Raspberry Pi Broker Web Server - ESP8266 MQTT

Auto-Updates Every 10 Seconds

Page Updated(13h:47m:54s)

GPIO 0 is currently **off**

Turn on

GPIO 16 is currently **off**

Turn on

GPIO 4 is currently **off**

Turn on

GPIO 5 is currently **off**

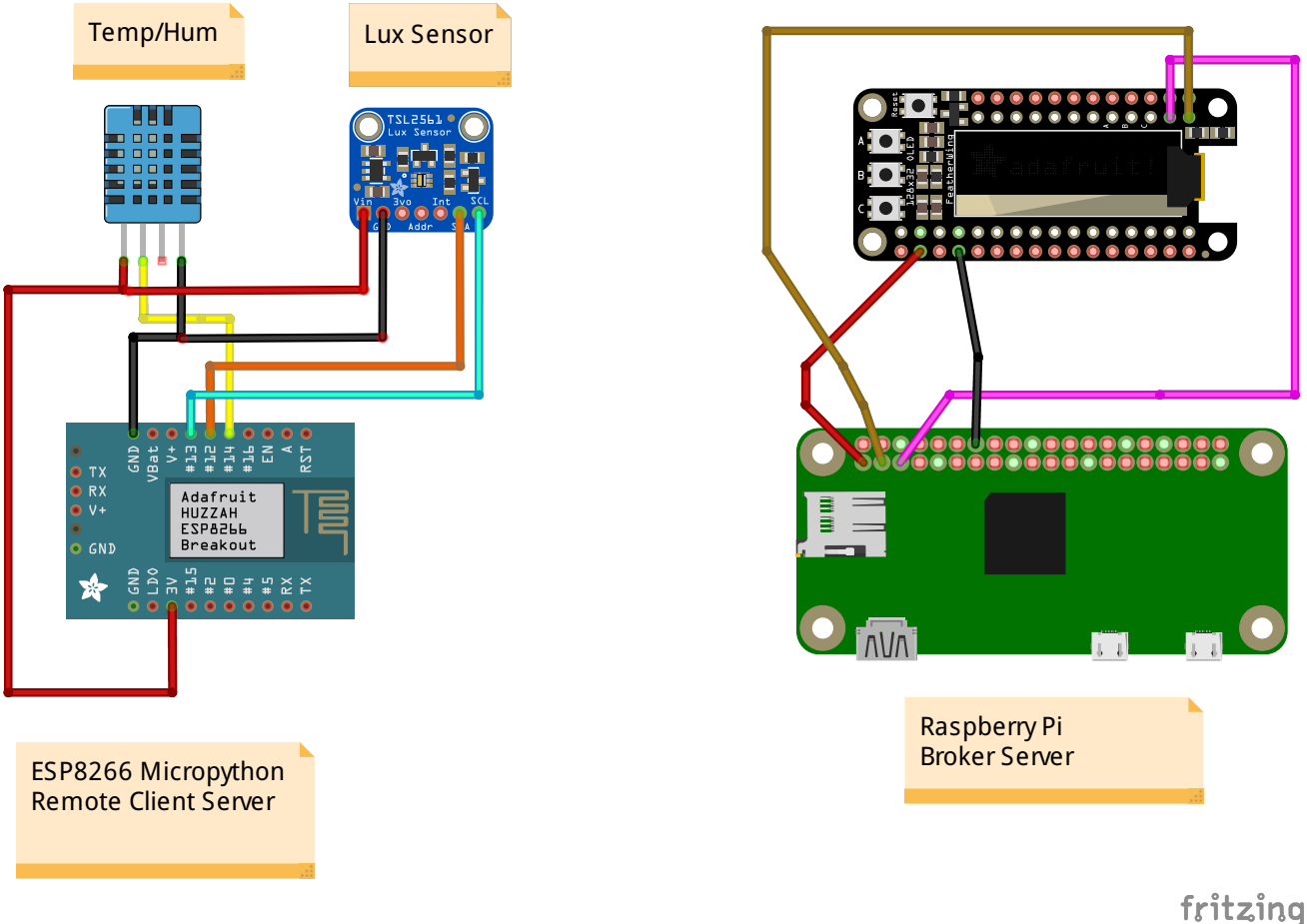
Turn on

Sensor Readings

Temperature: 69.8°F

Humidity: 36%RH

Light Intensity: 122.342Lux


Temp/Hum
Lux Sensor
OLED Display
TSL2561
Lux Sensor
Vin
3vo
Int
SCL
GND
Addr
SDA
Adafruit
HUZZAH
ESP8266
Breakout
TX
RX
V+
GND
VBat
#13
#12
#14
#16
EN
RST
LDO
3V
#15
#2
#0
#4
#5
Reset
OLED
128x32
Featherwing
ESP8266 Micropython
Remote Client Server
Raspberry Pi
Broker Server
fritzing

```
"""
Main Program Script
===================

Add the following libraries using PIP or PIP3 commands:
 - sudo apt-get install mosquitto mosquitto-clients
 - sudo pip install adafruit-ssd1306 (if using)
 - sudo pip install flask
 - sudo pip install eventlet
 - sudo pip install paho-mqtt
 - sudo pip install flask-socketio
Next Create a file called "web-server" by typing in the terminal window the following:
 - mkdir web-server
 - Create the main Broker program using "Idle" called "app.py" and store it in that
 folder.
Next Create a file called "templates" and place this folder in the "web-server"
directory by typing in the terminal window the following:
 - mkdir templates
 - Create the html code using "Idle" and call it "main.html" and store it in the
 template folder.

a) We can test each "publish topic" by subscribing with the following
command in a window mosquitto_sub -d -t "/sensor1/temp"(change path
for other topics).
b) We can test each "subscribe topic" by publishing with the following
command in a window mosquitto_pub -r -t "/esp8266/4" -m "1" or "0"
(change path for other topics)...

Start by the following (turn on ESP8266, then run Pi Brokker, then pull-up WebPage.
You can monitor WebPage activities by hitting "F12" on a laptop using Google Browser.

Note(s)-
1) If you get an occasional error, re-start the Pi Brooker and WebPage

2) If you start getting many socketio HTTP 404 errors, Uncomment evenlet
   and/or try the gevent and monkey patch
3) Or may want to turn off ASYNC Mode

"""

# Import Modules
# --------------
#from gevent import monkey
#monkey.patch_all()
#import eventlet
import paho.mqtt.client as mqtt
from threading import Thread
from flask import Flask, render_template, request, session
from flask_socketio import SocketIO, emit
from time import sleep

app = Flask(__name__,template_folder='/home/pi/templates')#,template_folder location
app.config['SECRET_KEY'] = 'secret123!' # Unique ID
socketio = SocketIO(app)

# Global String Variables
# -----------------------
x = "" # Temperature
y = "" # Humidity
z = "" # Lux


# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))

    # Subscribing in on_connect() means that if we lose the connection and
