='"+LAMPU_RUANG+"'
          ORDER BY lampu_waktu ASC",null);
      daftar = new String[cursor.getCount()];
```

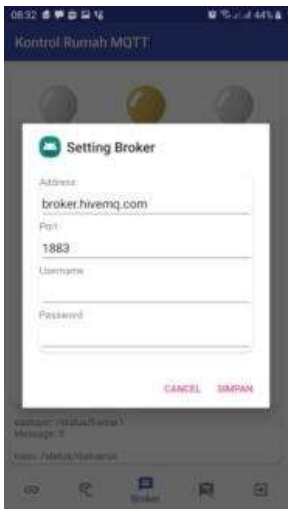
■ **Gambar 10**. Perintah *database* SQLIte untuk menampilkan isi jadwal waktu.

```
try {
    String sql_lampu = "INSERT INTO lampu_db (lampu_waktu, lampu_ruang, lampu_kondisi)
        VALUES ('" +totalSelectedMinute+"','" +LAMPU_RUANG+"', '0');";
    db.execSQL(sql_lampu);
```

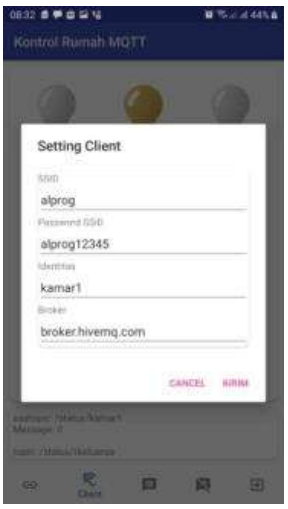
■ **Gambar 11**. Perintah *database* SQLIte untuk menambah jadwal waktu.

Komunikasi antara *SmartPhone* dengan ESP8266 terjadi dimana *SmartPhone* akan mem-*publish* topik ”/order/(ruang)” dengan *payload* ”0” atau ”1”. Pengiriman *payload* ”0” atau ”1” tergantung dari kondisi lampu. Jika lampu sudah menyala, maka *payload* ”0” yang dikirim agar ESP8266 mematikan lampu, begitu pula sebaliknya.

Tampilan Setting Broker digunakan untuk mengatur alamat MQTT *broker* yang digunakan. Tampilan ini dapat dilihat pada Gambar 12a. Pada aplikasi ini, MQTT *broker* yang digunakan adalah broker.hivemq.com, dengan port 1883. Sedangkan tampilan Setting Client digunakan untuk mengatur SSID, password SSD, identitas / ruang dan MQTT broker yang harus sama dengan MQTT broker SmartPhone. Tampilan Setting Client dapat dilihat pada Gambar 12b



Tampilan pengaturan alamat *broker*
a



Tampilan pengaturan untuk pengendali lampu
b

■ **Gambar 12**. Tampilan Setting *Broker* dan Setting Client.

Perancangan rangkaian pengendali ESP8266.

Rangkaian pengendali ESP8266 terdiri atas *power supply*, rangkaian *driver relay*, IC ESP8266 dan sensor sentuh TPP223. Gambar komponen untuk rangkaian pengendali dapat dilihat pada Gambar 13.

Modul ESP8266 Sensor TTP223 Relay

■ **Gambar 13**. Komponen pengendali

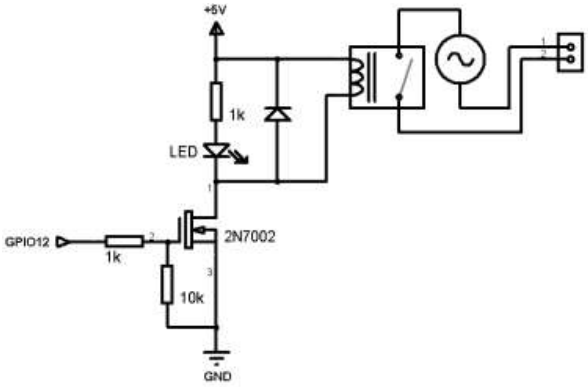
Pada perancangan *driver relay*, *relay* yang digunakan mempunyai arus 30mA dan tegangan 5 volt. Mosfet yang digunakan adalah E-mosfet 2N7002 type N. Mosfet ini mempuyai IDmaks = 300mA, VGSon = 2,5 Volt dengan RDSon 2,5 - 4,5 ohm[8]. Karakteristk tranfer menunjukkan bahwa pada VGS = 3 arus arus drain (ID) adalah 90mA, sehingga dengan arus 90mA ini cukup untuk mengendalikan *relay* ini.

