LAMPIRAN 1 : *Source Code*

1.1. nodeMCU_LED_TLS.ino

```
1   #include "FS.h"
2   #include <ESP8266WiFi.h>
3   #include <PubSubClient.h>

4   #define LED_RED D1
5   #define LED_GREEN D2
6   #define LED_BLUE D3
7   #define WIFI_SSID "NAMA SSID WIFI"
8   #define WIFI_PASS "PASSWORD WIFI"
9   #define MQTT_SERVER "IP MQTT BROKER"
10  #define MQTT_PORT "PORT MQTT BROKER"
11  #define MQTT_USERNAME "TOKEN JWT"
12  #define MQTT_PASSWORD " "

13  int ledPin = LED_BUILTIN;
14  boolean redStatus, greenStatus, blueStatus;
15  WiFiClientSecure espClient;
16  PubSubClient client(MQTT_SERVER, MQTT_PORT, callback,
    espClient);
17  void setup() {
18    Serial.begin(115200);
19    pinMode(LED_RED, OUTPUT);
20    pinMode(LED_GREEN, OUTPUT);
21    pinMode(LED_BLUE, OUTPUT);
22    pinMode(ledPin, OUTPUT);
23    setup_wifi();
24    delay(500);
25    setup_certificates();
26  }

27  void loop() {
28    if (!client.connected()) {
29      reconnect();
30    }
31    delay(500);
32    client.loop();
33  }

34  void setup_wifi() {
35    delay(10);
36    WiFi.begin(WIFI_SSID, WIFI_PASS);
37    while (WiFi.status() != WL_CONNECTED) {
38      digitalWrite(ledPin, HIGH);
39      delay(200);
40      digitalWrite(ledPin, LOW);
41      delay(200);
42    }
43  }

44  void setup_certificate() {
45    if (!SPIFFS.begin()) {
46      return;
47    }
```

```
48    File ca = SPIFFS.open("/mqtt_ca.crt", "r");
49    espClient.loadCertificate(ca);
50  }

51  void reconnect() {
52    while (!client.connected()) {
53      if (client.connect("nodeMCU_LED", MQTT_USERNAME,
    MQTT_PASSWORD)) {
54        Serial.println("connected");
55        boolean sub;
56        do {
57          delay(10);
58          sub = client.subscribe("nodeMCU/LED/+/toggle");
59        } while (!sub);
60        updateStatus();
61        publishStatus("red");
62        publishStatus("green");
63        publishStatus("blue");
64      } else {
65        delay(5000);
66      }
67    }
68  }

69  void callback(char* topic, byte* payload, unsigned int
70  length) {
71    boolean state;
72    for (int i = 0; i < length; i++) {
73      char receivedChar = (char)payload[i];
74      if (receivedChar == '1') {
75        state = true;
76      }
77      if (receivedChar == '0') {
78        state = false;
79      }
80    }
81    String color = splitString((String)topic, '/', 2);
82    if (color == "red") {
83      digitalWrite(LED_RED, state);
84    } else if (color == "green") {
85      digitalWrite(LED_GREEN, state);
86    } else if (color == "blue") {
87      digitalWrite(LED_BLUE, state);
88    }
89    updateStatus();
90    publishStatus(color);
91  }

92  void publishStatus(String color) {
93    char buffer[5];
94    if (color == "red") {
95      dtostrf(redStatus, 0, 0, buffer);
96      client.publish("nodeMCU/LED/red/status", buffer);
97    } else if (color == "green") {
98      dtostrf(greenStatus, 0, 0, buffer);
99      client.publish("nodeMCU/LED/green/status", buffer);
100   } else if (color == "blue") {
101     dtostrf(blueStatus, 0, 0, buffer);
102     client.publish("nodeMCU/LED/blue/status", buffer);
```

```
103   }
104 }

105 void updateStatus() {
106   redStatus   = digitalRead(LED_RED);
107   greenStatus = digitalRead(LED_GREEN);
108   blueStatus  = digitalRead(LED_BLUE);
109 }

110 String splitString(String data, char separator, int index)
    {
111   int found = 0;
112   int strIndex[] = {0, -1};
113   int maxIndex = data.length() - 1;
114   for (int i = 0; i <= maxIndex && found <= index; i++) {
115     if (data.charAt(i) == separator || i == maxIndex) {
116       found++;
117       strIndex[0] = strIndex[1] + 1;
118       strIndex[1] = (i == maxIndex) ? i + 1 : i;
119     }
120   }
121   return found > index ? data.substring(strIndex[0],
    strIndex[1]) : "";
122 }
```

