# **[Demo 16: How to update firmware OTA for a batch of Arduino ESP32](http://www.iotsharing.com/2017/05/how-to-update-firmware-ota-for-batch-esp32.html)**

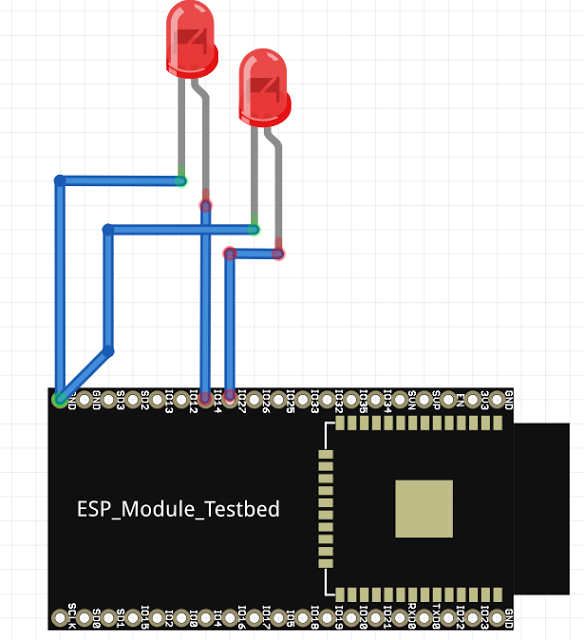
**1. Introduction**

- IoT (Internet of Things) means many devices connect to the Internet. If there is bug in the software of devices, the process to fix bug and re-flash the software will cost much effort, time and money. That is why people use FOTA (firmware update over the air) for IoT devices. You do not need to unplug the device from the network and use hardware flashing tool to re-flash software for it. Everything happens over the air, just create the wireless connection to the device and transfer the software to it chunk by chunk. The device will receive the chunk and write it to a specific address in Flash. After finishing flashing, the device will reboot and run the new software. It is similar to update software for your mobile over the WiFi/3G.

- We will create a demo for FOTA, first we flash the firmware with FOTA functionality and LED blinky on pin GPIO12 by normal way (USB-TTL port). Then we update the firmware with FOTA functionality but LED blinky on pin GPIO14 by FOTA way.

**2. Hardware**

We connect 2 LEDs on GPIO12 and GPIO14. After first flashing, LED on GPIO12 will blink. But after second flashing, LED on GPIO14 will blink.

[](https://1.bp.blogspot.com/-n4ZUk_T5CQs/WSxYW2y1awI/AAAAAAAAD-8/758mvchT7YYNoT2bCz27utkCpMOGTcVKwCLcB/s1600/ota.png)

**3. Software**

- Arduino ESP32 core supplied ArduinoOTA library for us to use FOTA functionality. So we just include **"ArduinoOTA.h"**. I will explain the code by the comments in code, so you just read it.  
Note: FOTA functionality will use use 2 TCP/Ip connections. First connection is to communicate with ESP32 to control flashing work flow. Second connection is to transfer the firmware chunk by chunk.  
- Create an Arduino project with code below, save as **esp32otafw1**and flash it to ESP32 by normal way:

|  |
| --- |
| #include <WiFi.h>  #include <ESPmDNS.h>  #include <WiFiUdp.h>  #include <ArduinoOTA.h>  /\* fill your ssid and password here \*/  const char\* ssid = "dd-wrt";  const char\* password = "0000000000";  long lastMsg = 0;  int flag = false;  /\* LED is pin GIO12 \*/  int led = 12;  void setup() {    Serial.begin(115200);  /\* connect to wifi \*/  WiFi.mode(WIFI\_STA);  WiFi.begin(ssid, password);  /\* Wait for connection \*/  while (WiFi.status() != WL\_CONNECTED) {  delay(500);  Serial.print(".");  }  /\* set LED as output \*/  pinMode(led, OUTPUT);    /\* create a connection at port 3232 \*/  ArduinoOTA.setPort(3232);  /\* we use mDNS instead of IP of ESP32 directly \*/  ArduinoOTA.setHostname("esp32");  /\* we set password for updating \*/  ArduinoOTA.setPassword("iotsharing");  /\* this callback function will be invoked when updating start \*/  ArduinoOTA.onStart([]() {  Serial.println("Start updating");  });  /\* this callback function will be invoked when updating end \*/  ArduinoOTA.onEnd([]() {  Serial.println("\nEnd updating");  });  /\* this callback function will be invoked when a number of chunks of software was flashed  so we can use it to calculate the progress of flashing \*/  ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {  Serial.printf("Progress: %u%%\r", (progress / (total / 100)));  });  /\* this callback function will be invoked when updating error \*/  ArduinoOTA.onError([](ota\_error\_t error) {  Serial.printf("Error[%u]: ", error);  if (error == OTA\_AUTH\_ERROR) Serial.println("Auth Failed");  else if (error == OTA\_BEGIN\_ERROR) Serial.println("Begin Failed");  else if (error == OTA\_CONNECT\_ERROR) Serial.println("Connect Failed");  else if (error == OTA\_RECEIVE\_ERROR) Serial.println("Receive Failed");  else if (error == OTA\_END\_ERROR) Serial.println("End Failed");  });  /\* start updating \*/  ArduinoOTA.begin();  Serial.print("ESP IP address: ");  Serial.println(WiFi.localIP());  }  void loop() {  /\* this function will handle incomming chunk of SW, flash and respond sender \*/  ArduinoOTA.handle();  /\* we can not use delay() here, because it will block the ArduinoOTA work  so we count the time, if it reach we toggle the lED \*/  long now = millis();  if (now - lastMsg > 3000) {  lastMsg = now;  if(flag == false){  digitalWrite(led, HIGH);  flag = true;  }else{  digitalWrite(led, LOW);  flag = false;  }  }  } |

After flashing, you will see the LED on GPIO12 blink  
- Create an Arduino project, save as **esp32otafw2**with code:

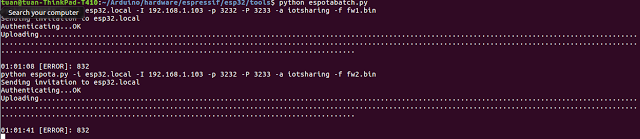
|  |
| --- |
| #include <WiFi.h>  #include <ESPmDNS.h>  #include <WiFiUdp.h>  #include <ArduinoOTA.h>  /\* fill your ssid and password here \*/  const char\* ssid = "dd-wrt";  const char\* password = "0000000000";  long lastMsg = 0;  int flag = false;  /\* LED is pin GIO12 \*/  int led = 14;  void setup() {    Serial.begin(115200);  /\* connect to wifi \*/  WiFi.mode(WIFI\_STA);  WiFi.begin(ssid, password);  /\* Wait for connection \*/  while (WiFi.status() != WL\_CONNECTED) {  delay(500);  Serial.print(".");  }  /\* set LED as output \*/  pinMode(led, OUTPUT);    /\* create a connection at port 3232 \*/  ArduinoOTA.setPort(3232);  /\* we use mDNS instead of IP of ESP32 directly \*/  ArduinoOTA.setHostname("esp32");  /\* we set password for updating \*/  ArduinoOTA.setPassword("iotsharing");  /\* this callback function will be invoked when updating start \*/  ArduinoOTA.onStart([]() {  Serial.println("Start updating");  });  /\* this callback function will be invoked when updating end \*/  ArduinoOTA.onEnd([]() {  Serial.println("\nEnd updating");  });  /\* this callback function will be invoked when a number of chunks of software was flashed  so we can use it to calculate the progress of flashing \*/  ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {  Serial.printf("Progress: %u%%\r", (progress / (total / 100)));  });  /\* this callback function will be invoked when updating error \*/  ArduinoOTA.onError([](ota\_error\_t error) {  Serial.printf("Error[%u]: ", error);  if (error == OTA\_AUTH\_ERROR) Serial.println("Auth Failed");  else if (error == OTA\_BEGIN\_ERROR) Serial.println("Begin Failed");  else if (error == OTA\_CONNECT\_ERROR) Serial.println("Connect Failed");  else if (error == OTA\_RECEIVE\_ERROR) Serial.println("Receive Failed");  else if (error == OTA\_END\_ERROR) Serial.println("End Failed");  });  /\* start updating \*/  ArduinoOTA.begin();  Serial.print("ESP IP address: ");  Serial.println(WiFi.localIP());  }  void loop() {  /\* this function will handle incomming chunk of SW, flash and respond sender \*/  ArduinoOTA.handle();  /\* we can not use delay() here, because it will block the ArduinoOTA work  so we count the time, if it reach we toggle the lED \*/  long now = millis();  if (now - lastMsg > 3000) {  lastMsg = now;  if(flag == false){  digitalWrite(led, HIGH);  flag = true;  }else{  digitalWrite(led, LOW);  flag = false;  }  }  } |