Tegangan keluaran maksimum terminal GPIO12 ESP8266 adalah 3,3 volt. Oleh karena itu, dibutuhkan rangkaian pembagi tegangan untuk mengubah menjadi tegangan 3 volt. Perhitungan pembagi tegangan dapat dilihat pada persamaan

$$V_{out} = \frac{Rb}{Ra + Rb} \cdot V_{in} \quad \text{.....(1)}$$

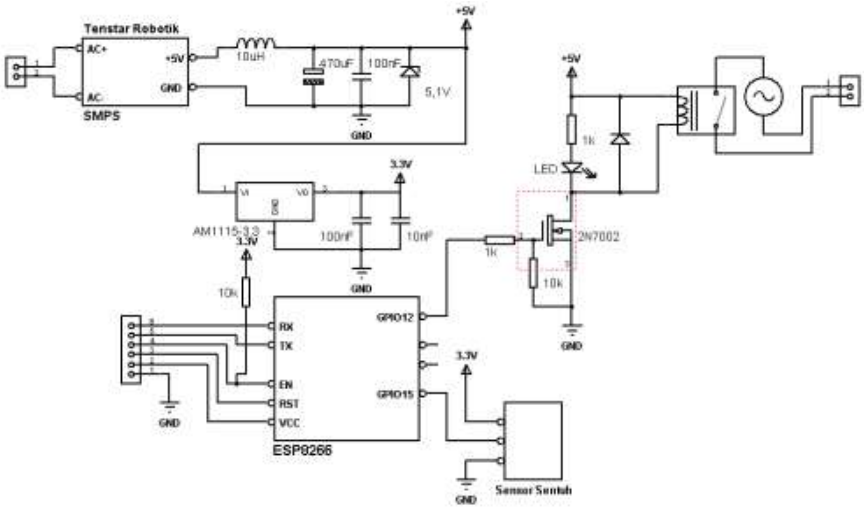
$$3volt = \frac{Rb}{Ra + Rb} x3{,}3volt$$

Persamaan (1) menunjukkan jika *Rb* = 10 kΩ, maka didapatkan *Ra* = 1 kΩ, sehingga rangkaian pengendali *relay* menjadi seperti Gambar 14.



■ **Gambar 14**. Rangkaian *Driver Relay*

Power supply yang digunakan berukuran kecil dan memiliki tingkat noise yang rendah yaitu SMPS Tenstar Robot. Tegangan keluaran SMPS Tenstar Robot adalah 5 volt, sehingga dibutuhkan IC regulator AMS1117 untuk menurunkan tegangan catu menjadi 3.3 volt sebagai catu daya ESP8266. Skema keseluruhan rangkaian pengendali ESP8266 dapat dilihat pada Gambar 15 sedangkan foto rangkaian dapat dilihat pada Gambar 16.



■ **Gambar 15**. Skema rangkaian pengendali ESP8266

■ **Gambar 16**. Foto perangkat ESP8266

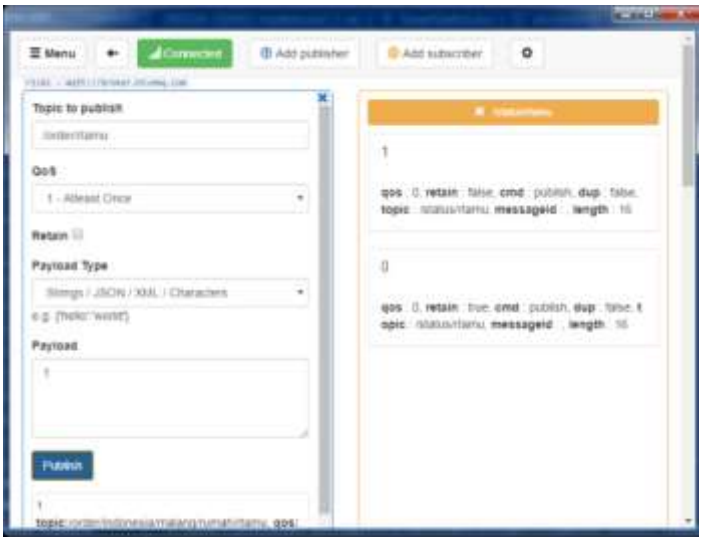
PENGUJIAN DAN ANALISIS SISTEM

Pengujian sistem dilakukan untuk mengetahui apakah sistem dapat bekerja sesuai dengan tujuan perancangan. Pengujian sistem dibagi menjadi beberapa bagian yaitu pengujian perangkat ESP8266 dengan MQTT Box, Pengujian perangkat ESP8266 dengan aplikasi "Kontrol Rumah MQTT", Pengujian waktu respon perangkat pengendali ESP8266 dan pengujian jadwal lampu pada ESP8266.