1.2. nodeMCU_DHT_TLS.ino

```
1   #include "FS.h"
2   #include "DHT.h"
3   #include <ESP8266WiFi.h>
4   #include <PubSubClient.h>

5   #define DHTPIN D5
6   #define DHTTYPE DHT11
7   #define WIFI_SSID "NAMA SSID WIFI"
8   #define WIFI_PASS "PASSWORD WIFI"
9   #define MQTT_SERVER "IP MQTT BROKER"
10  #define MQTT_PORT "PORT MQTT BROKER"
11  #define MQTT_USERNAME "TOKEN JWT"
12  #define MQTT_PASSWORD " "

13  int defPin = LED_BUILTIN;
14  unsigned long readTime;
15  DHT dht(DHTPIN, DHTTYPE);
16  WiFiClientSecure espClient;
17  PubSubClient   client(MQTT_SERVER,   MQTT_PORT,   callback,
    espClient);

18  void setup() {
19    Serial.begin(115200);
20    pinMode(defPin, OUTPUT);
21    setup_wifi();
22    delay(500);
23    setup_certificates();
24    dht.begin();
25    readTime = 0;
26  }

27  void loop() {
```

```
28    if (!client.connected()) {
29      reconnect();
30    }
31    delay(500);
32    client.loop();
33    if(millis() > readTime+60000){
34      publishStatus();
35    }
36  }

37  void setup_wifi() {
38    delay(10);
39    WiFi.begin(WIFI_SSID, WIFI_PASS);
40    while (WiFi.status() != WL_CONNECTED) {
41      digitalWrite(ledPin, HIGH);
42      delay(200);
43      digitalWrite(ledPin, LOW);
44      delay(200);
45    }
46  }

47  void setup_certificate() {
48    if (!SPIFFS.begin()) {
49      return;
50    }
51    File ca = SPIFFS.open("/mqtt_ca.crt", "r");
52    espClient.loadCertificate(ca));
53  }

54  void reconnect() {
55    while (!client.connected()) {
56      if     (client.connect("nodeMCU_DHT",     MQTT_USERNAME,
    MQTT_PASSWORD)) {
57
58      } else {
59        delay(5000);
60      }
61    }
62  }

63  void  callback(char*  topic,  byte*  payload,  unsigned  int
    length) {
64
65  }

66  void publishStatus() {
67    float t, h;
68    readTime = millis();
69    do{
70      t = dht.readTemperature();
71      h = dht.readHumidity();
72    }while (isnan(h) || isnan(t));
73    char bufferTemp[10];
74    char bufferHum[10];
75    dtostrf(t,0, 0, bufferTemp);
76    dtostrf(h,0, 0, bufferHum);
77    Client.publish("nodeMCU/DHT/temperature/status",
    bufferTemp);
78    delay(50);
```

```
  Client.publish("nodeMCU/DHT/humidity/status", bufferHum);
  delay(50);
}
```

1.3. Auth-Server.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys
import bottle
import jwt
import MySQLdb
#From https://github.com/mitsuhiko/python-pbkdf2
import pbkdf2_test as pbkdf2
from bottle import response, request

db = MySQLdb.connect(host="localhost",
                     user="user",
                     passwd="pass",
                     db="nama database")
app = application = bottle.Bottle()

def decodeJWT(encoded):
    trim = encoded[7:]
    data = jwt.decode(trim, 'Kode Rahasia', algorithms=['HS256'])
    return data

@app.route('/auth', method='POST')
def auth():
    response.content_type = 'text/plain'
    response.status = 403
    encoded = request.get_header('Authorization')
    if encoded is None:
         return
    data = decodeJWT(encoded)
    username = data['username']
    password = data['pw']
    data_sql = userQuery(username)me
    if data_sql is None:
         return
    checkhash = pbkdf2.check_hash(password, data_sql['pw'])
    if username == data['username'] and checkhash:
        print "username and password match!"
        response.status = 200
    return

def userQuery(username):
    cur = db.cursor()
    sql = "SELECT * FROM mqtt_users WHERE username = '%s' " % (username)
    try:
        cur.execute(sql)
        result = cur.fetchone()
        data = {'id': result[0], 'username': result[1], 'pw': result[2], 'role': result[3]}
        return data
    except:
```

```
45          print "Fetch User Gagal (username : \'" +username
    +"\' tidak ditemukan)"
46        return None
47      db.close()