```

```
65      # reconnect then subscriptions will be renewed.
66      client.subscribe("/sensor1/temp")
67      client.subscribe("/sensor1/hum")
68      client.subscribe("/sensor3/lux")
69      client.publish('/esp8266/0', '0', qos=0, retain=True)
70      client.publish('/esp8266/4', '0', qos=0, retain=True)
71      client.publish('/esp8266/5', '0', qos=0, retain=True)
72      client.publish('/esp8266/16', '0', qos=0, retain=True)
73
74  # The callback for when a PUBLISH message is received from the ESP8266.
75  def on_message(client, userdata, message):
76      global x, y, z
77      #socketio.emit('my variable')
78      print("Received message '" + str(message.payload) + "' on topic '" + message.topic
        + "' with QoS " + str(message.qos))
79      if message.topic == "/sensor1/temp":
80          print("temperature update")
81          x = message.payload.decode('UTF-8') # Convert bytes to String or use ASCII
82          #socketio.emit('dht_temperature', {'data': message.payload},
            namespace='/broker')
83      if message.topic == "/sensor1/hum":
84          print("humidity update")
85          y = message.payload.decode('UTF-8') # Convert bytes to String
86          #socketio.emit('dht_humidity', {'data': message.payload}, namespace='/broker')
87      if message.topic == "/sensor3/lux":
88          print("light intensity update")
89          z = message.payload.decode('UTF-8') # Convert bytes to String
90          #socketio.emit('light_lux', {'data': z}, namespace='/broker')
91          #sleep(1)
92
93  mqttc=mqtt.Client()
94  mqttc.on_connect = on_connect
95  mqttc.on_message = on_message
96  mqttc.connect("localhost",1883,60) # MQTT Server Information
97  mqttc.loop_start()
98
99  # Create a dictionary called pins to store the pin number, name, and pin state:
100 pins = {
101    0 : {'name' : 'GPIO 0', 'board' : 'esp8266', 'topic' : '/esp8266/0', 'state' :
       'False'},
102    4 : {'name' : 'GPIO 4', 'board' : 'esp8266', 'topic' : '/esp8266/4', 'state' :
       'False'},
103    5 : {'name' : 'GPIO 5', 'board' : 'esp8266', 'topic' : '/esp8266/5', 'state' :
       'False'},
104    #12 : {'name' : 'GPIO 12', 'board' : 'esp8266', 'topic' : '/esp8266/12', 'state' :
       'False'},# This is used for sensor
105    16 : {'name' : 'GPIO 16', 'board' : 'esp8266', 'topic' : '/esp8266/16', 'state' :
       'False'}
106    }
107
108 # Put the pin dictionary into the template data dictionary:
109 templateData = {
110    'pins' : pins
111    }
112
113 @app.route("/")
114 def main():
115    # Pass the template data into the template main.html and return it to the user
116    return render_template('main.html', async_mode=socketio.async_mode, **templateData)
117
118 # The function below is executed when someone requests a URL with the pin number and