Pengujian perangkat ESP8266 dengan MQTT Box

Pengujian perangkat ESP8266 dengan aplikasi MQTT Box dilakukan untuk mengetahui apakah perangkat ESP8266 dapat dikendalikan / dikontrol melalui komunikasi protokol MQTT dengan baik. Selain itu, juga untuk mengetahui apakah perangkat ini dapat mengirimkan kembali status kondisi lampu

pada MQTT Box. Gambar 17 merupakan aplikasi MQTT Box saat mengirim topik "/order/rtamu" dengan payload "0" dan "1" ke ESP8266 dan hasil kiriman balik status dari ESP8266 dengan topik "/status/rtamu dengan payload "0" dan "1".



■ **Gambar 17**. MQTT Box

Pengujian ini dilaksanakan untuk semua ruangan sehingga didapatkan hasil yang dapat dilihat pada Tabel 2. Kolom Topik Order merupakan topik yang dikirimkan oleh aplikasi MQTT box ke semua perangkat ESP8266, kemudian kolom Topik Status adalah topik / respon balasan dari perangkat yang telah men-*subscibe* sesuai dengan Topik Order, sedangkan kolom *Payload* adalah data untuk mengatur kondisi lampu *on/off* .

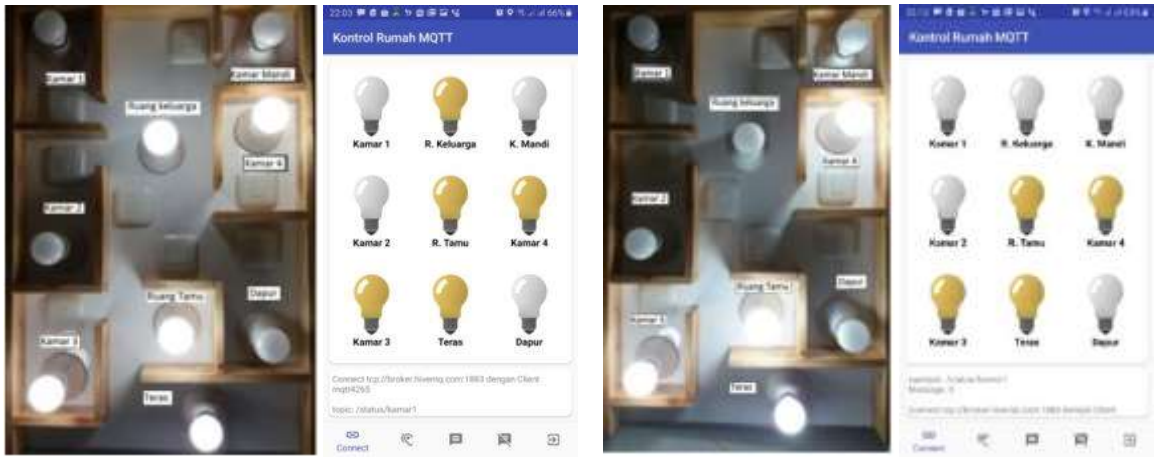
■ **Tabel 2.** Tabel pengujian perangkat ESP8266 dengan MQTT Box

No	Topik order	Payload	Topik Status	Payload	Kondisi
1	/order/ kamar1	”1”	/status/ kamar1	”1”	Terkirim
2	/order/ kamar2	”0”	/status/ kamar2	”0”	Terkirim
3	/order/ kamar3	”1”	/status/ kamar3	”1”	Terkirim
4	/order/ kamar4	”1”	/status /kamar4	”1”	Terkirim
5	/order/ rkeluarga	”0”	/status/rkeluarga	”0”	Terkirim
6	/order/ kmandi	”0”	/status /kmandi	”0”	Terkirim
7	/order/ rtamu	”0”	/status /rtamu	”0”	Terkirim
8	/order/ teras	”1”	/status /teras	”1”	Terkirim
9	/order/ dapur	”0”	/status /dapur	”0”	Terkirim

Dari data hasil pengamatan yang dilakukan terlihat bahwa setiap perangkat mampu untuk menerima Topik Order beserta *payload* datanya tanpa ada kerusakan data. Dan mengirimkan Topik Status kembali ke aplikasi MQTT box dengan persentasi keberhasilan 100%.

