Part 05
-
Over-The-Air Programming

Version: 2020-10-15

Over-the-air (OTA) Programming

A fantastic feature of any WiFi-enabled microcontroller like ESP32 is the ability to update its firmware wirelessly. This is known as Over-The-Air (OTA) programming.

What is OTA programming in ESP32?
The OTA programming allows updating/uploading a new program to ESP32 using Wi-Fi instead of requiring the user to connect the ESP32 to a computer via USB to perform the update.
OTA functionality is extremely useful in case of no physical access to the ESP module. It helps reduce the amount of time spent for updating each ESP module at the time of maintenance.
One important feature of OTA is that one central location can send an update to multiple ESPs sharing same network.
The only disadvantage is that you have to add an extra code for OTA with every sketch you upload, so that you're able to use OTA in the next update.

Ways To Implement OTA In ESP32
There are two ways to implement OTA functionality in ESP32.

- Basic OTA – Over-the-air updates are sent through Arduino IDE.
- Web Updater OTA – Over-the-air updates are sent through a web browser.

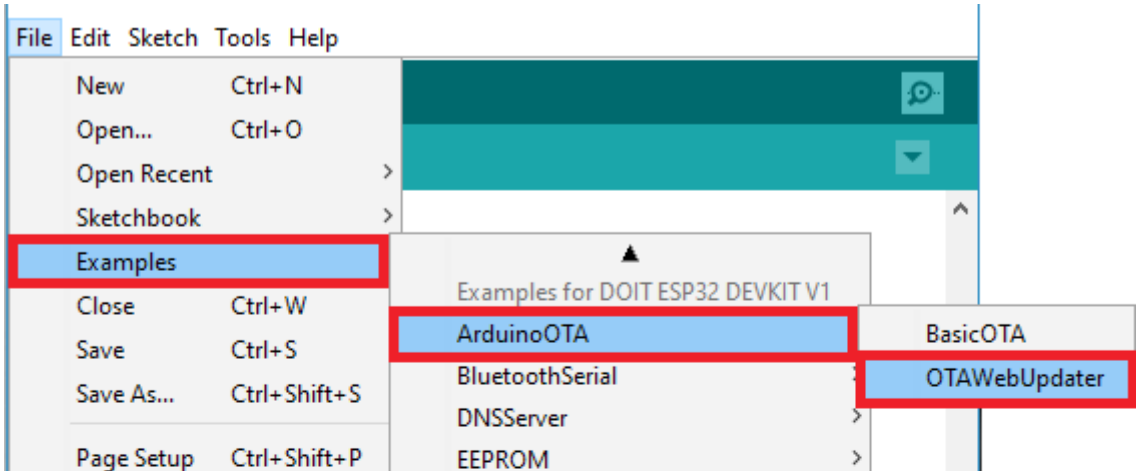
Each one has its own advantages. You can implement any one according to your project's requirement.

OTA programming is useful when you need to update code to ESP32 boards that are not easily accessible. The example we'll show here works when the ESP32 and your browser are on your local network.
The only disadvantage of the OTA Web Updater is that you have to add the code for OTA in every sketch you upload, so that you're able to use OTA in the future.

How does OTA Web Updater Work?

- The first sketch should be uploaded via serial port. This sketch should contain the code to create the OTA Web Updater, so that you are able to upload code later using your browser.
- The OTA Web Updater sketch creates a web server you can access to upload a new sketch via web browser.
- Then, you need to implement OTA routines in every sketch you upload, so that you're able to do the next updates/uploads over-the-air.

OTA Web Updater
When you installed the ESP32 add-on for the Arduino IDE, it has automatically installed the ArduinoOTA library. Go to `File > Examples >ArduinoOTA> OTAWebUpdater`.



The following code should load.

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include <ESPmDNS.h>
#include <Update.h>

const char* host = "esp32";
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

WebServer server(80);

/*
 * Login page
 */
const char* loginIndex =
 "<form name='loginForm'>"
    "<table width='20%' bgcolor='A09F9F' align='center'>"
        "<tr>"
            "<td colspan=2>"
                "<center><font size=4><b>ESP32 Login Page</b></font></center>"
                "<br>"
            "</td>"
            "<br>"
            "<br>"
        "</tr>"
        "<td>Username:</td>"
        "<td><input type='text' size=25 name='userid'><br></td>"
        "</tr>"
        "<br>"
        "<br>"
        "<tr>"
            "<td>Password:</td>"
            "<td><input type='Password' size=25 name='pwd'><br></td>"
            "<br>"
            "<br>"
        "</tr>"
        "<tr>"
            "<td><input type='submit' onclick='check(this.form)' value='Login'></td>"
        "</tr>"
    "</table>"
"</form>"
"<script>"
    "function check(form)"
    "{"
    "if(form.userid.value=='admin' && form.pwd.value=='admin')"
```

