LAMPIRAN

Lampiran 1: Program connectedWifi.py

```
1   import socket
2   import sys
3   import websocket
4   import json
5   import time
6   import pyping
7
8   host, port = '', 8000
9
10  sys.path.append("/home/pi/skripsi/server")
11  from database.database_handler import *
12  from ws.websocket_handler import *
13
14
15  sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
16  start_time = time.time()
17  print("discovery start_time: %s" % start_time)
18  sock.bind((host, port))
19
20  data, addr = sock.recvfrom(1024)
21  print addr[0]
22  print addr[1]
23  print("")
24  dataRecv = json.loads(data)
25  print dataRecv
26  print ("")
27  deviceCon = "WIFI"
28  insert_to_database(dataRecv, addr[0] + ":" + str(addr[1]),
29  deviceCon)
30  deviceData = read_device_data()
31  websocket_send_once('dev' + deviceData)
32  end_time = time.time()
33  print("discovery end_time: %s" % end_time)
34  time_taken = end_time - start_time
35  print("Discovery time: %s" % time_taken)
36  print("")
37
38  def update(data):
39          update_service_data(data)
40          deviceData = read_device_data()
41          websocket_send(ws, 'dev' + deviceData)
42
43  def wifi_service_handler(data):
44          time.sleep(0.5)
45          if data[1] == "00013":
46                  sock.sendto(data[1] + "_" + data[2] + "_" +
47  data[3] + "_" + data[4], addr)
48                  print data[1] + "_" + data[2] + "_" + data[3] +
49  "_" + data[4]
50                  update(data)
51
52          else:
53                  sock.sendto(data[1] + "_" + data[2], addr)
54                  print data[1] + "_" + data[2]
55                  update(data)
56
57  def userinterface_handler():
58          while 1:
59                  try:
60                          result = []
61                          try:
```

```
62                                      message = websocket_receive(ws)
63                                      print message
64                              except ValueError:
65                                      pass
66
67                              if message[0:3] == 'ser':
68                                      print message[0:3]
69                                      data = message[3:].split('_')
70                                      print data
71                                      if data[0] == "WIFI":
72                                              data =
73  message[8:].split('_')
74                                              print data
75                                              start_response_time =
76  time.time()
77
78          print("start_response_time: %s" % start_response_time)
79
80          wifi_service_handler(data)
81                                              end_response_time =
82  time.time()
83
84          print("end_response_time: %s" % end_response_time)
85                                              response_time =
86  end_response_time - start_response_time
87                                              print("Response time:
88  %s" % response_time)
89
90                              time.sleep(0.1)
91                      except KeyboardInterrupt:
92                              break
93
94  try:
95          ws = connect_websocket()
96  except ValueError:
97          print "cannot find websocket server!!!"
98
99  while 1:
100         r = pyping.ping(addr[0])
101         if r.ret_code == 0:
102                 userinterface_handler()
103
104         elif r.ret_code == 1:
105                 print "peer terputus"
106                 delete_from_database(addr[0] + ":" +
107 str(addr[1]))
108                 deviceData = read_device_data()
109                 websocket_send_once('dev' + deviceData)
110                 break
111
112 disconnect_websocket(ws)
113
114 sock.close()
```

Lampiran 2: disconnectedWifi.py

```
1  import socket
2  import sys
3  import websocket
4  import json
5
6  sys.path.append("/home/pi/skripsi/server")
7  from database.database_handler import delete_from_database,
8  read_device_data
9  from ws.websocket_handler import websocket_send_once
10
11 #def send_portData(message):
12 delete_from_database("192.168.1.36:2390")
13 deviceData = read_device_data()
14 websocket_send_once('dev' + deviceData)
```

Lampiran 3: Program websocket_handler.py

```
1  #handler for websocket communication to websocket server
2
3  #dependency library
4  import websocket
5
6  #defined websocket server that the application connected to
7  websocketServer = "ws://127.0.0.1:8888/ws"
8
9  #connect to websocket server
10 def connect_websocket():
11     ws = websocket.create_connection(websocketServer)
12     return ws
13
14 #disconnect from websocket server
15 def disconnect_websocket(ws):
16     #print("ws disconnected")
17     ws.close()
18
19 #connect to websocket server, send data once, and then get
20 disconnected
21 def websocket_send_once(message):
22     ws = websocket.create_connection(websocketServer)
23     ws.send(message)
24     ws.close()
25
26 #websocket function for listening message from websocket server
27 def websocket_receive(ws):
28          result = ws.recv()
29          return result
30
31 #send message to connected websocket server
32 def websocket_send(ws, message):
33          ws.send(message)
```