Pengujian perangkat ESP8266 dengan aplikasi "Kontrol Rumah MQTT"

Pengujian kendali perangkat ESP8266 ini menggunakan aplikasi Android, Kontrol Rumah MQTT. Pengujian ini menggunakan maket rumah yang di setiap ruangannya masing-masing terdiri dari perangkat pengendali ESP8266 dan lampu. Tujuan dari pengujian ini adalah untuk mengetahui apakah seluruh rangkaian ESP8266 yang sudah terpasang lampu dapat dikendalikan dengan baik. Gambar 18 menyajikan pengujian kontrol perangkat pada maket dengan aplikasi "Kontrol Rumah MQTT".

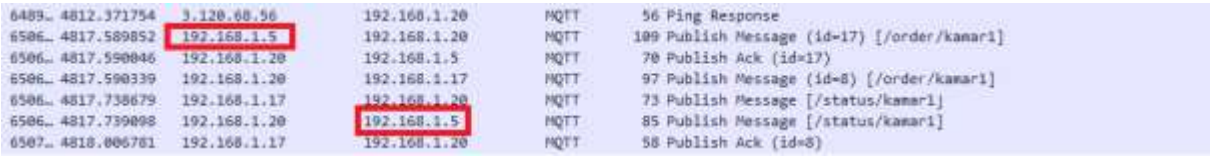


■ **Gambar 18**. Pengujian perangkat ESP8266 dengan aplikasi "Kontrol Rumah MQTT"

Gambar 18 menunjukkan bahwa ketika gambar lampu pada aplikasi "Kontrol Rumah MQTT" dihidupkan maka lampu pada maket juga hidup, begitu sebaliknya. Hasil pengamatan dari percobaan menggunakan kesembilan pengendali lampu ESP8266 pada maket rumah dapat disimpulkan bahwa persentase seluruh lampu dapat dikendalikan dari aplikasi "Kontrol Rumah MQTT" adalah 100%.

Pengujian waktu respon perangkat pengendali ESP8266

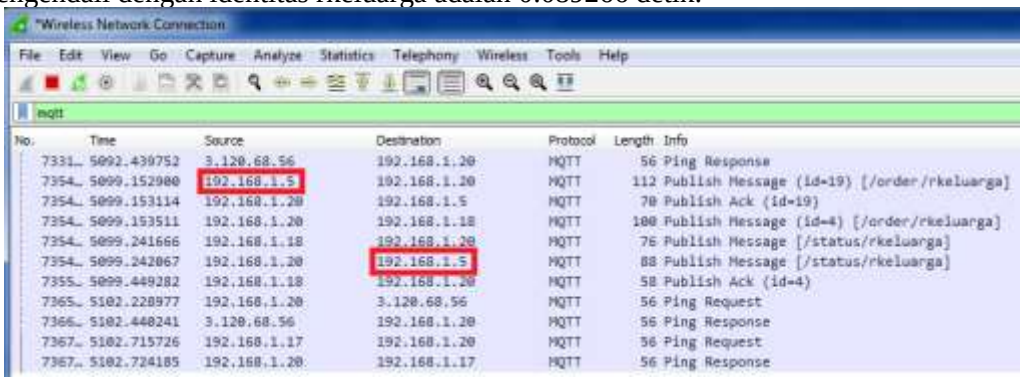
Pengujian ini bertujuan untuk mengetahui berapa waktu yang diperlukan saat gambar lampu pada aplikasi "Kontrol Rumah MQTT" ditekan dan diterima kembali status dari perangkat pengendali. Pengujian ini menggunakan aplikasi *wireshark* dengan melakukan *capture interface* jaringan yang telah dibuat. IP *broker* yang digunakan adalah 192.168.1.20 (komputer), IP *SmartPhone* Android adalah 192.168.1.5 sedangkan IP perangkat pengendali dengan identitas kamar1 adalah 192.168.1.17.