```
    "{"
    "window.open('/serverIndex')"
    "}"
    "else"
    "{"
    " alert('Error Password or Username')/*displays error message*/"
    "}"
    "}"
"</script>";

/*
 * Server Index Page
 */

const char* serverIndex =
"<script src='https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js'></
script>"
"<form method='POST' action='#' enctype='multipart/form-data' id='upload_form'>"
   "<input type='file' name='update'>"
        "<input type='submit' value='Update'>"
    "</form>"
 "<div id='prg'>progress: 0%</div>"
 "<script>"
  "$('form').submit(function(e){"
  "e.preventDefault();"
  "var form = $('#upload_form')[0];"
  "var data = new FormData(form);"
  " $.ajax({"
  "url: '/update',"
  "type: 'POST',"
  "data: data,"
  "contentType: false,"
  "processData:false,"
  "xhr: function() {"
  "var xhr = new window.XMLHttpRequest();"
  "xhr.upload.addEventListener('progress', function(evt) {"
  "if (evt.lengthComputable) {"
  "var per = evt.loaded / evt.total;"
  "$('#prg').html('progress: ' + Math.round(per*100) + '%');"
  "}"
  "}, false);"
  "return xhr;"
  "},"
  "success:function(d, s) {"
  "console.log('success!')"
 "},"
 "error: function (a, b, c) {"
 "}"
 "});"
 "});"
 "</script>";

/*
 * setup function
 */
void setup(void) {
  Serial.begin(115200);

  // Connect to WiFi network
  WiFi.begin(ssid, password);
  Serial.println("");

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
```

```
  Serial.println(WiFi.localIP());

  /*use mdns for host name resolution*/
  if (!MDNS.begin(host)) { //http://esp32.local
    Serial.println("Error setting up MDNS responder!");
    while (1) {
      delay(1000);
    }
  }
  Serial.println("mDNS responder started");
  /*return index page which is stored in serverIndex */
  server.on("/", HTTP_GET, []() {
    server.sendHeader("Connection", "close");
    server.send(200, "text/html", loginIndex);
  });
  server.on("/serverIndex", HTTP_GET, []() {
    server.sendHeader("Connection", "close");
    server.send(200, "text/html", serverIndex);
  });
  /*handling uploading firmware file */
  server.on("/update", HTTP_POST, []() {
    server.sendHeader("Connection", "close");
    server.send(200, "text/plain", (Update.hasError()) ? "FAIL" : "OK");
    ESP.restart();
  }, []() {
    HTTPUpload& upload = server.upload();
    if (upload.status == UPLOAD_FILE_START) {
      Serial.printf("Update: %s\n", upload.filename.c_str());
      if (!Update.begin(UPDATE_SIZE_UNKNOWN)) { //start with max available size
        Update.printError(Serial);
      }
    } else if (upload.status == UPLOAD_FILE_WRITE) {
      /* flashing firmware to ESP*/
      if (Update.write(upload.buf, upload.currentSize) != upload.currentSize) {
        Update.printError(Serial);
      }
    } else if (upload.status == UPLOAD_FILE_END) {
      if (Update.end(true)) { //true to set the size to the current progress
        Serial.printf("Update Success: %u\nRebooting...\n", upload.totalSize);
      } else {
        Update.printError(Serial);
      }
    }
  });
  server.begin();
}