48  @app.route('/superuser', method='POST')
49  def superuser():
50      response.content_type = 'text/plain'
51      response.status = 403
52      encoded = request.get_header('Authorization')
53      if encoded is None:
54           return
55      data = decodeJWT(encoded)
56      username = data['username']
57      su = superQuery(username)
58      if su is None:
59           return
60      if su>0:
61          response.status = 200
62      return message

63  def superQuery(username):
64      cur = db.cursor()
65      sql = "SELECT COUNT(*) FROM mqtt_users u JOIN mqtt_roles
    r ON u.role = r.title WHERE username='%s' AND r.super=1" %
    (username)
66      try:
67          cur.execute(sql)
68          result = cur.fetchone()
69          return result[0]
70      except:
71          print "Fetch Super Gagal (username : \'" +username
    +"\' tidak ditemukan"
72      db.close()

73  @app.route('/acl', method='POST')
74  def acl():
75      response.content_type = 'text/plain'
76      response.status = 403
77      encoded = request.get_header('Authorization')
78      if encoded is None:
79           return
80      data = decodeJWT(encoded)
81      username = data['username']
82      topic    = request.forms.get('topic')
83      acc      = request.forms.get('acc')
84      result = aclQuery(username, int(acc))
85      if result is None:
86           return
87      for row in result:
88          if(topic == row[0]):
89              response.status = 200
90              break
91      return

92  def aclQuery(username, acc):
93      cur = db.cursor()
94      sql = "SELECT p.topic FROM mqtt_permissions p JOIN
    mqtt_roles r ON r.title = p.role RIGHT JOIN mqtt_users u ON
```

```
    u.role = r.title WHERE username = '%s' AND rw >= '%d'" %
    (username, acc)
95      try:
96          cur.execute(sql)
97          result = cur.fetchall()
98          return result
99      except:
100         print "Fetch Gagal"
101     db.close()

102 @app.route('/getToken', method='POST')
103 def getToken():
104     response.content_type = 'text/plain'
105     response.status = 403
106     payload ={}
107     payload['username'] = request.forms.get('username')
108     payload['pw']       = request.forms.get('pw')
109     bearer  = "Bearer "
110     encoded    =    jwt.encode(payload,    'Kode    Rahasia',
    algorithm='HS256')
111     data   = bearer +encoded
112     response.status = 200
113     return data

114 @app.route('/user', method='POST')
115 def register():
116     response.content_type = 'text/plain'
117     response.status = 403
118     payload  = {}
119     username = request.forms.get('username')
120     password = request.forms.get('pw')
121     role     = request.forms.get('role')
122     username_check = checkUsername(username)
123     if(username_check>0):
124          return
125     insert_data    =    {'username':    username,    'pw':
    pbkdf2.make_hash(password), 'role': role}
126     success = insertUser(insert_data)
127     if success == 0:
128          return
129     payload = {}
130     payload['username'] = username
131     payload['pw']       = password
132     bearer  = "Bearer "
133     encoded    =    jwt.encode(payload,    'Kode    Rahasia',
    algorithm='HS256')
134     token   = bearer +encoded
135     response.status = 200
136     return

137 @app.route('/user', method='PUT')
138 def putUser():
139     response.content_type = 'text/plain'
140     response.status = 403
141     payload  = {}
142     username = request.forms.get('username')
143     password = request.forms.get('pw')
144     role     = request.forms.get('role')
145     username_check = checkUsername(username)
```

```
146     if(username_check==0):
147          return
148     update_data = {'pw': pbkdf2.make_hash(password), 'role':
    role}
149     success = updateUser(username, update_data)
150     if success == 0:
151          return
152     payload = {}
153     payload['username'] = username
154     payload['pw']       = password
155     bearer  = "Bearer "
156     encoded    =    jwt.encode(payload,    'Kode    Rahasia',
    algorithm='HS256')
157     token   = bearer +encoded
158     response.status = 200
159     return

160 @app.route('/user', method='DELETE')
161 def delUser():
162     response.content_type = 'text/plain'
163     response.status = 403
164     payload  = {}
165     username = request.forms.get('username')
166     password = request.forms.get('pw')
167     username_check = checkUsername(username)
168     if(username_check==0):
169          return
170     data_sql = userQuery(username)
171     checkhash = pbkdf2.check_hash(password, data_sql['pw'])
172     if not checkhash:
173          return
174     success = deleteUser(username)
175     if success == 0:
176          return
177     response.status = 200
178     return message

179 def checkUsername(username)
180     cur = db.cursor()
181     sql = "SELECT COUNT(*) FROM mqtt_users WHERE username =
    '%s'" % (username)
182     try:
183         cur.execute(sql)
184         result = cur.fetchone()
185         return result[0]
186     except:
187         print "Fetch Gagal"
188     db.close()