6489…	4812.371754	3.120.68.56	192.168.1.20	MQTT	56 Ping Response
6506…	4817.589852	192.168.1.5	192.168.1.20	MQTT	109 Publish Message (id=17) [/order/kamar1]
6506…	4817.590046	192.168.1.20	192.168.1.5	MQTT	70 Publish Ack (id=17)
6506…	4817.590339	192.168.1.20	192.168.1.17	MQTT	97 Publish Message (id=8) [/order/kamar1]
6506…	4817.738679	192.168.1.17	192.168.1.20	MQTT	73 Publish Message [/status/kamar1]
6506…	4817.739008	192.168.1.20	192.168.1.5	MQTT	85 Publish Message [/status/kamar1]
6507…	4818.006781	192.168.1.17	192.168.1.20	MQTT	58 Publish Ack (id=8)

■ **Gambar 19**. Pengujian waktu respon dengan *wireshark* untuk pengendali dengan identitas kamar1

Gambar 19 menunjukkan pengujian ketika Gambar Lampu kamar 1 pada aplikasi "Kontrol Rumah MQTT" ditekan, maka *smartphone* Android (192.168.1.5) akan mengirimkan topik "/order/kamar1" ke *broker* (192.168.1.20) dan selanjutnya *broker* meneruskannya ke perangkat pengendali (192.168.1.17). Setelah itu, perangkat pengendali merespon dengan mengirimkan topik "/status/kamar1" ke *broker* dan *broker* meneruskan kembali ke S*martPhone* Android. Waktu yang diperlukan dari *publish* topik "/order/kamar1" hingga mendapatkan topik "/status/kamar1" adalah 0.1492 detik. Gambar 20 merupakan Gambar Pengujian waktu respon untuk pengendali dengan identitas rkeluarga menggunakan aplikasi *wireshark*. Dari pengujian tersebut didapatkan waktu respon untuk pengendali dengan identitas rkeluarga adalah 0.089266 detik.



No.	Time	Source	Destination	Protocol	Length	Info
7331…	5092.439752	3.120.68.56	192.168.1.20	MQTT	56	Ping Response
7354…	5099.152980	192.168.1.5	192.168.1.20	MQTT	112	Publish Message (id=19) [/order/rkeluarga]
7354…	5099.153114	192.168.1.20	192.168.1.5	MQTT	70	Publish Ack (id=19)
7354…	5099.153511	192.168.1.20	192.168.1.18	MQTT	100	Publish Message (id=4) [/order/rkeluarga]
7354…	5099.241666	192.168.1.18	192.168.1.20	MQTT	76	Publish Message [/status/rkeluarga]
7354…	5099.242067	192.168.1.20	192.168.1.5	MQTT	88	Publish Message [/status/rkeluarga]
7355…	5099.449282	192.168.1.18	192.168.1.20	MQTT	58	Publish Ack (id=4)
7365…	5102.228977	192.168.1.20	3.120.68.56	MQTT	56	Ping Request
7366…	5102.440241	3.120.68.56	192.168.1.20	MQTT	56	Ping Response
7367…	5102.715726	192.168.1.17	192.168.1.20	MQTT	56	Ping Request
7367…	5102.724185	192.168.1.20	192.168.1.17	MQTT	56	Ping Response

■ **Gambar 20**. Pengujian waktu respon dengan *wireshark* untuk pengendali dengan identitas keluarga

Hasil pengujian waktu respon untuk perangkat pengendali yang lain dapat dilihat pada Tabel 3. Rata-rata waktu respon perangkat pengendali adalah 0.0832 detik. Jadi pada saat gambar tombol lampu pada aplikasi "Kontrol Rumah MQTT" ditekan , maka warna tombol tersebut akan berubah seketika dengan waktu dibawah 1 detik, respon dari topik "/status/(ruang).

■ **Tabel 3.** Tabel pengujian waktu respon perangkat pengendali

No	Ruang / identitas	Waktu Respon (s)
1	kamar1	0,1492
2	kamar2	0,0765
3	kamar3	0,0585
4	kamar4	0,1185
5	rkeluarga	0,0892
6	kmandi	0,0341
7	rtamu	0,0585
8	teras	0,1065
9	dapur	0,0585
	Rata-rata	0.0832

Pengujian Jadwal Lampu pada ESP8266 dengan aplikasi Kontrol Rumah MQTT

Pengujian ini digunakan untuk mengetahui apakah jadwal yang telah dikirimkan ke perangkat ESP8266 dapat bekerja sesuai dengan waktu dan kondisi lampu yang telah diberikan. Aplikasi Kontrol Rumah MQTT diatur sesuai dengan waktu yang diinginkan seperti pada Gambar 21. Gambar 21 menunjukkan bahwa jam 13:04 lampu dikendalikan untuk menyala dan pada jam 13:09 lampu dikendalikan untuk padam.