void loop(void) {
  server.handleClient();
  delay(1);
}
```

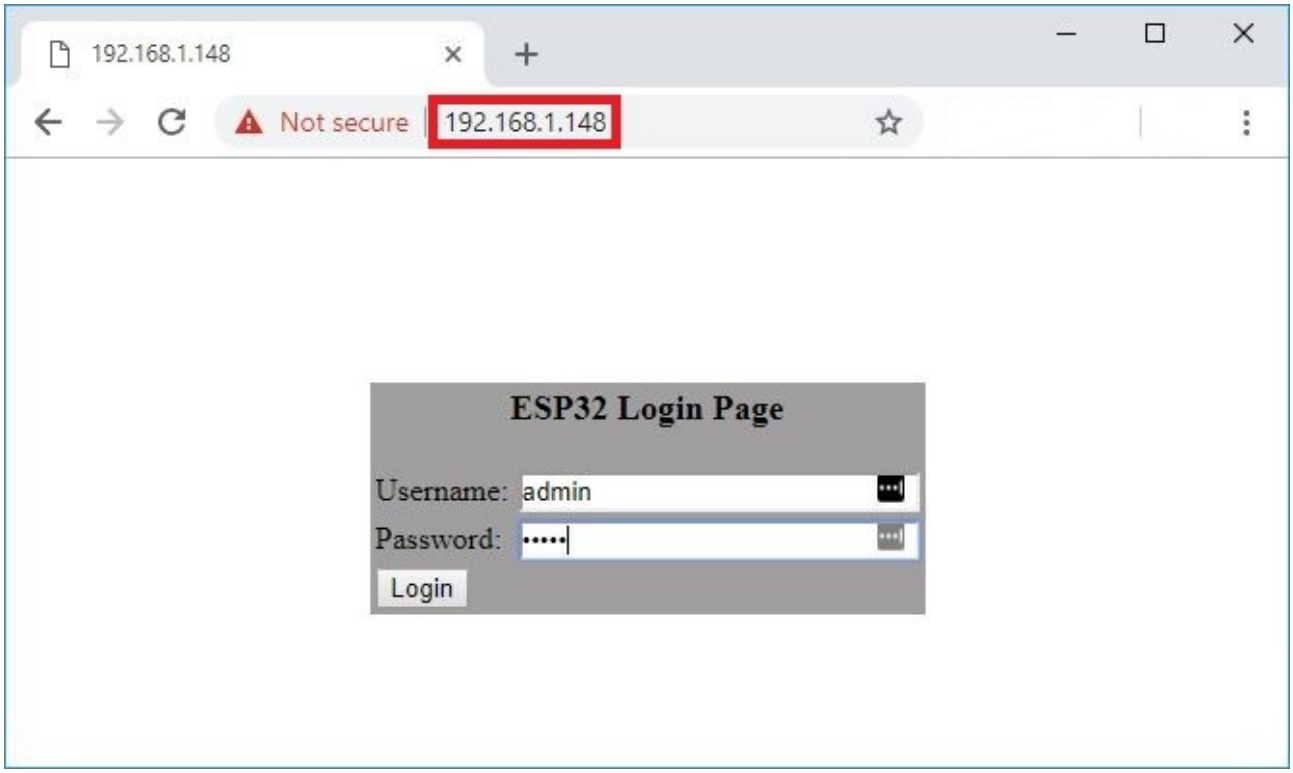
You should change the following lines on the code to include your own network credentials:

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

The OTAWebUpdater example creates an asynchronous web server where you can upload new code to your board without the need for a serial connection.
Upload the code above to your ESP32 board. Don’t forget to enter your network credentials and select the right board and serial port.
After uploading the code, open the Serial Monitor at a baud rate of 115200, press the ESP32 enable button, and you should get the ESP32 IP address:

Now, you can upload code to your ESP32 over-the-air using a browser on your local network. To test the OTA Web Updater you can disconnect the ESP32 from your computer and power it using a power bank, for example.

Update New Code using OTA Web Updater
Open a browser in your network and enter the ESP32 IP address. You should get the following:



Enter the username and the password:

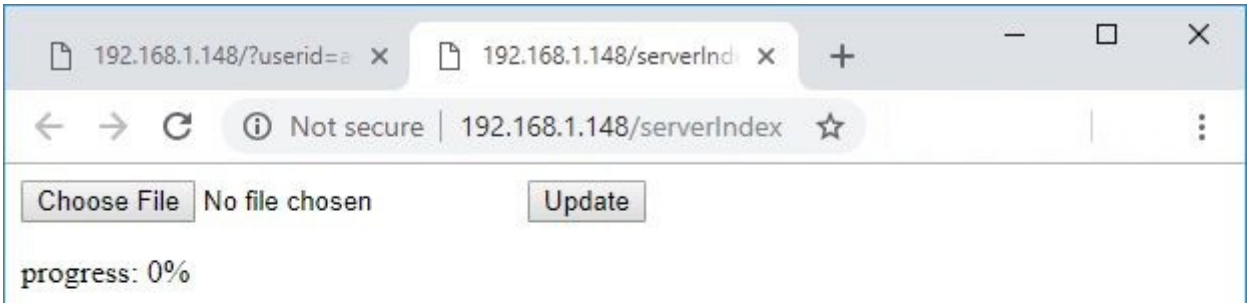
```
Username: admin
Password: admin
```

You can change the username and password on the code.

After you enter the username and password, you are redirected to the */serverIndex* URL.

Note: You don't need to enter the username and password to access the */serverIndex* URL. So, if someone knows the URL to upload new code, the username and password don't protect the web page from being accessible from others.

This page allows you to upload a new code to your ESP32. You should upload *.bin* files.



Preparing the New Sketch

When uploading a new sketch over-the-air, you need to keep in mind that you need to add code for OTA in your new sketch, so that you can always overwrite any sketch with a new one in the future. So, we recommend that you modify the OTAWebUpdater sketch to include your own code. For learning purposes let's upload a new code that blinks an LED (without delay). Copy the following code to your Arduino IDE.

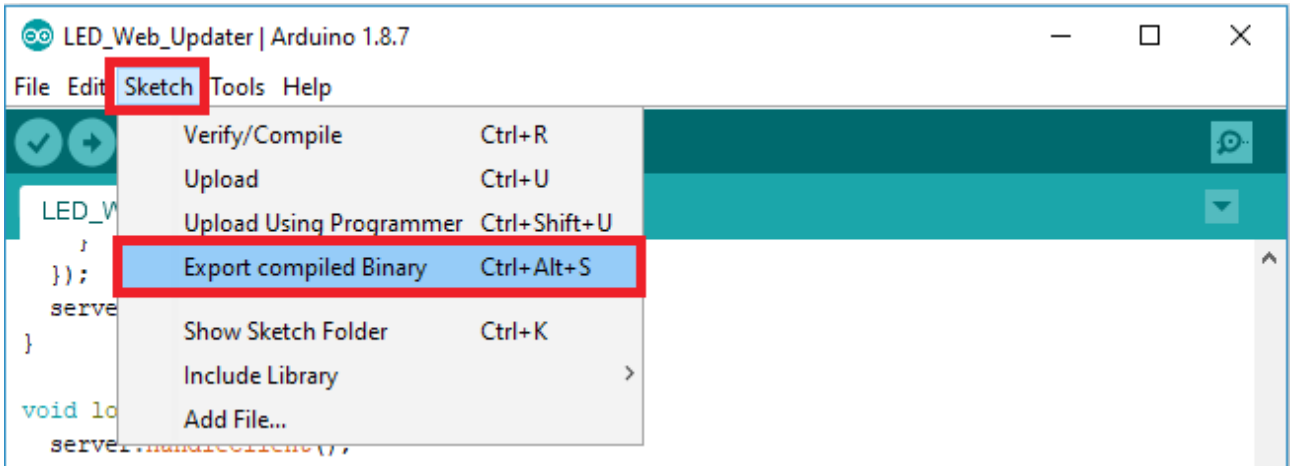
```
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include <ESPmDNS.h>
#include <Update.h>