189 def insertUser(data):
190     cursor = db.cursor()
191     sql = "INSERT INTO mqtt_users(username, pw, role) VALUES
    ('%s',   '%s',   '%s')"   %   (data['username'],   data['pw'],
    data['role'])
192     try:
193           cursor.execute(sql)
194           db.commit()
195     except:
196          db.rollback()
```

```
197           return 0
198       return 1
199       db.close()

200   def updateUser(username, data):
201       cursor = db.cursor()
202       sql = "UPDATE mqtt_users SET pw= '%s', role= '%s' WHERE
      username= '%s'" % (data['pw'], data['role'], username)
203       try:
204             cursor.execute(sql)
205             db.commit()
206       except:
207            db.rollback()
208            return 0
209       return 1
210       db.close()

211   def deleteUser(username):
212       cursor = db.cursor()
213       sql = "DELETE FROM mqtt_users WHERE username = '%s'" %
      (username)
214       try:
215             cursor.execute(sql)
216             db.commit()
217       except:
218            db.rollback()
219            return 0
220       return 1
221       db.close()

222   @app.route('/role', method='POST')
223   def addRole():
224       response.content_type = 'text/plain'
225       response.status = 403
226       encoded = request.get_header('Authorization')
227       if encoded is None:
228            return
229       data = decodeJWT(encoded)
230       username = data['username']
231       password = data['pw']
232       admin = checkAdmin(username, password)
234       if not admin:
235            return
236       title = request.forms.get('title')
237       role_check = checkRole(title)
238       if(role_check>0):
239            return
240       success = insertRole(title)
241       if success == 0:
242            return
243       response.status = 200
244       return

245   @app.route('/role', method='DELETE')
246   def delRole():
247       response.content_type = 'text/plain'
248       response.status = 403
249       encoded = request.get_header('Authorization')
250       if encoded is None:
```

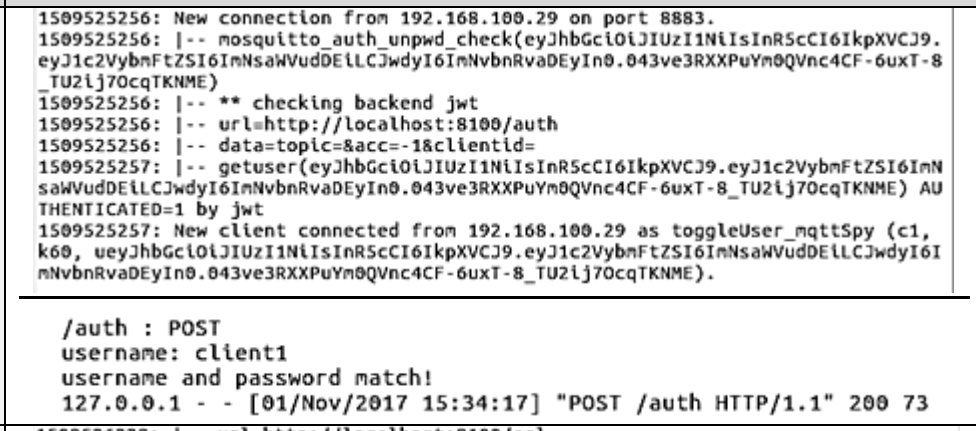
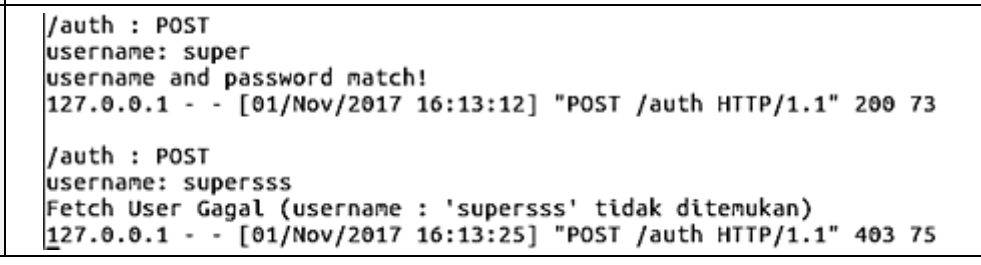
```
251           return
252     data = decodeJWT(encoded)
253     username = data['username']
254     password = data['pw']
255     admin = checkAdmin(username, password)
256     if not admin:
257           return
258     title = request.forms.get('title')
259     role_check = checkRole(title)
260     if(role_check==0):
261           return
262     success = deleteRole(title)
263     if success == 0:
264           return
265     response.status = 200
266     return