■ **Gambar 21**. Tampilan setting jadwal lampu untuk Kamar 1

Hasil pengujian ini dipantau dengan mengamati keadaan lampu pada maket rumah untuk kamar 1 sesuai dengan waktu yang telah dijadwalkan. Dari pengujian tersebut, hasilnya dapat dilihat pada Tabel 4.

■ **Tabel 4.** Tabel pengujian jadwal lampu

No	Waktu	Kondisi	Hasil
1	13:04	"1"	Lampu Menyala
2	13:05	"0"	Lampu Mati
3	13:06	"1"	Lampu Menyala
4	13:07	"0"	Lampu Mati
5	13:08	"1"	Lampu Menyala
6	13:09	"0"	Lampu Mati
7	13:10	"1"	Lampu Menyala
8	13:11	"0"	Lampu Mati

Hasil pengamatan dari pengujian diatas dapat disimpulkan bahwa waktu dan kondisi lampu antara hasil pengujian Tabel 3 dengan data tampilan setting jadwal lampu pada Gambar 21 adalah sama.

KESIMPULAN

Berdasarkan hasil penelitian dan pengujian maka dapat diambil beberapa kesimpulan antar lain:

1. Sistem pengendali lampu dengan protokol MQTT dapat berjalan dengan baik. Pesan "/order/(ruang)" dari aplikasi Android "Kontrol Rumah MQTT" ke pengendali lampu ESP8266 dapat diterima dan dikirim kembali melalui "/status/(ruang)" dengan baik tanpa ada kerusakan data.
2. Seluruh pengendali lampu ESP8266 pada maket rumah dapat dikendalikan dari aplikasi "Kontrol Rumah MQTT", seolah-olah lampu pada maket secara langsung dikendalikan oleh aplikasi "Kontrol Rumah MQTT"
3. Rata-rata waktu respon perangkat pengendali dari menerima pesan "/order/(ruang)" dan mengirimkan kembali ke *SmartPhone* Android dengan topik "/status/(ruang)" adalah 0.0832 detik.
4. Aplikasi Android "Kontrol Rumah MQTT" dapat mengirimkan data jadwal yang telah diatur oleh pengguna ke perangkat pengendali lampu dengan benar dan jadwal lampu pada perangkat dapat bekerja secara otomatis sesuai jadwal yang diberikan.

DAFTAR PUSTAKA

[1] Horo, Nishant,"*Domotics using internet of things*", in Proceedings of 40th IRF International Conference, Chennai, India: 2016, pp. 44 – 47.

[2] Nausheen Belim, et al. "*Automate and Secure Your Home Using ZigBee Technology*", in International Journal of Innovative Research in Computer and Communication Engineering Vol. 1, Issue 1, 2013.

[3] Kumar. Manu, Ekta, Agarwal. Shruti, Gaur dan Gupta, Yashdeep, "*Internet Based Home Automation. International Journal of Research and Development Organization*", in Journal of Electronics and Computer Science Vol. 2, Issue 8, Aug. 2015.

[4] Pratama, Rizki Priya. "*Metode Pemrograman Frame pada Modul ENC28J60 untuk Aplikasi miniwebserver AVR*" . In SITIA 2013-Smart and Intelligent Technology Application for Distributed Generation Systems.

[5] ESP8266 – 12E Datasheet, https://www.adafruit.com/datasheets/ ESP8266_Specifications_English.pdf

[6] Mqtt-broker Documentation, https://media.readthedocs.org/pdf/mqtt-broker/latest/mqtt-broker.pdf

[7] MQTT A Practical Guide, https://www.myelectronicslab.com/wp-content/uploads/2016/03/ Getting-Started-With-MQTT-My-Electronics-Lab.pdf

[8] Malvino A.P., "*Mosfet*", in Prinsip-prinsip Elektronika Jilid 1 , Jakarta : Penerbit Erlangga , 1985.