- In order to transfer the new firmware to ESP32 OTA we will use a python tool called **"espota.py**" in   
**["where\_you\_install\_Arduino/Arduino/hardware/espressif/esp32/tools"](http://www.iotsharing.com/2017/05/introduction-to-esp32.html" \t "http://www.iotsharing.com/2017/05/_blank)**  
This tool has arguments:  
**"python espota.py -i ESP\_IP\_address -I Host\_IP\_address -p ESP\_port -P Host\_port [-a password] -f sketch.bin"**  
**Note:**in order to get **"sketch.bin"**, from Arduino menu bar, **choose  Sketch - Export Compiled Binary**  
and then   
from Arduino menu bar, choose**Sketch - Show Sketch Folder**- you will see a **".bin"** file there. So get the absolute path to this **".bin"** file for later usage.  
- In order to update for multiple ESP32 clients I created a small Python script **"espotabatch.py"**, you just change a little bit to adapt your need. This script will be put in the same folder with "espota.py"

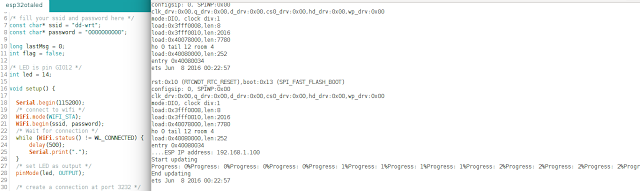
|  |
| --- |
| import subprocess  import time  #this list contains array of esp32 clients,  # and each client contains mDNS name and the path to .bin file  #I only have 1 ESP so I duplicate mDNS entry for testing  esps = [  #mDNS name of ESP #absolute path to ".bin" file  ['esp32.local', 'fw1.bin'],  ['esp32.local', 'fw2.bin']  ]  ip\_of\_sender = '192.168.1.103'  esp\_respond\_sender\_port = '3232'  sender\_to\_esp\_port = '3232'  update\_password = 'iotsharing'  for esp in esps:  cmd = 'python espota.py -i '+esp[0]+' -I '+ip\_of\_sender+ ' -p '+sender\_to\_esp\_port+' -P '+esp\_respond\_sender\_port+' -a '+update\_password+' -f '+esp[1]  print (cmd)  p = subprocess.Popen(cmd, shell=True, stdout=subprocess.PIPE, stderr=subprocess.STDOUT)  for line in p.stdout.readlines():  print line,  retval = p.wait()  time.sleep(10) |

- Now open Terminal in the path**:**  
**"where\_you\_install\_Arduino/Arduino/hardware/espressif/esp32/tools"**  
and type:**"python espotabatch.py"**

1. **Result**

[](https://3.bp.blogspot.com/-3W7YhOXeUr4/WSxizgWIc9I/AAAAAAAAD_M/GJVkhhEdB8wcLo-GlU9LnXivixRX8cK5wCLcB/s1600/ota2.png)

If you see ERROR here, do not worry, no problem.

[](https://1.bp.blogspot.com/-TRfhkSOiwpE/WSxizkOOjtI/AAAAAAAAD_Q/FxhbA6lqVfQYBNgqbkDt5cQEIUISr9xQACLcB/s1600/ota3.png)

**Figure: FOTA progress**