Lampiran 4: Program local_websocket_server.py

```
1  #websocket server for local application usage, including local
2  webserver
3  #using local database
4
5  #dependencies library
6  import json
7  import serial
8  import sqlite3
9  import sys
10 import time
11 import tornado.httpserver
12 import tornado.websocket
13 import tornado.ioloop
14 import tornado.web
15 from datetime import datetime
16 #application package
17 sys.path.append("/home/pi/skripsi/server") #root directory for
18 application
19 from database.database_handler import read_device_data
20
21 #declaring variables
22 clients = [] #variable reserved for connected client to websocket
23 port = 8888 #websocket used port
24
25 #read device data from database
26 def read():
27     deviceData = read_device_data()
28     return deviceData
29
30 #received device information handler
31 def device_handler(data):
32     data_device = "upd"
33
34     #read data device from received message
35     for line in data:
36         data_device += line
37
38     #send data device to all client, actually intended to web
39 browser
40     for c in clients:
41         c.write_message(data_device)
42
43 #websocket handler
44 class WSHandler(tornado.websocket.WebSocketHandler):
45     def check_origin(self, origin):
46         return True
47
48     #when connection has been made from clients
49     def open(self):
50         clients.append(self) #add new client to clients variable
51         deviceJson = read() #read device data from database
52         self.write_message("upd" + deviceJson) #send data device to
53 recently connected client
54
55     #when message received
56     def on_message(self, message):
57         print (message)
58
59         #when data device information is received
60         if message[0:3] == 'dev':
61             device_handler(message[3:])
62
63         #when message service request is received
64         elif message[0:3] == 'ser':
```

```
65              for c in clients:
66                  c.write_message(message)
67
68          #when message service respons has been received
69          elif message[0:3] == "res" :
70              message = message[3:]
71              print(message)
72
73      #when connection is closed by a client
74      def on_close(self):
75           #message = "putus"
76          clients.remove(self)
77
78  application = tornado.web.Application([
79    (r'/ws', WSHandler),
80  ])
81
82  if __name__ == "__main__":
83      http_server = tornado.httpserver.HTTPServer(application)
84      http_server.listen(port)
85      tornado.ioloop.IOLoop.instance().start()
```