const char* host = "esp32";
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

//variabls to blink without delay:
const int led = 2;
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

WebServer server(80);

/*
 * Login page
 */

const char* loginIndex =
 "<form name='loginForm'>"
    "<table width='20%' bgcolor='A09F9F' align='center'>"
        "<tr>"
            "<td colspan=2>"
                "<center><font size=4><b>ESP32 Login Page</b></font></center>"
                "<br>"
            "</td>"
            "<br>"
            "<br>"
        "</tr>"
        "<td>Username:</td>"
        "<td><input type='text' size=25 name='userid'><br></td>"
        "</tr>"
        "<br>"
        "<br>"
        "<tr>"
            "<td>Password:</td>"
            "<td><input type='Password' size=25 name='pwd'><br></td>"
            "<br>"
            "<br>"
        "</tr>"
        "<tr>"
            "<td><input type='submit' onclick='check(this.form)' value='Login'></td>"
        "</tr>"
    "</table>"
"</form>"
"<script>"
    "function check(form)"
    "{"
    "if(form.userid.value=='admin' && form.pwd.value=='admin')"
    "{"
    "window.open('/serverIndex')"
    "}"
    "else"
    "{"
    " alert('Error Password or Username')/*displays error message*/"
    "}"
    "}"
"</script>";
```

```
/*
 * Server Index Page
 */

const char* serverIndex =
"<script src='https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js'></
script>"
"<form method='POST' action='#' enctype='multipart/form-data' id='upload_form'>"
   "<input type='file' name='update'>"
        "<input type='submit' value='Update'>"
    "</form>"
 "<div id='prg'>progress: 0%</div>"
 "<script>"
  "$('form').submit(function(e){"
  "e.preventDefault();"
  "var form = $('#upload_form')[0];"
  "var data = new FormData(form);"
  " $.ajax({"
  "url: '/update',"
  "type: 'POST',"
  "data: data,"
  "contentType: false,"
  "processData:false,"
  "xhr: function() {"
  "var xhr = new window.XMLHttpRequest();"
  "xhr.upload.addEventListener('progress', function(evt) {"
  "if (evt.lengthComputable) {"
  "var per = evt.loaded / evt.total;"
  "$('#prg').html('progress: ' + Math.round(per*100) + '%');"
  "}"
  "}, false);"
  "return xhr;"
  "},"
  "success:function(d, s) {"
  "console.log('success!')"
 "},"
 "error: function (a, b, c) {"
 "}"
 "});"
 "});"
 "</script>";

/*
 * setup function
 */
void setup(void) {
  pinMode(led, OUTPUT);

  Serial.begin(115200);

  // Connect to WiFi network
  WiFi.begin(ssid, password);
  Serial.println("");

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  /*use mdns for host name resolution*/
  if (!MDNS.begin(host)) { //http://esp32.local
    Serial.println("Error setting up MDNS responder!");
    while (1) {
      delay(1000);
```