    action in it:
119 @app.route("/<board>/<changePin>/<action>")
120 def action(board, changePin, action):
121    # Convert the pin from the URL into an integer:
122    changePin = int(changePin)
123    # Get the device name for the pin being changed:
```

```
124       devicePin = pins[changePin]['name']
125       # If the action part of the URL is "1" execute the code indented below:
126       if action == "1" and board == 'esp8266':
127          mqttc.publish(pins[changePin]['topic'],"1",qos=0,retain=True)
128          pins[changePin]['state'] = 'True'
129       if action == "0" and board == 'esp8266':
130          mqttc.publish(pins[changePin]['topic'],"0",qos=0,retain=True)
131          pins[changePin]['state'] = 'False'
132       # Along with the pin dictionary, put the message into the template data dictionary:
133       templateData = {
134          'pins' : pins
135       }
136       return render_template ('main.html', async_mode=socketio.async_mode, **templateData)
          #Changed, removed (**templateData)
137
138    @socketio.on('my event', namespace='/broker')
139    def broker_event(json): # Could also be bytes?
140        print('received json data here: ' + str(json)) # Could also be bytes?
141
142
143    @socketio.on('connect', namespace='/broker')
144    def broker_connect():
145        global x, y, z
146        print('connection established')
147        socketio.emit('dht_temperature', {'data': x}, namespace='/broker') # Send
           Temperature Data to WebPage
148        #sleep(0.5) # 0.5 second delay
149        socketio.emit('dht_humidity', {'data': y}, namespace='/broker') # Send Humidity
           Data to WebPage
150        #sleep(0.5) # 0.5 second delay
151        socketio.emit('light_lux', {'data': z}, namespace='/broker') # Send Light Data to
           WebPage
152        #pass
153
154    if __name__ == "__main__":
155       socketio.run(app, host='0.0.0.0', port=8181, debug=True) # Web Page Server
156
157
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Raspberry Pi Broker Web Server</title>
       <meta charset="utf-8">
       <meta name="viewport" content="width=device-width, initial-scale=1">
        <!-- The below code refreshes page every 10 seconds -->
        <meta http-equiv="refresh" content="10" >
       <!-- Get rid of favicon Error -->
       <link rel="shortcut icon" href="">
       <!-- Latest compiled and minified CSS -->
       <link rel="stylesheet"
       href="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css ">
       <!-- jQuery library -->
       <script
       src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js "></script>
       <!-- Popper JS -->
       <script
       src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js "></
       script>
       <!-- Latest compiled JavaScript -->
       <script