Lampiran 5: Program database_handler.py

```
1   #database handler for local aaplication
2
3   #dependencies library
4   import sqlite3
5
6   #define used database for local application
7   database = "/home/pi/skripsi/server/database/serverDatabase.db"
8
9   #connect to used database
10  def database_connect():
11      try:
12          conn = sqlite3.connect(database)
13      except:
14          pass
15      return conn, conn.cursor()
16
17  #read devices data from database
18  def read_device_data():
19      conn, c = database_connect()
20      deviceData = ''
21      try:
22           c.execute('SELECT * FROM device')
23           device = c.fetchall()
24           for rowD in device:
25               deviceData = deviceData + '{\"device_id\":\"' +
26  rowD[0] + '\",\"device_name\":\"' + rowD[1] + '\",\"port\":\"' +
27  rowD[2] + '\",\"device_con\":\"' + rowD[3] +  '\",\"service\":['
28               c.execute('SELECT s.*, d.device_id FROM device d
29  INNER JOIN device_service ds ON ds.device_id = d.device_id INNER
30  JOIN service s ON s.service_id = ds.service_id WHERE
31  d.device_id=\"' + rowD[0] + '\"')
32               service = c.fetchall()
33               for rowS in service:
34                   deviceData = deviceData + '{\"service_id\":\"' +
35  str(rowS[0]) + '\",\"service_name\":\"' + rowS[1] +
36  '\",\"service_type\":\"' + rowS[2] + '\",\"service_data\":\"' +
37  rowS[3] + '\",\"service_value\":\"' + str(rowS[4]) + '\"},'
38               deviceData = deviceData[:-1] + ']}\n'
39      except:
40           pass
41      conn.close()
42      return deviceData
43
44  #insert new device data to database
45  def insert_to_database(data, port, con):
46      conn, c = database_connect()
47      c.execute("INSERT INTO device VALUES (\'" + data["device_id"]
48  + "\',\'" + data["device_name"] + "\',\'" + port + "\',\'" + con +
49  "\')")
50      for service in data["service"]:
51          c.execute("INSERT INTO service VALUES (\'" +
52  service["service_id"] + "\',\'" + service["service_name"] +
53  "\',\'" + service["service_type"] + "\',\'" +
54  service["service_data"] + "\',\'" + service["service_value"] +
55  "\')")
56          c.execute("INSERT INTO device_service VALUES (\'" +
57  data["device_id"] + "\',\'" + service["service_id"] + "\')")
58
59      conn.commit()
60      conn.close()
61
62  #delete target device data from database
63  def delete_from_database(port):
64      conn, c = database_connect()
```

```
65      query = ""
66
67      #get device_id from the taget device
68      query = "SELECT device_id FROM device WHERE port=\'" + port +
69  "\'"
70      c.execute(query)
71      device_id = c.fetchone()
72
73      #get all service_id from target device
74      query = "SELECT service_id FROM device_service WHERE
75  device_id=\'" + str(device_id[0]) + "\'"
76      c.execute(query)
77      service_id = c.fetchall()
78
79      #delete target device data from DEVICE_SERVICE table
80      query = "DELETE FROM device_service WHERE device_id=\'" +
81  str(device_id[0]) + "\'"
82      c.execute(query)
83
84      #delete target device data from DEVICE table
85      query = "DELETE FROM device WHERE device_id=\'" +
86  str(device_id[0]) + "\'"
87      c.execute(query)
88
89      #delete target device's all service from SERVICE table
90      for sid in service_id:
91          query = "DELETE FROM service WHERE service_id=\'" +
92  str(sid[0]) + "\'"
93          c.execute(query)
94
95      conn.commit()
96      conn.close()
97
98  #update service data after service was done
99  def update_service_data(data):
100     conn, c = database_connect()
101
102     if data[1] == "00013":
103         data1 = "00013"
104         s = '_'
105         data = (data[2], data[3], data[4])
106         data_join = s.join( data )
107         #data2 = "dunia"
108
109         query = "UPDATE service SET service_value = \'" +
110 data_join + "\' WHERE service_id = \'" + data1 + "\'"
111         c.execute(query)
112
113     else:
114         query = "UPDATE service SET service_value = " + data[2] +
115 " WHERE service_id = \'" + data[1] + "\'"
116         c.execute(query)
117
118     conn.commit()
119     conn.close()
120
121 #delete all database data
122 def clear_database():
123     conn, c = database_connect()
124
125     c.execute("DELETE FROM device")
126     c.execute("DELETE FROM service")
127     c.execute("DELETE FROM device_service")
128     conn.commit()
129   conn.close()
```

Lampiran 6: Basis Data serverDatabase.db

```
1  sqlite3 serverDatabase.db
2  create table device(
3      device_id varchar(32) primary key,
4      device_name varchar(32),
5      port varchar(32),
6      device_con varchar(10)
7  );
8  create table service(
9      service_id varchar(32) primary key,
10     service_name varchar(32),
11     service_type varchar(32),
12     service_data varchar(32),
13     service_value varchar(32)
14 );
15 create table device_service(
16     device_id varchar(32) primary key,
17     service_id varchar(32)
18 );
```

Lampiran 7: Program clear_database.py

```
1  #clear all the data on used database
2  #used for debugging only
3
4  #dependencies library
5  import sys
6  #application package
7  sys.path.append("/home/pi/skripsi/server")  #root directory for
8  application
9  from database.database_handler import clear_database
10 from ws.websocket_handler import websocket_send_once
11
12 try:
13     clear_database() #call clear database function
14     websocket_send_once('dev') #send empty device data to websocket
15 server
16 except:
17     pass
```

