3 Simple Steps to Use Basic OTA with ESP32

Installing Python 2.7.x series: The first step is to install Python 2.7.x series in your computer.

Download Basic OTA Firmware Serially: Serially download the code containing the OTA firmware. It is a mandatory step, so you can perform the following updates or uploads live.

Download New Sketch Over-The-AirNow: You can upload new code to ESP32 from the Arduino IDE over the air.

Step 1: Install Python 2.7.x series

To use the OTA function, you need to have Python version 2.7.x installed, if it is not already installed on your device.

Go to the official Python website and download 2.7.x (selected version) for Windows (MSI installer).

Open the installer and follow the installation wizard.

In the Python 2.7 customization section. X, make sure the last option is enabled to add python.exe to the path

Step 2: Upload OTA Routine Serially

The factory image in ESP32 does not have the possibility of an OTA upgrade. Therefore, you need to download the OTA firmware on ESP32 through the serial interface first.

It is a mandatory step to update the firmware at first, so that you are able to make the following updates or downloads

via live streaming.

The ESP32 add-on for Arduino IDE comes with an example of an OTA and BasicOTA library. You can access it through File>Examples>ArduinoOTA>BasicOTA

The following code must be uploaded. But, before you head over to download the chart, you need to make a few changes to make it work for you. You need to modify the following two variables using your network credentials, so that ESP32 can establish a connection to the current network.

```
nclude <WiFi.h>
#include <ESPmDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>

const char* ssid = ".....";
const char* password = ".....";

void setup() {
  Serial.begin(115200);
  Serial.println("Booting");
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.waitForConnectResult() !=
WL_CONNECTED) {
    Serial.println("Connection Failed!
Rebooting...");
    delay(5000);
    ESP.restart();
  }

  // Port defaults to 3232
  // ArduinoOTA.setPort(3232);
```

```
  // Hostname defaults to esp3232-[MAC]
  // ArduinoOTA.setHostname("myesp32");

  // No authentication by default
  // ArduinoOTA.setPassword("admin");

  // Password can be set with it's md5 value as well
  // MD5(admin) = 21232f297a57a5a743894a0e4a801fc3
  // ArduinoOTA.setPasswordHash("21232f297a57a5a743894a0e4a801fc3");

  ArduinoOTA
    .onStart([]() {
      String type;
      if (ArduinoOTA.getCommand() == U_FLASH)
        type = "sketch";
      else // U_SPIFFS
        type = "filesystem";

      // NOTE: if updating SPIFFS this would be the place to unmount SPIFFS using SPIFFS.end()
      Serial.println("Start updating " + type);
    })
    .onEnd([]() {
      Serial.println("\nEnd");
    })
    .onProgress([](unsigned int progress, unsigned int total) {
      Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
    })
    .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");
    });

  ArduinoOTA.begin();

  Serial.println("Ready");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  ArduinoOTA.handle();
}
```

Now, open Serial Monitor at 115200. And press the EN button in ESP32. If all is well, the IP address obtained will come out of your router. Notice that .

Step 3: Upload New Sketch Over-The-Air

Now, let’s upload a new code on air.

Remember! You need to add an OTA code in every code you upload. Otherwise, you will lose the possibility of OTA and will not be able to make the following downloads live. Therefore, it is recommended that the code above be amended to include the new code.

For example, we will include a simple Blink code in the basic OTA code. Remember to modify SSID and password variables using network credentials.

Changes in the basic OTA software are highlighted in button

```
#include <WiFi.h>
#include <ESPmDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>

const char* ssid = ".........";
const char* password = ".........";

//variabls for blinking an LED with Millis
const int led = 2; // ESP32 Pin to which onboard LED is connected
unsigned long previousMillis = 0;  // will store last time LED was updated
const long interval = 1000;  // interval at which to blink (milliseconds)
int ledState = LOW;  // ledState used to set the LED

void setup() {

pinMode(led, OUTPUT);

  Serial.begin(115200);
  Serial.println("Booting");
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.waitForConnectResult() != WL_CONNECTED) {
    Serial.println("Connection Failed! Rebooting...");
    delay(5000);
    ESP.restart();
  }

  // Port defaults to 3232
  // ArduinoOTA.setPort(3232);

  // Hostname defaults to esp3232-[MAC]
  // ArduinoOTA.setHostname("myesp32");

  // No authentication by default
  // ArduinoOTA.setPassword("admin");

  // Password can be set with it's md5 value as well
  // MD5(admin) = 21232f297a57a5a743894a0e4a801fc3
  // ArduinoOTA.setPasswordHash("21232f297a57a5a743894a0e4a801fc3");

  ArduinoOTA
    .onStart([]() {
      String type;
      if (ArduinoOTA.getCommand() == U_FLASH)
        type = "sketch";
      else // U_SPIFFS
        type = "filesystem";

      // NOTE: if updating SPIFFS this would be the place to unmount SPIFFS using SF
      Serial.println("Start updating " + type);
    })
    .onEnd([]() {
      Serial.println("\nEnd");
    })
    .onProgress([](unsigned int progress, unsigned int total) {
      Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
    })
    .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");
    });

  ArduinoOTA.begin();

  Serial.println("Ready");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  ArduinoOTA.handle();

//loop to blink without delay
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
  // save the last time you blinked the LED
  previousMillis = currentMillis;
  // if the LED is off turn it on and vice-versa:
  ledState = not(ledState);
  // set the LED with the ledState of the variable:
  digitalWrite(led,  ledState);
  }

}
```

Once you have copied the code above to Arduino IDE, go to Tools>Port (Port option) and you will see something like this: esp32-xxxxxx at your_esp_ip_address If you can't find it, you may need to restart your IDE

Select the port and click the “Load” button. Within a few seconds, the new code will load. And you’ll see the LED flicker on board