```
    }
  }
  Serial.println("mDNS responder started");
  /*return index page which is stored in serverIndex */
  server.on("/", HTTP_GET, []() {
    server.sendHeader("Connection", "close");
    server.send(200, "text/html", loginIndex);
  });
  server.on("/serverIndex", HTTP_GET, []() {
    server.sendHeader("Connection", "close");
    server.send(200, "text/html", serverIndex);
  });
  /*handling uploading firmware file */
  server.on("/update", HTTP_POST, []() {
    server.sendHeader("Connection", "close");
    server.send(200, "text/plain", (Update.hasError()) ? "FAIL" : "OK");
    ESP.restart();
  }, []() {
    HTTPUpload& upload = server.upload();
    if (upload.status == UPLOAD_FILE_START) {
      Serial.printf("Update: %s\n", upload.filename.c_str());
      if (!Update.begin(UPDATE_SIZE_UNKNOWN)) { //start with max available size
        Update.printError(Serial);
      }
    } else if (upload.status == UPLOAD_FILE_WRITE) {
      /* flashing firmware to ESP*/
      if (Update.write(upload.buf, upload.currentSize) != upload.currentSize) {
        Update.printError(Serial);
      }
    } else if (upload.status == UPLOAD_FILE_END) {
      if (Update.end(true)) { //true to set the size to the current progress
        Serial.printf("Update Success: %u\nRebooting...\n", upload.totalSize);
      } else {
        Update.printError(Serial);
      }
    }
  });
  server.begin();
}

void loop(void) {
  server.handleClient();
  delay(1);