Lampiran 8: Wifi_skripsi_fix.ino

```
1   #include <ArduinoJson.h>
2   #include <FiniteStateMachine.h>
3   #include <LED.h>
4   #include <ESP8266WiFi.h>
5   #include <WiFiUDP.h>
6
7   int redPin= D5;
8   int greenPin = D1;
9   int bluePin = D3;
10
11  void onOff(String value){
12    if(value == "1"){
13      digitalWrite(D6, HIGH);
14    }
15    if(value == "0"){
16      digitalWrite(D6, LOW);
17    }
18  }
19
20  void dimmer(String value){
21    if(value <= "255"){
22      analogWrite(D7, value.toInt());
23    }
24    else{
25      analogWrite(D7, LOW);
26    }
27  }
28
29  void rgb(String value){
30    if(value){
31      int indexRGB0 = value.indexOf("_");
32      int indexRGB1 = value.indexOf("_", indexRGB0+1);
33      String reqID0 = value.substring(0,indexRGB0);
34      String reqID1 = value.substring(indexRGB0+1, indexRGB1);
35      String reqID2 = value.substring((indexRGB1+1));
36      setColor(reqID0.toInt(), reqID1.toInt(), reqID2.toInt());
37    }
38  }
39
40  void setColor(int redValue, int greenValue, int blueValue) {
41    analogWrite(redPin, redValue);
42    analogWrite(greenPin, greenValue);
43    analogWrite(bluePin, blueValue);
44  }
45
46  String sendDevice;
47
48  String device_id = "0001";
49  String device_name = "Wemos D1 R2";
50
51  String serviceList[][5] = {
52                             {"00011", "on_off", "input", "bool",
53  "0"},
54                               {"00012", "dimming", "input", "int",
55  "0"},
56                                 {"00013", "rgb", "input", "int",
57  "0_0_0"}
58  };
59
60  void (*serviceHandler[])(String value) = {onOff, dimmer, rgb};
61  //void (*serviceHandlerRGB[])(String value) = {rgb};
62
63  StaticJsonBuffer<500> jsonBuffer;
64  JsonObject& deviceData = jsonBuffer.createObject();
```

```
65  JsonObject& serviceData = jsonBuffer.createObject();
66  int totalService;
67
68  void idleClientState(), serverWaitReq(), waitRes();
69  State idle = State(idleClientState);
70  State waitRequest = State(serverWaitReq);
71  State waitResponse = State(waitRes);
72
73  FSM arduinoStateMachine = FSM(idle);
74
75  WiFiUDP Udp;
76
77  char packetBuffer[255];
78
79  void setup() {
80    // put your setup code here, to run once:
81    Serial.begin(115200);
82    WiFi.begin("LG Magna");
83    pinMode(D6, OUTPUT);
84    pinMode(D1, OUTPUT);
85    pinMode(D3, OUTPUT);
86    pinMode(D5, OUTPUT);
87    digitalWrite(D6, LOW);
88    digitalWrite(D1, LOW);
89    digitalWrite(D3, LOW);
90    digitalWrite(D5, LOW);
91    while (WiFi.status() != WL_CONNECTED){
92      Serial.print(".");
93      delay(200);
94    }
95    Serial.println();
96    Serial.print("Terhubung ke SSID Jaringan");
97    Serial.println();
98    Serial.print("Dengan Alamat IP ");
99    Serial.print(WiFi.localIP());
100   totalService = ((sizeof(serviceList)/sizeof(String))/5);
101   generateJSON();
102   Udp.begin(2390);
103 }
104
105 void loop() {
106   // put your main code here, to run repeatedly:
107   arduinoStateMachine.update();
108   delay(500);
109 }
110
111 void idleClientState(){
112   deviceData.printTo(sendDevice);
113   Udp.beginPacket("255.255.255.255", 8000);
114   Udp.print(sendDevice);
115   Udp.println("");
116   Udp.endPacket();
117   sendDevice.remove(0);
118   arduinoStateMachine.transitionTo(waitRequest);
119   arduinoStateMachine.update();
120 }
121
122 void serverWaitReq(){
123   int packetSize = Udp.parsePacket();
124   if(packetSize){
125     Serial.println();
126     Serial.print("Menerima Paket Dengan Ukuran: ");
127     Serial.println(packetSize);
128
129     int len = Udp.read(packetBuffer, packetSize);
```

```
130     if(len > 0){
131       packetBuffer[len] = 0;
132     }
133     Serial.println();
134     Serial.println("Konten ");
135     Serial.print(String(packetBuffer));
136     int index = String(packetBuffer).indexOf('_');
137     String serviceId = String(packetBuffer).substring(0,index);
138     String valueData = String(packetBuffer).substring((index+1));
139     for(int i = 0; i < totalService; i++){
140       if(serviceList[i][0] == serviceId){
141         serviceHandler[i](valueData);
142         serviceList[i][4] = valueData;
143         generateJSON();
144         delay(500);
145         deviceData.printTo(sendDevice);
146         Udp.beginPacket("255.255.255.255", 8000);
147         Udp.print(sendDevice);
148         Udp.println("");
149         Udp.endPacket();
150         sendDevice.remove(0);
151         arduinoStateMachine.transitionTo(waitResponse);
152         arduinoStateMachine.update();
153       }
154     }
155   }
156 }
157
158 void waitRes(){
159   arduinoStateMachine.transitionTo(waitRequest);
160   arduinoStateMachine.update();
161 }
162
163 void generateJSON(){
164   deviceData["device_id"] = device_id;
165   deviceData["device_name"] = device_name;
166   JsonArray& service = deviceData.createNestedArray("service");
167   for(int i = 0; i < totalService; i++){
168     JsonObject& serviceData = jsonBuffer.createObject();
169     serviceData["service_id"] = serviceList[i][0];
170     serviceData["service_name"] = serviceList[i][1];
171     serviceData["service_type"] = serviceList[i][2];
172     serviceData["service_data"] = serviceList[i][3];
173     serviceData["service_value"] = serviceList[i][4];
174     service.add(serviceData);
175   }
176 }
```

Lampiran 9: tornado.php

```
<!doctype html>
<html>
<head>
                <title>Device Management</title>
                <meta charset="utf-8" />
                <style type="text/css">
                         body{
                        text-align: center;
                        min-width: 500px;
                        }
                </style>
<script src="jquery-1.11.1.min.js"></script>
<script>
                var tes;
                $(function(){
                         var ws;
                         tes = function tes(a,b,c,d) {
                                  var reqData =
document.getElementById("status"+d).value
                                  ws.send("ser" + a + "_" + b +
"_" +  c + "_" + reqData);
                                 //  alert("ser" + a + "_" + b +
"_" +  c + "_" + reqData);
                                  //  d = new Date();
                                  //  alert(d.getTime());
                         }
                         var logger = function(msg) {
                                  if (msg.slice(0,3) == "upd"){
                                           d = new Date();
                                          //  alert(msg);
                                           msg = msg.slice(3);
                                           var str =
msg.split("\n");
                                           var i, cnt =
"<table><tbody>";
                                           cnt = cnt + "<tr><th
class='hed' style='width: 25%;'>Port</th>" +
                                           "<th class='hed'
style='width: 25%;'>Device</th>" +
                                           "<th class='hed'
style='width: 50%;'>Service</th></tr>";
                                           for (i = 0; i <
(str.length-1); i++){
                                                    var x =
JSON.parse(str[i]);
                                                    cnt = cnt +
"<tr><td class='cont'>" + x.port +
                                                    "</td><td
class='cont'>" + x.device_name + "</td><td class='cont'>";
                                                    for (j = 0; j <
(x.service).length; j++){
                                                             cnt =
cnt + x.service[j].service_name + ": " + x.service[j].service_value
+
"</br><input type='text' id='status" + i + j + "' value='" +
x.service[j].service_value +
                                                                      "'>
<button type='button' onClick='tes(\"" + x.device_con + "\",\"" +
x.port + "\",\"" + x.service[j].service_id + "\",\"" + i + "" + j +
"\")' >Execute</button></br>";
                                                    }
                                                    cnt = cnt +
"</td></tr>";
```

```
                                                }
                                                cnt = cnt +
"</tbody></table>";
                                                $("#dev").html(cnt);
                                                 // d = new Date();
                                                 // alert(d.getTime());
                                        }
                                       //  d = new Date();
                                       //  alert(d.getTime());
                                }

                                ws = new
WebSocket("ws://127.0.0.1:8888/ws");
                                ws.onmessage = function(evt){
                                        logger(evt.data);
                                };
                        });

</script>
<style>
                table, th, td {
                        border-collapse:collapse;
                        text-align: center;
                }
                table {
                        border-collapse: collapse;
                        width: 80%;
                        margin-left: 10%;
                }
                body {
                        font-family:"arial", "MS PGothic", sans-
serif;
                        background-color:#f4efdc;
                        margin-left:auto;
                        margin-right:auto;
                        margin-top:0px;
                }
                td {
                        padding: 5px;
                }
                .td{
                        border:solid 1px#000000;
                }
                .tr{
                        border:hidden;
                }
                .cont {
                        border-collapse: collapse;
                        text-align: center;
                        border: 1px solid #000066;
                        background-color:#f4edfc;
                }
                .hed{
                        color: white;
                        border-collapse: collapse;
                        text-align: center;
                        border: 1.5px solid #000066;
                        background-color:#0000FF;
                }
</style>
</head>

<body>
                <h1>Device Management</h1>
                <div id="dev"></div>
```

```
</body>
</html>
```