267 def checkAdmin(username, password):
268     admin = superQuery(username)
269     data  = userQuery(username)
270     check_hash = pbkdf2.check_hash(password, data['pw'])
271     if admin>0 and check_hash:
272           return True
272     else:
273           return False

274 def checkRole(title):
275     cur = db.cursor()
276     sql = "SELECT COUNT(*) FROM mqtt_roles WHERE title =
    '%s'" % (title)
277     try:
278         cur.execute(sql)
279         result = cur.fetchone()
280         return result[0]
281     except:
282         print "Fetch Gagal"
283     db.close()

284 def insertRole(title):
285     cursor = db.cursor()
286     sql = "INSERT INTO mqtt_roles(title) VALUES ('%s')" %
    (title)
287     try:
288           cursor.execute(sql)
289           db.commit()
290     except:
291           db.rollback()
292           return 0
293     return 1
294     db.close()

295 def deleteRole(title):
296     cursor = db.cursor()
297     sql = "DELETE FROM mqtt_roles WHERE title = '%s'" %
    (title)
298     try:
299           cursor.execute(sql)
300       db.commit()
301     except:
```

```
302             db.rollback()
303             return 0
304         return 1
305         db.close()

306 @app.route('/permission', method='POST')
307 def addPermission():
308         response.content_type = 'text/plain'
309         response.status = 403
310         encoded = request.get_header('Authorization')
311         if encoded is None:
312             return
313         data = decodeJWT(encoded)
314         username = data['username']
315         password = data['pw']
316         admin = checkAdmin(username, password)
317         if not admin:
318             return
319         role  = request.forms.get('role')
320         topic = request.forms.get('topic')
321         acc   = int(request.forms.get('acc'))
322         role_check = checkPermission(role, topic, acc)
323         if(role_check>0):
324             return
325         success = insertPermission(role, topic, acc)
326         if success == 0:
327             return
328         response.status = 200
329         return

330 @app.route('/permission', method='DELETE')
331 def delPermission():
332         response.content_type = 'text/plain'
333         response.status = 403
334         encoded = request.get_header('Authorization')
335         if encoded is None:
336             return
337         data = decodeJWT(encoded)
338         username = data['username']
339         password = data['pw']
340         admin = checkAdmin(username, password)
341         if not admin:
342             return
343         role  = request.forms.get('role')
344         topic = request.forms.get('topic')
345         acc   = int(request.forms.get('acc'))
346         perm_check = checkPermission(role, topic, acc)
347         if(perm_check==0):
348             return
349         success = deletePermission(role, topic, acc)
350         if success == 0:
351             return
352         response.status = 200
353         return

354 def checkPermission(role, topic, acc):
355         cur = db.cursor()
356         sql = "SELECT COUNT(*) FROM mqtt_permissions WHERE role
    = '%s' AND topic = '%s' AND rw = '%d'" % (role, topic, acc)
```

```
357     try:
358         cur.execute(sql)
359         result = cur.fetchone()
360         return result[0]
361     except:
362         print "Fetch Gagal"
363     db.close()

364 def insertPermission(role, topic, acc):
365     cursor = db.cursor()
366     sql = "INSERT INTO mqtt_permissions(role, topic, rw)
    VALUES ('%s', '%s', '%d')" % (role, topic, acc)
367     try:
368           cursor.execute(sql)
369           db.commit()
370     except:
371          db.rollback()
372          return 0
373     return 1
374     db.close()

375 def deletePermission(role, topic, acc):
376     cursor = db.cursor()
377     sql = "DELETE FROM mqtt_permissions WHERE role = '%s'
    AND topic = '%s' AND rw = '%d'" % (role, topic, acc)
378     try:
379           cursor.execute(sql)
380           db.commit()
381     except:
382          db.rollback()
383          return 0
384     return 1
385     db.close()