       src="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js "></script
       >
       <!-- Required For Socketio-->
       <script
       src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/1.3.6/socket.io.min.js "></sc
       ript>
       <br>
  </head>
  <body>
    <div class="container">
     <div class="jumbotron">
      <div><h1><b><u>Raspberry Pi Broker Web Server - ESP8266 MQTT</u></b></h1></div>
      <div><h1>Auto-Updates Every 10 Seconds</h1></div>
      <div><h2>Page Updated(<span id="WebPage_Updated"></span>)</h2></div>
      <br>
      <div>{% for pin in pins %}
      <h2>{{ pins[pin].name }}
      {% if pins[pin].state == 'True' %}
      is currently <strong>on</strong></h2><div class="row"><div class="col-md-2">
      <a href="/esp8266/{{pin}}/0" class="btn btn-block btn-lg btn-primary"
      role="button">Turn off</a></div></div>
      {% else %}
      is currently <strong>off</strong></h2><div class="row"><div class="col-md-2">
      <a href="/esp8266/{{pin}}/1" class="btn btn-block btn-lg btn-primary"
      role="button">Turn on</a></div></div>
      {% endif %}
      {% endfor %}</div>
      <br>
      <div><h2><b><u>Sensor Readings</u></b></h2></div>
      <div><h3>Temperature: <span id="temperature"></span>ºF</h3></div>
      <div><h3>Humidity: <span id="humidity"></span>%RH</h3></div>
      <div><h3>Light Intensity: <span id="intensity"></span>Lux</h3></div>
     </div>
    </div>
     <script>
      // Note - socket.emit "Sends Data" and socket.on "Listens for Data"
      $(document).ready(function() {
        //Connect to the socket server.
        var socket = io.connect('http://' + document.domain + ':' + location.port +
        '/broker');
        //Receive details from socket server
        socket.on('connect', function() {
        console.log("Socket Web-Page Connected");
      // Send a JSON response back to client broker server over socket
        socket.emit('my event', {data: 'I\'m connected!'});
```

```
          // Display Clock for Page Update
          var nDate = new Date();
          console.log("Updating Time");
            $('#WebPage_Updated').text(nDate.getHours() + 'h:' + nDate.getMinutes() +
                'm:' + nDate.getSeconds() + 's').html();
          });
        // Listen for Temperature, Humidity, and Lux Data.
          socket.on('dht_temperature', function(msg) {
          console.log("Temperature Is: " + msg.data);
           $('#temperature').html(msg.data);

          });
          socket.on('dht_humidity', function(msg) {
          console.log("Humidity Is: " + msg.data);
            $('#humidity').html(msg.data);
          });
          socket.on('light_lux', function(msg) {
          console.log("Light Intensity Is: " + msg.data);
              $('#intensity').html(msg.data);
          });
        });
      </script>
    </body>
  </html>


```

```
"""
Program to read a DHT11 & Lux Sensor, and send to Raspberry Pi Broker via MQTT
================================================================================
 - For ESP8266 firmware and micropython setup, see my separate document.
 - Uncoment sections to include OLED display & Lux values for light
 - No resistor is needed on DHT as the "Internal Pull-Up" is activated.
 - Control the red LED State through a "subcription"

Notes:
-----
1) To Enter the WiFi SSID type:
 f = open('SSID.txt', 'w')
 f.write('xxxxxxxxxx')
 f.close()
2) To Enter the WiFi Passworrd type:
 f = open('PASS.txt', 'w')
 f.write('xxxxxxxxxx')
 f.close()
3) When testing is complete, save this program as "main.py" on ESP8266
or add the program "MQTT_Sensor_R1" to the directory, and add this line
to the "import MQTT_Sensor_R1" to the "main.py" file.
"""

# Import Modules:
# ---------------
from time import sleep
import time
from ubinascii import hexlify
from machine import unique_id
from umqtt.simple import MQTTClient
from machine import Pin
import machine
import network
#from ssd1306 import SSD1306_I2C
from machine import I2C
from dht import DHT11 # Change to DHT22 for the 22 type sensor
import os
import tsl2561

sleep(5)

# Constants:
# ----------
DHT_PIN = 14
WIFI_LED_PIN = 2
SUB_LED_PIN = 0
MESUREMENT_INTERVAL = 5 # 5 Seconds
SERVER = "192.168.1.20"  # Rasperry pi IP Address
CLIENT_ID = hexlify(unique_id())
TOPIC1 = b"/sensor1/temp"
TOPIC2 = b"/sensor1/hum"
TOPIC3 = b"/esp8266/0"
TOPIC4 = b"/sensor3/lux"
TOPIC5 = b"/esp8266/4"
TOPIC6 = b"/esp8266/5"
#TOPIC7 = b"/esp8266/12"
TOPIC8 = b"/esp8266/16"


# Initial Setup
# -------------
d = DHT11(Pin(DHT_PIN, Pin.IN, Pin.PULL_UP))# Change to DHT22 for the 22 type sensor
blueled = Pin(WIFI_LED_PIN, Pin.OUT, value=1) # Sets up LED and starts off
redled = Pin(SUB_LED_PIN, Pin.OUT, value=1) # Sets up LED and starts off
GPIO4_OUT = Pin(4, Pin.OUT, value=0) # Sets up GPIO 4 as ouput, initially off
GPIO5_OUT = Pin(5, Pin.OUT, value=0) # Sets up GPIO 5 as ouput, initially off
#GPIO12_OUT = Pin(12, Pin.OUT, value=0) # Sets up GPIO 12 as ouput, initially off
```

```
GPIO16_OUT = Pin(16, Pin.OUT, value=0) # Sets up GPIO 13 as ouput, initially off
last_mesurement_time = time.time() # Create an initial time stamp
i2c = I2C(sda = Pin(12), scl = Pin(13)) # IIC Pins
sensor = tsl2561.TSL2561(i2c)
sensor.active(True)
#display = SSD1306_I2C(128, 32, i2c) # IIC for Display

def Lux_Read():
    y = sensor.read()
    print("Lux Value =: " + str(y))
    return y

# Generic Function to Display on OLED (Optional)
# ----------------------------------------------
def Display_Stat(line1,line2,line3,delaytime):
    display.fill(0)
    display.text(line1,0,0,1)
    display.text(line2,0,12,1)
    display.text(line3,0,24,1)
    display.show()
    sleep(delaytime)

# Function to Read Temp and Humidity
# ----------------------------------
def TemHum():
    try:
        d.measure()
        t = d.temperature()
        tf = (t*(9/5.0))+32
        h = d.humidity()
        print("Temperature Deg.F. =: " + str(tf))
        print("% Humidity =: " + str(h))
        return (tf,h)
    except Exception as e:
                        print(e)
                        return (0,0)

# Function Check to see if Station mode wifi is active
# ----------------------------------------------------
def stat_mode():
    sta_if = network.WLAN(network.STA_IF) # Set up Station Mode case
    if (sta_if.isconnected()): # If wifi is connected station mode
        blueled.off() # Turn on blue LED
        return
    else: # If not in Wifi station mode then try to connect
        blueled.on() # Turn off blue LED
        sta_if.active(True) # Station Mode is turned on
        sta_if.connect(SSID, PASS) # Configure WiFi, if not already connected
        sleep(1) # time delay 1 seconds
        attempt = 0
        while (attempt < 11 and not sta_if.isconnected()): # Try to connect
            print('Connecting.....')
            print('Attempt: ' + str(attempt))
            #L1 = "Connecting....."
            #L2 = "Attempt: " + str(attempt)
            #L3 = ""
            #delayx = 1
            #Display_Stat(L1,L2,L3,delayx) # Goto display function
            #attempt += 1
        if sta_if.isconnected(): # If we connect
            print('WiFi Connected')
            #L1 = "WiFi Connected"
            #L2 = ""
            #L3 = ""
            #delayx = 2
            #Display_Stat(L1,L2,L3,delayx) # Goto display function
            machine.reset()
```

```
135         else:  # If connection fails
136             print('WiFi Connection Failed')
137             #L1 = "WiFi Connection"
138             #L2 = "Failed"
139             #L3 = ""
140             #delayx = 2
141             #Display_Stat(L1,L2,L3,delayx) # Goto display function
142             while True: #Stay Here as there is a problem
143                 pass
144 
145 def envioMQTT(server=SERVER, topic="/stuff", data=None):
146     try:
147         c = MQTTClient(CLIENT_ID, server)
148         c.connect()
149         c.publish(topic, data)
150         sleep(0.2)
151         c.disconnect()
152     except Exception as e:
153         print(e)
154         pass
155 
156 #state = 0
157 
158 def sub_cb(topic, msg):
159     print((topic, msg))
160     if topic == b"/esp8266/0":
161         if msg == b"1":
162             redled.off()
163         elif msg == b"0":
164             redled.on()
165     if topic == b"/esp8266/4":
166         if msg == b"1":
167             GPIO4_OUT.on()
168         elif msg == b"0":
169             GPIO4_OUT.off()
170     if topic == b"/esp8266/5":
171         if msg == b"1":
172             GPIO5_OUT.on()
173         elif msg == b"0":
174             GPIO5_OUT.off()
175     #if topic == "/esp8266/12":
176         #if msg == b"1":
177             #GPIO12_OUT.on()
178         #elif msg == b"0":
179             #GPIO12_OUT.off()
180     if topic == b"/esp8266/16":
181         if msg == b"1":
182             GPIO16_OUT.on()
183         elif msg == b"0":
184             GPIO16_OUT.off()
185 
186 def recepcionMQTT(server, topic):
187     c = MQTTClient(CLIENT_ID, server)
188     # Subscribed messages will be delivered to this callback
189     c.set_callback(sub_cb)
190     c.connect()
191     c.subscribe(topic)
192     print("Connected to %s, subscribed to %s topic" % (server, topic))
193     try:
194         c.wait_msg()
195     finally:
196         c.disconnect()
197 
198 # Display Initial Start Banner
199 # ----------------------------
200 #L1 = "MQTT Sensor"
201 #L2 = "Server to Ras-Pi"
```

```
#L3 = "By, Roy Guerra"
#delayx = 5
#Display_Stat(L1,L2,L3,delayx) # Goto display function

# Check for Missing Files or Read Files
# -------------------------------------
global SSID,PASS
try:
    a = open('SSID.txt')
    SSID = a.read() # Open and read SSID from file
    a.close()
    b = open('PASS.txt') # Open and read Password from file
    PASS = b.read()
    b.close()
    print("SSID : " + SSID)
    print("PASS : " + PASS)
except Exception as e:
                    print(e)
                    #L1 = "Cannot Find"
                    #L2 = "SSID/Pass or"
                    #L3 = "API_KEY Files"
                    #delayx = 3
                    #Display_Stat(L1,L2,L3,delayx) # Goto function

# Start of Main Program:
# ----------------------
while True:
    try:
        stat_mode()# Goto Function
        current_time = time.time() # Get a time stamp
        recepcionMQTT(SERVER,TOPIC3)
        sleep(0.1)
        recepcionMQTT(SERVER,TOPIC5)
        sleep(0.1)
        recepcionMQTT(SERVER,TOPIC6)
        sleep(0.1)
        #recepcionMQTT(SERVER,TOPIC7)
        #sleep(0.1)
        recepcionMQTT(SERVER,TOPIC8)
        sleep(0.1)
        if current_time - last_mesurement_time > MESUREMENT_INTERVAL:
            (tf,h) = TemHum() # Goto get temp & hum Function
            l = Lux_Read() # Goto light read function and return Lux value
            # Place OLED display function here
            envioMQTT(SERVER,TOPIC1,str(tf))
            envioMQTT(SERVER,TOPIC2,str(h))
            envioMQTT(SERVER,TOPIC4,str(l))
            last_mesurement_time = current_time # New time stamp
    except Exception as e:
                    print(e)
```