  //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);
  }
}
```

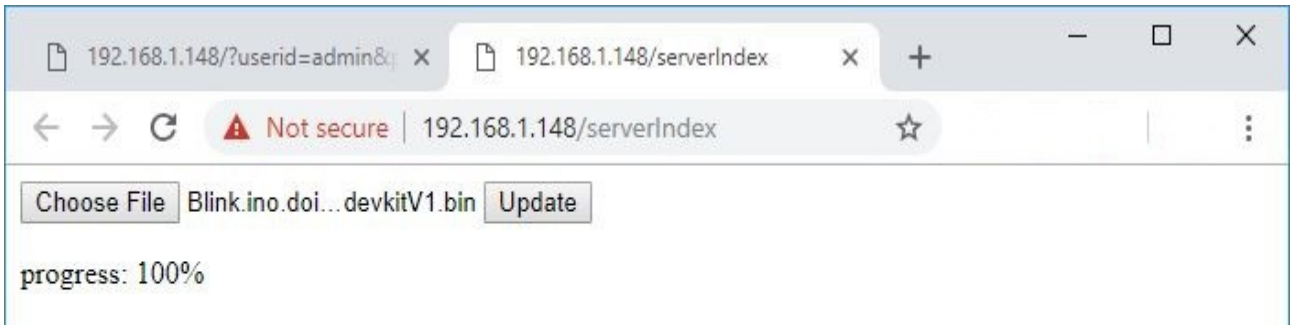
As you can see, we've added the "blink without delay" code to the OTAWebUpdater code, so that we're able to make updates later on. After copying the code to your Arduino IDE, you should generate a *.bin* file.

Generate a .bin file in Arduino IDE
Save your sketch. To generate a *.bin* file from your sketch, go to `Sketch > Export compiled Binary`



A new file on the folder sketch should be created. Go to `Sketch > Show Sketch Folder`. You should have two files in your Sketch folder: the *.ino* and the *.bin* file. You should upload the *.bin* file using the OTA Web Updater.

Upload a new sketch over-the-air to the ESP32
In your browser, on the ESP32 OTA Web Updater page, click the `Choose File` button. Select the *.bin* file generated previously, and then click `Update`. After a few seconds, the code should be successfully uploaded.



The ESP32 built-in LED should be blinking.

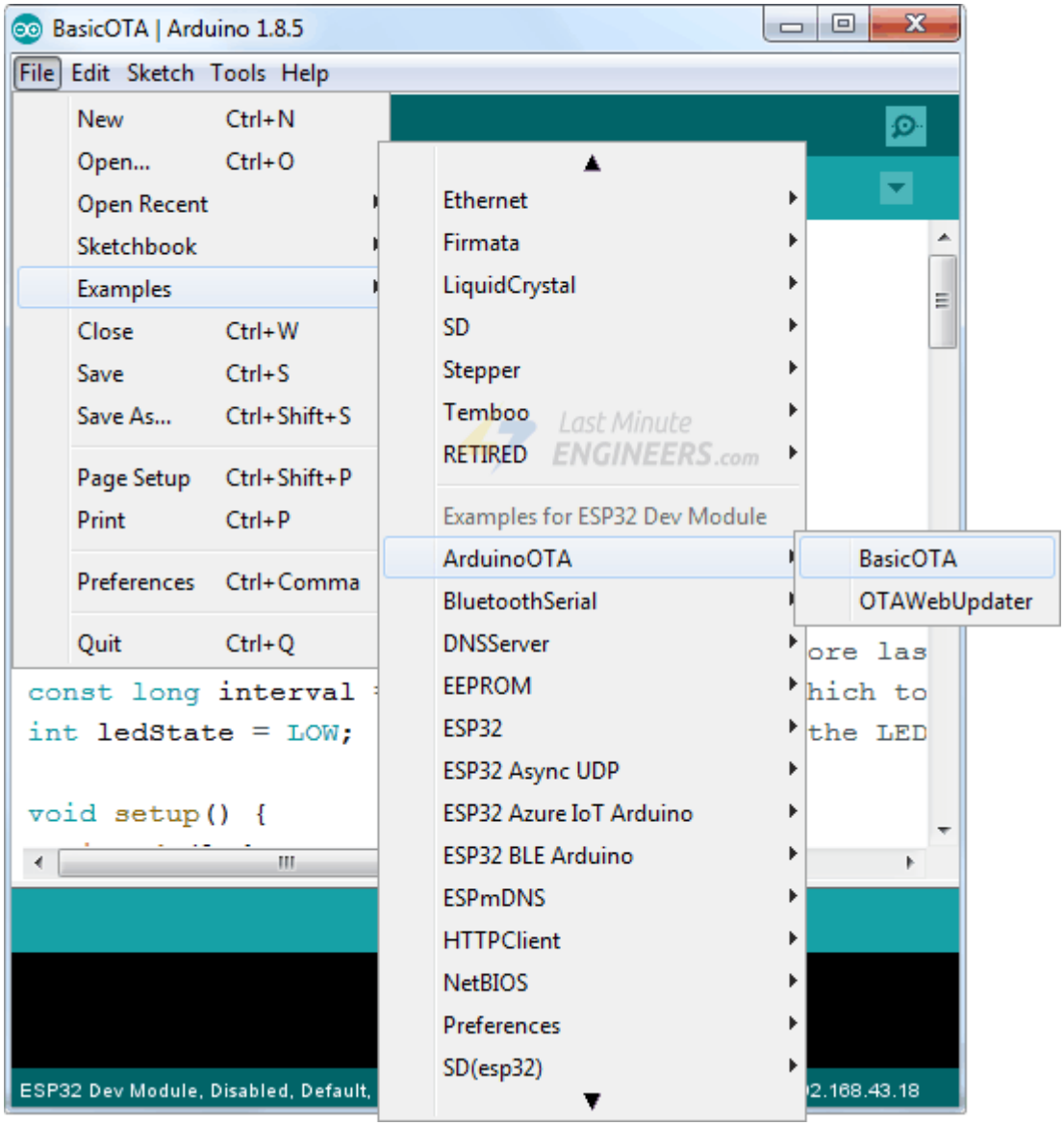
Basic OTA Programming In Arduino IDE

Steps to take:

1. Upload Basic OTA Firmware SeriallyUpload the sketch containing OTA firmware serially. It's a mandatory step, so that you're able to do the next updates/uploads over-the-air.

2. Upload New Sketch Over-The-AirNow, you can upload new sketches to the ESP32 from Arduino IDE over-the-air.

Upload OTA Routine Serially
The factory image in ESP32 doesn't have an OTA Upgrade capability. So, you need to load the OTA firmware on the ESP32 through serial interface first.
It's a mandatory step to initially update the firmware, so that you're able to do the next updates/uploads over-the-air.
The ESP32 add-on for the Arduino IDE comes with a OTA library & BasicOTA example. You can access it through `File > Examples > ArduinoOTA > BasicOTA`.



The following code should load. But, before you head for uploading the sketch, you need to make some changes to make it work for you. You need to modify the following two variables with your network credentials, so that ESP32 can establish a connection with existing network.

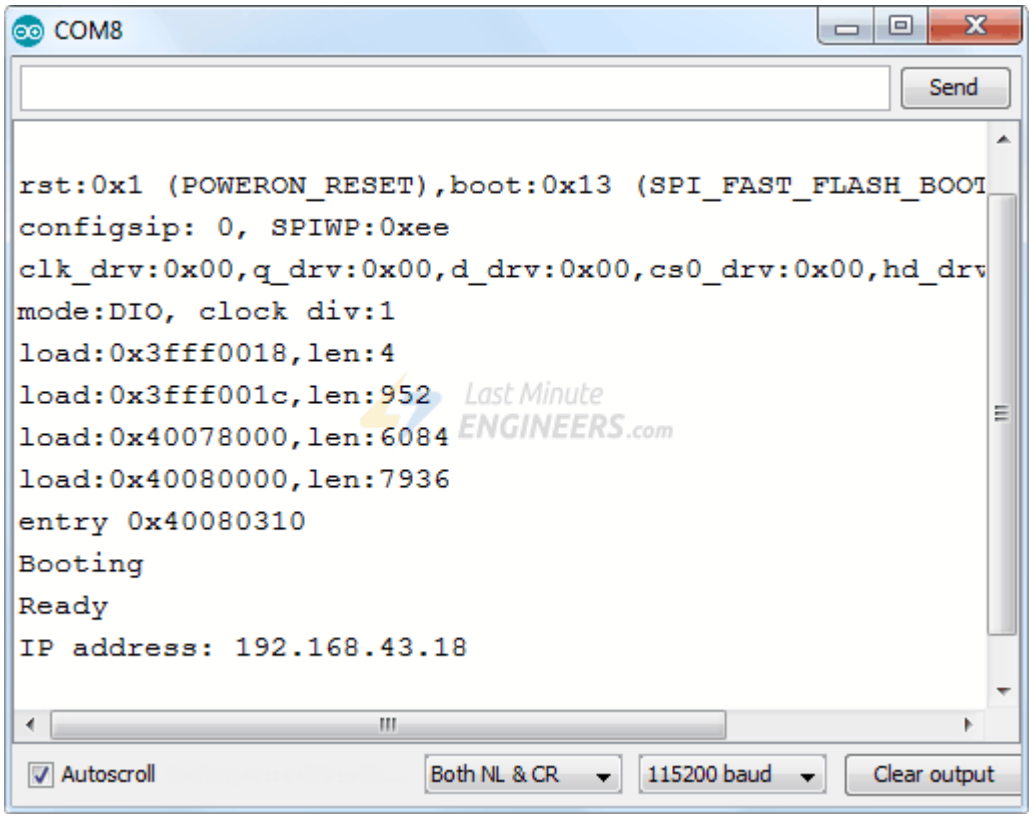
```
const char* ssid = ".........";
const char* password = ".........";
```

Once you are done, go ahead and upload the sketch.