386 if __name__ == '__main__':
387     bottle.debug(True)
388     bottle.run(app,
389         host= "localhost",
390         port= 8100)
```

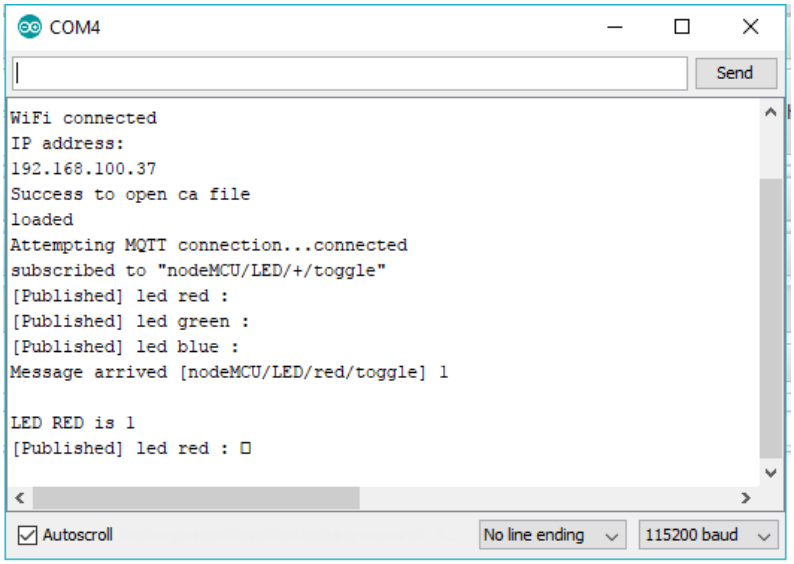
LAMPIRAN 2 : Gambar Hasil Pengujian

2.1. Screenshot Hasil Pengujian Fungsional Sistem

Kode	Screenshot Hasil Pengujian
PFS_001	
PFS_002	
PFS_003	
PFS_004	

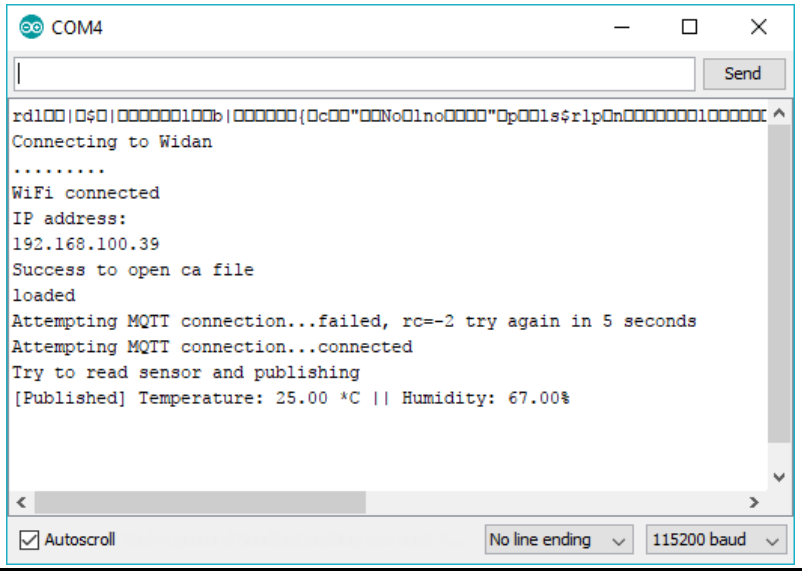
PFS_005	/acl : POST username: client2 topic: nodeMCU/LED/red/status acc: 1 information found permissions match 127.0.0.1 - - [01/Nov/2017 16:20:53] "POST /acl HTTP/1.1" 200 81 /acl : POST username: client2 topic: nodeMCU/LED/red/status acc: 2 information found no permissions match 127.0.0.1 - - [01/Nov/2017 16:21:01] "POST /acl HTTP/1.1" 403 84
PFS_006	/getToken : POST Username dan password ada di db username and password match! Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InN1cGVyIiwicHciOiIx MjM0NSJ9.FLPvEDDm242H3jiu70AyCVjE1D3WZXp1GMLIouCkG_s 127.0.0.1 - - [01/Nov/2017 16:07:46] "POST /getToken HTTP/1.1" 200 132 /getToken : POST Fetch User Gagal (username : 'supers' tidak ditemukan) Username dan password tidak ada di db 127.0.0.1 - - [01/Nov/2017 16:07:51] "POST /getToken HTTP/1.1" 403 37
PFS_007	/user : POST username valid insert data success Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InNpQ29iYSIsInB3Ijoi Y29iYWNvYmEifQ.ZJxgMZAObZ-uFNEZvYPpwMANrQCmLQbtq9mjhZ9-vq4 127.0.0.1 - - [01/Nov/2017 16:37:59] "POST /user HTTP/1.1" 200 200
PFS_008	/user : PUT username valid update data success Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6IlNpQ29iYSIsInB3Ijoi Y29iYWNvYmFhamEifQ.KMZNHV9Gjr4SJFigihZxHtTfwI49I0FVeKODpaLOWg8 127.0.0.1 - - [01/Nov/2017 16:43:33] "PUT /user HTTP/1.1" 200 204
PFS_009	/user : DELETE username valid password match delete data success 127.0.0.1 - - [01/Nov/2017 16:46:34] "DELETE /user HTTP/1.1" 200 64
PFS_010	/role : POST found header user is admin role valid insert data success 127.0.0.1 - - [01/Nov/2017 16:48:37] "POST /role HTTP/1.1" 200 84
PFS_011	/role : DELETE found header user is admin role valid delete data success 127.0.0.1 - - [01/Nov/2017 16:49:47] "DELETE /role HTTP/1.1" 200 84
PFS_012	/permission : POST found header user is admin permission valid insert data success 127.0.0.1 - - [01/Nov/2017 16:51:40] "POST /permission HTTP/1.1" 200 96
PFS_013	/permission : DELETE found header user is admin permission valid delete data success 127.0.0.1 - - [01/Nov/2017 16:52:58] "DELETE /permission HTTP/1.1" 200 96

PFS_014



Topic	Content	Browse	Messages	Last received
nodeMCU/LED/red/status	1	☑	3	2017/11/01 17:22:30:079

PFS_015



Topic ▲	Content	Browse	Messages	Last received
nodeMCU/DHT/humidity/status	68	☑	1	2017/11/01 17:26:50:591
nodeMCU/DHT/temperature/st...	25	☑	1	2017/11/01 17:26:50:492

2.2. Screenshot Auth-Server Pengujian Keamanan

Kode	Screenshot Hasil Pengujian
PMO_201	^Cardhiaan@ardhiaan-VirtualBox:~$ python mqtt-auth-server/http-auth.py Bottle v0.12.13 server starting up (using WSGIRefServer())... Listening on http://localhost:8100/ Hit Ctrl-C to quit. /auth : POST 127.0.0.1 - - [02/Nov/2017 17:15:11] "POST /auth HTTP/1.1" 403 58
PMO_202	ardhiaan@ardhiaan-VirtualBox:~$ python mqtt-auth-server/http-auth.py Bottle v0.12.13 server starting up (using WSGIRefServer())... Listening on http://localhost:8100/ Hit Ctrl-C to quit. /auth : POST username: asal Fetch User Gagal (username : 'asal' tidak ditemukan) 127.0.0.1 - - [02/Nov/2017 17:49:07] "POST /auth HTTP/1.1" 403 75
PMO_203	^Cardhiaan@ardhiaan-VirtualBox:~$ python mqtt-auth-server/http-auth.py Bottle v0.12.13 server starting up (using WSGIRefServer())... Listening on http://localhost:8100/ Hit Ctrl-C to quit. /auth : POST username: client1 username and password match! 127.0.0.1 - - [02/Nov/2017 17:21:49] "POST /auth HTTP/1.1" 200 73
PMO_301	^Cardhiaan@ardhiaan-VirtualBox:~$ python mqtt-auth-server/http-auth.py Bottle v0.12.13 server starting up (using WSGIRefServer())... Listening on http://localhost:8100/ Hit Ctrl-C to quit. /auth : POST 127.0.0.1 - - [02/Nov/2017 18:08:27] "POST /auth HTTP/1.1" 403 58
PMO_302	^Cardhiaan@ardhiaan-VirtualBox:~$ python mqtt-auth-server/http-auth.py Bottle v0.12.13 server starting up (using WSGIRefServer())... Listening on http://localhost:8100/ Hit Ctrl-C to quit. /auth : POST username: asal Fetch User Gagal (username : 'asal' tidak ditemukan) 127.0.0.1 - - [02/Nov/2017 17:57:25] "POST /auth HTTP/1.1" 403 75
PMO_303	^Cardhiaan@ardhiaan-VirtualBox:~$ python mqtt-auth-server/http-auth.py Bottle v0.12.13 server starting up (using WSGIRefServer())... Listening on http://localhost:8100/ Hit Ctrl-C to quit. /auth : POST username: client1 username and password match! 127.0.0.1 - - [02/Nov/2017 18:12:55] "POST /auth HTTP/1.1" 200 73

PMOP_201	/acl : POST username: siMerah topic: topik/pengujian/apaSaja acc: 2 information found no permissions match 127.0.0.1 - - [04/Nov/2017 15:01:35] "POST /acl HTTP/1.1 " 403 84
PMOP_202	/acl : POST username: siMerah topic: nodeMCU/DHT/temperature/status acc: 2 information found no permissions match 127.0.0.1 - - [04/Nov/2017 15:08:07] "POST /acl HTTP/1.1 " 403 84
PMOP_203	/acl : POST username: siMerah topic: nodeMCU/LED/red/toggle acc: 2 information found permissions match 127.0.0.1 - - [04/Nov/2017 15:16:37] "POST /acl HTTP/1.1 " 200 81
PMOP_301	/acl : POST username: siMerah topic: topik/pengujian/apaSaja acc: 2 information found no permissions match 127.0.0.1 - - [04/Nov/2017 15:23:23] "POST /acl HTTP/1.1 " 403 84
PMOP_302	/acl : POST username: siMerah topic: nodeMCU/DHT/temperature/status acc: 2 information found no permissions match 127.0.0.1 - - [04/Nov/2017 15:28:15] "POST /acl HTTP/1.1 " 403 84
PMOP_303	/acl : POST username: siMerah topic: nodeMCU/LED/red/toggle acc: 2 information found permissions match 127.0.0.1 - - [04/Nov/2017 15:30:44] "POST /acl HTTP/1.1 " 200 81
PMOS_201	/acl : POST username: siSuhu topic: topik/pengujian/apaSaja acc: 1 information found no permissions match 127.0.0.1 - - [04/Nov/2017 16:43:15] "POST /acl HTTP/1.1 " 403 84

PMOS_202	/acl : POST username: siSuhu topic: nodeMCU/LED/red/status acc: 1 information found no permissions match 127.0.0.1 - - [04/Nov/2017 16:54:21] "POST /acl HTTP/1.1 " 403 84
PMOS_203	/acl : POST username: siSuhu topic: nodeMCU/DHT/temperature/status acc: 1 information found permissions match 127.0.0.1 - - [04/Nov/2017 16:57:30] "POST /acl HTTP/1.1 " 200 81
PMOS_301	/acl : POST username: siSuhu topic: topik/pengujian/apaSaja acc: 1 information found no permissions match 127.0.0.1 - - [04/Nov/2017 17:06:08] "POST /acl HTTP/1.1 " 403 84
PMOS_302	/acl : POST username: siSuhu topic: nodeMCU/LED/red/status acc: 1 information found no permissions match 127.0.0.1 - - [04/Nov/2017 17:11:44] "POST /acl HTTP/1.1 " 403 84
PMOS_303	/acl : POST username: siSuhu topic: nodeMCU/DHT/temperature/status acc: 1 information found permissions match 127.0.0.1 - - [04/Nov/2017 17:15:04] "POST /acl HTTP/1.1 " 200 81

2.3. Screenshot Pengujian Performa Sistem

Kode	Screenshot Hasil Pengujian
PPS_001	C:\Users\Ardhian>mqtt-benchmark --broker=mqtt://192.168.100.40:1883 --clients 10 --count 100 --topic nodeMCU/LED/red/toggle Trying to conenct to 10 clients ... Conencted to 10 sucessfully Average result for each client of the 10 clients. Establishing connection (sec): 0.0288 Success in publishing (msg): 100/100 Failure in publishing (msg): 0/100 Duraion in publishing (sec): 0.1379 Throughput (msg/sec): 925.3822

PPS_002	C:\Users\Ardhian>mqtt-benchmark --broker=mqtt://192.168.100.40:1883 --clients 10 --count 100 --topic nodeMCU/LED/red/toggle --username eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InNpTWVyYWgiLCJwdyI6InJlZDEyMzQ1In0.vvcTUByumkomj-oxQpqrGVkjK89zncElr4kFKxgkIQI --password asal Trying to conenct to 10 clients ... Conencted to 10 sucessfully Average result for each client of the 10 clients. Establishing connection (sec): 0.0662 Success in publishing (msg): 100/100 Failure in publishing (msg): 0/100 Duraion in publishing (sec): 0.4883 Throughput (msg/sec): 204.8549
PPS_003	C:\Users\Ardhian>mqtt-benchmark --broker=mqtt://192.168.100.40:1883 --clients 10 --count 100 --topic nodeMCU/LED/red/toggle --username eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InNpTWVyYWgiLCJwdyI6InJlZDEyMzQ1In0.vvcTUByumkomj-oxQpqrGVkjK89zncElr4kFKxgkIQI --password asal --ca C:\Users\Ardhian\Dropbox\SKRIPSI\apps\mosquitto\mqtt-ssl-ca\mqtt_ca.crt Trying to conenct to 10 clients ... Conencted to 10 sucessfully Average result for each client of the 10 clients. Establishing connection (sec): 0.0853 Success in publishing (msg): 100/100 Failure in publishing (msg): 0/100 Duraion in publishing (sec): 0.7092 Throughput (msg/sec): 141.0726