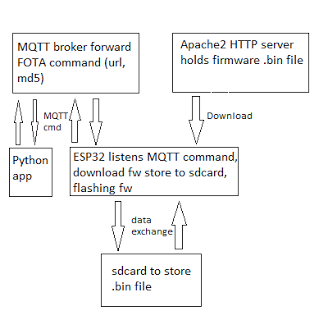
# **[Demo 34: firmware update OTA for ESP32 using HTTP and sdcard](http://www.iotsharing.com/2017/11/firmware-update-ota-for-esp32-using-http-sdcard.html)**

**1. Introduction**  
In [Demo 16](http://www.iotsharing.com/2017/05/how-to-update-firmware-ota-for-batch-esp32.html), we knew how to update firmware OTA for a batch of Arduino ESP32. Actually, the method behind that tutorial is UDP protocol. In this tutorial I will show you another way to update firmware for ESP OTA. That is using HTTP and sdcard. It means we will use Http to download the firmware from Http Apache server to sdcard and then flashing downloaded firmware from sdcard to ESP.  
**Note**: In case you do not want to store firmware file in sdcard, I will make another demo without using sdcard [Demo 35](http://www.iotsharing.com/2017/11/firmware-update-ota-for-esp32-using-http.html).  
We will re-use these demos for this tutorial:  
+ [Demo 14: How to use MQTT and Arduino ESP32 to build a simple Smart home system](http://www.iotsharing.com/2017/05/how-to-use-mqtt-to-build-smart-home-arduino-esp32.html" \t "http://www.iotsharing.com/2017/11/_blank)  
+ [Demo 7: How to use Arduino ESP32 to store data to sdcard](http://www.iotsharing.com/2017/05/how-to-use-arduino-esp32-to-store-data-to-sdcard.html" \t "http://www.iotsharing.com/2017/11/_blank)  
+ [Demo 1: Blinky - a Hello World on Arduino ESP32](http://www.iotsharing.com/2017/05/blinky-hello-world-on-arduino-esp32.html" \t "http://www.iotsharing.com/2017/11/_blank)   
+ [How to turn the Orange Pi/Raspberry Pi into an IoT node](http://www.iotsharing.com/2017/06/how-to-turn-the-Orange-Pi-into-an-IoT-node.html) (this tutorial will show you how to install Apache2 HTTP server to hold firmware .bin file. Following the steps in **2.4** of the tutorial and copy the .bin file to "**/var/www/html/phphello**". So the url to download the firmware file (**led.bin**) is: **http://192.168.1.107/phphello/led.bin**)  
  
**Note**: the firmware file "led.bin" is exported from Arduino IDE. It is a simple LED blink application that  blink the LED on pin GPIO4. In order to export the .bin file from Arduino IDE Menu, we choose **Sketch -> Export compiled Binary**.After finishing we choose **Sketch -> Show Sketch Folder**. You will se the .bin file there, rename it as **led.bin**  
The model of this demo is below:

**[](https://4.bp.blogspot.com/-pTtCYh6oXPg/WhI-C6kNqPI/AAAAAAAAFsU/Ycz5NKzP9sIL_ZBxvzdo2Z4fvcSmNztpgCLcBGAs/s1600/httpota.png)**

**Figure: Model of demo**

ESP will receive MQTT command (including **url** to download .bin firmware file and **md5** value of that file to verify whether downloading and flashing firmware is correct or not) from Python application "fotacontrol.py". After downloading and flashing process was finished the ESP send "ok" to Python application to notify the updating process was finished.  
  
**2. Hardware**  
Refer [Demo 7](http://www.iotsharing.com/2017/05/how-to-use-arduino-esp32-to-store-data-to-sdcard.html" \t "http://www.iotsharing.com/2017/11/_blank) to connect ESP32 pins with sdcard module. And [Demo 1](http://www.iotsharing.com/2017/05/blinky-hello-world-on-arduino-esp32.html" \t "http://www.iotsharing.com/2017/11/_blank)to connect ESP to LED.  
**3. Software**  
I made the library in [github](https://github.com/nhatuan84/esp32-http-ota). Just download, install it and run example **esp32httpota2**. **Change the Wifi ssid, password and IP of MQTT server according to yours**. The code will be explained below.

|  |
| --- |
| #include <WiFi.h>  #include <HttpOTA.h>  #include <mySD.h>  #include <PubSubClient.h>  typedef enum {  Runnning\_e = 0x01,  Fota\_e  }SysState;  const char\* ssid = "dd-wrt";  const char\* password = "0000000000";  const char\* mqtt\_server = "192.168.1.107";  char url[100];  char md5[50];  File file;  WiFiClient espClient;  PubSubClient client(espClient);  SysState state = Runnning\_e;  /\* topics \*/  #define OTA\_TOPIC "smarthome/room1/ota"  void saveData(uint8\_t \*buffer, int bytes){  file.write(buffer, bytes);  }  int readData(uint8\_t \*buffer, int bytes){  return file.read(buffer, bytes);  }  void progress(DlState state, int percent){  Serial.printf("state = %d - percent = %d\n", state, percent);  }  void receivedCallback(char\* topic, byte\* payload, unsigned int length) {    if(strncmp(OTA\_TOPIC, topic, strlen(OTA\_TOPIC)) == 0){  memset(url, 0, 100);  memset(md5, 0, 50);  char \*tmp = strstr((char \*)payload, "url:");  char \*tmp1 = strstr((char \*)payload, ",");  memcpy(url, tmp+strlen("url:"), tmp1-(tmp+strlen("url:")));    char \*tmp2 = strstr((char \*)payload, "md5:");  memcpy(md5, tmp2+strlen("md5:"), length-(tmp2+strlen("md5:")-(char \*)&payload[0]));  Serial.printf("started fota url: %s\n", url);  Serial.printf("started fota md5: %s\n", md5);  state = Fota\_e;  }  }  void mqttconnect() {  /\* Loop until reconnected \*/  while (!client.connected()) {  Serial.print("MQTT connecting ...");  /\* client ID \*/  String clientId = "ESP32Client";  /\* connect now \*/  if (client.connect(clientId.c\_str())) {  Serial.println("connected");  /\* subscribe topic \*/  client.subscribe(OTA\_TOPIC);  } else {  Serial.print("failed, status code =");  Serial.print(client.state());  Serial.println("try again in 5 seconds");  /\* Wait 5 seconds before retrying \*/  delay(5000);  }  }  }  void error(char \*message){  printf("%s\n", message);  }  void startDl(void){  //write bin file to sdcard  SD.remove("fw.bin");  file = SD.open("fw.bin", FILE\_WRITE);  }  void endDl(void){  file.close();  }  void startFl(void){  //write bin file to sdcard  file = SD.open("fw.bin", FILE\_READ);  }  void endFl(void){  file.close();  }  void setup() {  // put your setup code here, to run once:  // put your setup code here, to run once:  Serial.begin(115200);  Serial.print("Connecting to ");  Serial.print(ssid);  WiFi.begin(ssid, password);  while (WiFi.status() != WL\_CONNECTED) {  delay(500);  Serial.print(".");  }  Serial.println("");    Serial.print("WiFi connected, IP address: ");  Serial.println(WiFi.localIP());    if (!SD.begin(32, 14, 12, 27)) {  Serial.println("initialization failed!");  return;  }  Serial.println("initialization done.");  /\* configure the MQTT server with IPaddress and port \*/  client.setServer(mqtt\_server, 1883);  /\* this receivedCallback function will be invoked  when client received subscribed topic \*/  client.setCallback(receivedCallback);  }    void loop() {  switch(state)  {  case Runnning\_e:  /\* if client was disconnected then try to reconnect again \*/  if (!client.connected()) {  mqttconnect();  }  /\* this function will listen for incomming  subscribed topic-process-invoke receivedCallback \*/  client.loop();  break;  case Fota\_e:  DlInfo info;  info.url = url;  info.md5 = md5;  info.startDownloadCallback = startDl;  info.endDownloadCallback = endDl;  info.startFlashingCallback = startFl;  info.endFlashingCallback = endFl;    info.saveDataCallback = saveData;  info.readDataCallback = readData;  info.progressCallback = progress;  info.errorCallback = error;  httpOTA.start(info);    client.publish(OTA\_TOPIC, "ok");  break;  default:  break;  }  } |

In order to start fota updating process, using the Python code below and type "1" to publish FOTA command to ESP. The ESP will receive  the command, parse it and starting FOTA process. **Change the url and md5 value according to yours.**  
The library is simple. It only has 1 interface that is start with input is **DlInfo**structure. When calling this function, it will block the **loop()** function. We must initialize the structure before using it. The structure includes:  
- **url**: is the http url of the firmware file.  
- **md5**: is the md5 checksum of the firmware file.  
- **startDownloadCallback**: is the function that will be invoked before starting downloading. I used it to open firmware file in sdcard for writing downloaded data.  
- **endDownloadCallback**: is the function that will be invoked after downloading was finished. I used it to close firmware file in sdcard after finishing writing downloaded data.  
- **startFlashingCallback**: is the function that will be invoked before starting flashing downloaded firmware. I used it to open firmware file for flashing.  
- **endFlashingcallback**: is the function that will be invoked after flashing was finished. I used it to close firmware file after finishing flashing.  
- **saveDataCallback**: is the function that will be invoked to save downloading data to sdcard. I made it in generic way so that the library can be applied for SPIFFS.  
- **readDataCallback**: is the function that will be invoked to read data from sdcard for flashing. I made it in generic way so that the library can be applied for SPIFFS.  
- **progressCallback**: is the function that will be invoked to show the progress of downloading and flashing process.  
- **errorCallback**: is the function that will be invoked to show the error of downloading and flashing process.  
The functions: s**aveData, readData, progress, error, startDl, endDl, startFl, endFl** will be invoked by the library in the update process. They are assigned to the members of **DlInfo** structure.  
The function **receiveCallback** will be invoked whenever ESP received the MQTT command. It will check if the topic is OTA\_TOPIC then parsing the payload to get url and md5 value of firmware file. After that, it change the state of system to **Fota\_e** to start updating process.  
After finishing the whole process, the code "**client.publish(OTA\_TOPIC, "ok");**" will publish message "ok" back to Python application.    
When the system is in **Running\_e**, it just listen the MQTT command.  
The code "**SD.begin(32, 14, 12, 27)**" is ti initialize SD card before operating on it.

|  |
| --- |
| from random import randint  import thread  import sys  try:  import paho.mqtt.client as mqtt  except ImportError:  import os  import inspect  cmd\_subfolder = os.path.realpath(os.path.abspath(os.path.join(os.path.split(inspect.getfile( inspect.currentframe() ))[0],"../src")))  if cmd\_subfolder not in sys.path:  sys.path.insert(0, cmd\_subfolder)  import paho.mqtt.client as mqtt  server = "192.168.1.107";  topic = "smarthome/room1/ota"  def on\_connect(mqttc, obj, flags, rc):  print("rc: "+str(rc))  def on\_message(mqttc, obj, msg):  print(msg.topic+" "+str(msg.qos)+" "+str(msg.payload))  def on\_publish(mqttc, obj, mid):  print("mid: "+str(mid))  def on\_subscribe(mqttc, obj, mid, granted\_qos):  print("Subscribed: "+str(mid)+" "+str(granted\_qos))  def on\_log(mqttc, obj, level, string):  print(string)  mqttc = mqtt.Client()  mqttc.on\_message = on\_message  mqttc.on\_connect = on\_connect  mqttc.on\_publish = on\_publish  mqttc.on\_subscribe = on\_subscribe  mqttc.connect(server, 1883, 60)  mqttc.subscribe(topic, 0)  def fotaControl( threadName, delay):  while True:  val = raw\_input('Enter 1 to update firmware OTA ')  if(val == "1"):  mqttc.publish(topic, "url:http://192.168.1.107/phphello/led.bin,md5:6bd07139c21f572370242905c4465056")  try:  thread.start\_new\_thread( fotaControl, ("Fota Control", 0, ) )  except:  print "Error: unable to start thread"  mqttc.loop\_forever() |

The Python application is quite simple, It has 2 thread: listening MQTT thread and fota control thread to send FOTA command to ESP (pressing "1" to start FOTA).  
Before using paho MQTT we need to initialize some callback functions: **on\_message, on\_connect, on\_publish, on\_subscribe**.