```
#include <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 the Serial Monitor at a baud rate of 115200. And press the EN button on ESP32. If everything is OK, it will output the dynamic IP address obtained from your router. Note it down.



Upload New Sketch Over-The-Air

Now, let's upload a new sketch over-the-air.

Remember! you need to add the code for OTA in every sketch you upload. Otherwise, you'll loose OTA capability and will not be able to do next uploads over-the-air. So, it's recommended to modify the above code to include your new code.

As an example we will include a simple Blink sketch in the Basic OTA code. Remember to modify the SSID and password variables with your network credentials.

```
#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);
```

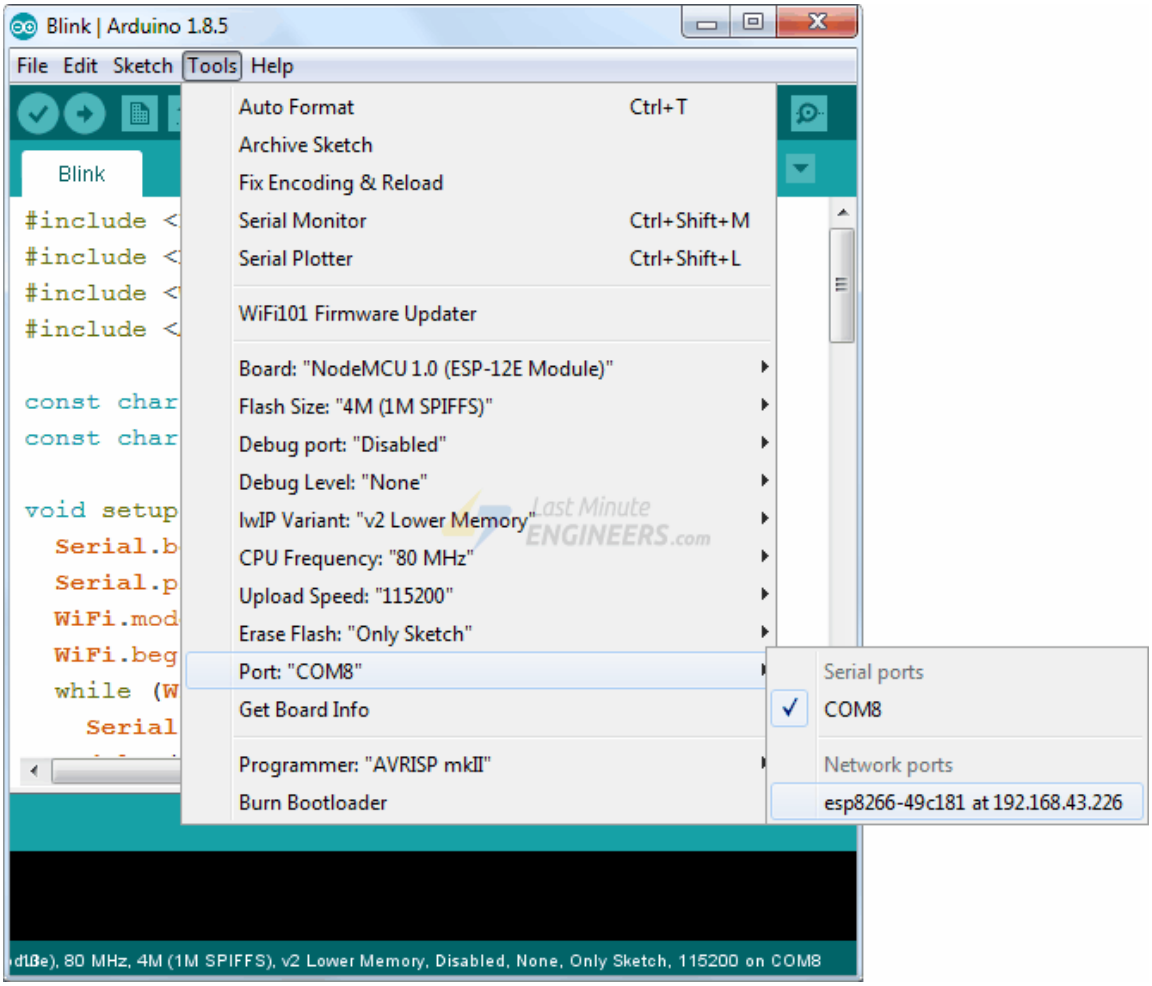
```
  while (WiFi.waitForConnectResult() != WL_CONNECTED) {
    Serial.println("Connection Failed! Rebooting...");
    delay(5000);
    ESP.restart();
  }

  // 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
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();
  //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);
  }
}
```

In above program, we have not used delay() for blinking an LED, because ESP32 pauses your program during the delay(). If next OTA request is generated while Arduino is paused waiting for the delay() to pass, your program will miss that request.

Transfer the new sketch to the ESP OTA

Option 1:

Once you copy above sketch to your Arduino IDE, go to `Tools > Port` option and you should see something like this: `esp32-xxxxxx at your_esp_ip_address`. If you can't find it, you may need to restart your IDE.

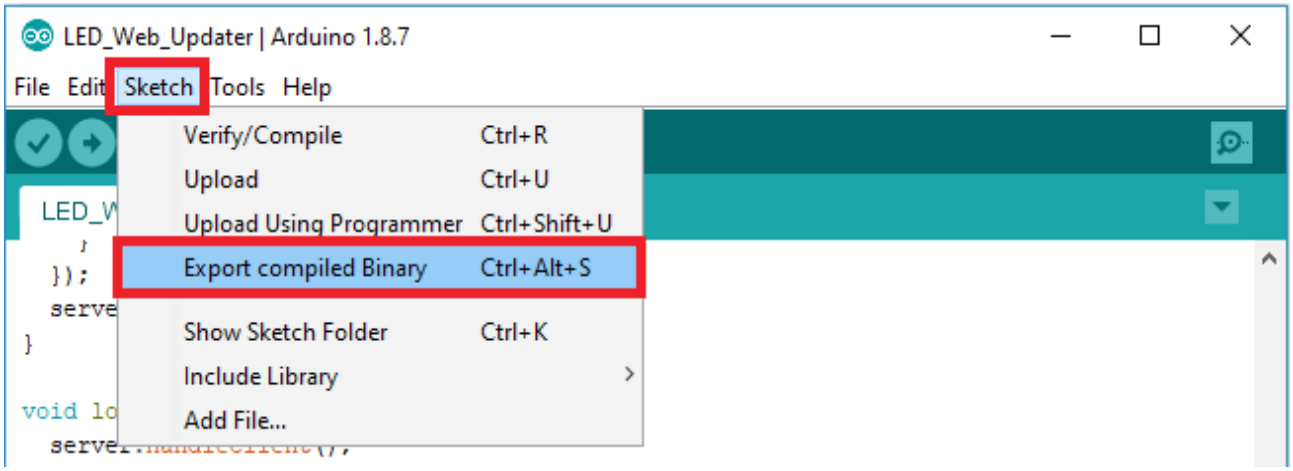


Select the port and click Upload button. Within a few seconds, the new sketch will be uploaded. And you should see the on-board LED blinking.

Option 2:

For this option you have to make sure all additional software and code is install according to the install instructions found here https://github.com/espressif/arduino-esp32.
Once this is done, you can go ahead with what is describe below.

Generate a .bin file in Arduino IDE
Save your sketch. To generate a *.bin* file from your sketch, go to `Sketch > Export compiled Binary`



A new file on the folder sketch should be created. Go to `Sketch > Show Sketch Folder`. You should have two files in your Sketch folder: the *.ino* and the *.bin* file.

To transfer the new firmware to ESP32 OTA we will use a python tool called "espota.py" or "espota.exe" in "where_you_install_Arduino/Arduino/hardware/espressif/esp32/tools"

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"

espota.exe -i ESP_IP_address -I Host_IP_address -p ESP_port -P Host_port [-a password] -f sketch.bin"

Open Terminal, command shell or Powershell in the path "where_you_install_Arduino/Arduino/hardware/espressif/esp32/tools" and type: