ESP8266 Over The Air (OTA) Programming In Arduino IDE

What is OTA programming in ESP8266?

OTA programming lets you update/upload a new program to the ESP8266 over Wi-Fi without having to connect the ESP8266 to the computer via USB.

The OTA functionality comes in handy when there is no physical access to the ESP module. In addition, it reduces the time required to update each ESP module during maintenance.

One key advantage of OTA is that a single central location can send an update to multiple ESPs on the same network.

The only disadvantage is that you must include an OTA code with each sketch you upload in order to use OTA in the next update.

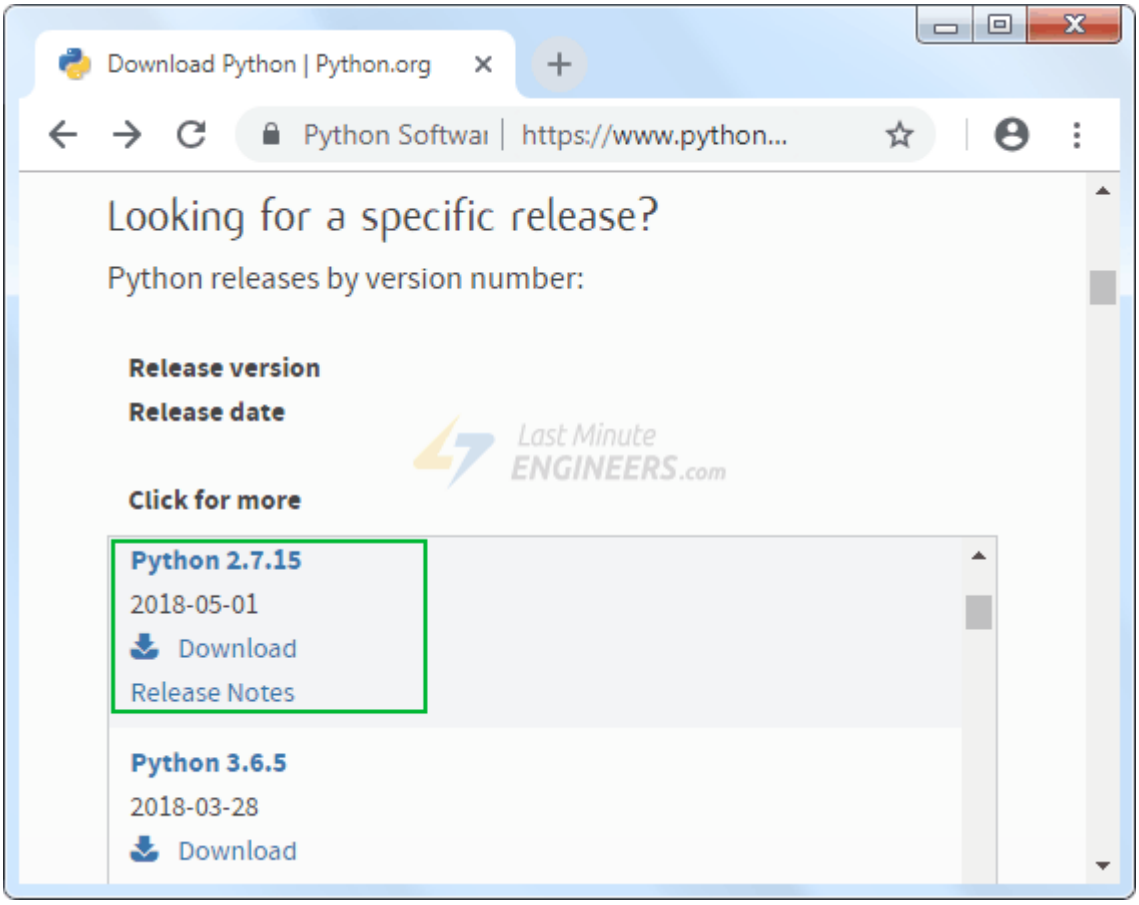
3 Simple Steps for Using OTA with the ESP8266

1. Installing Python 2.7.x series: The first step is to install the Python 2.7.x series on your computer.
2. Uploading Basic OTA Firmware Serially: Upload the sketch containing the OTA firmware serially. This is a required step in order to perform the subsequent updates over-the-air.
3. Uploading New Sketch Over-The-Air: You can now upload new sketches to the ESP8266 from the Arduino IDE over-the-air.

Step 1: Install Python 2.7.x series

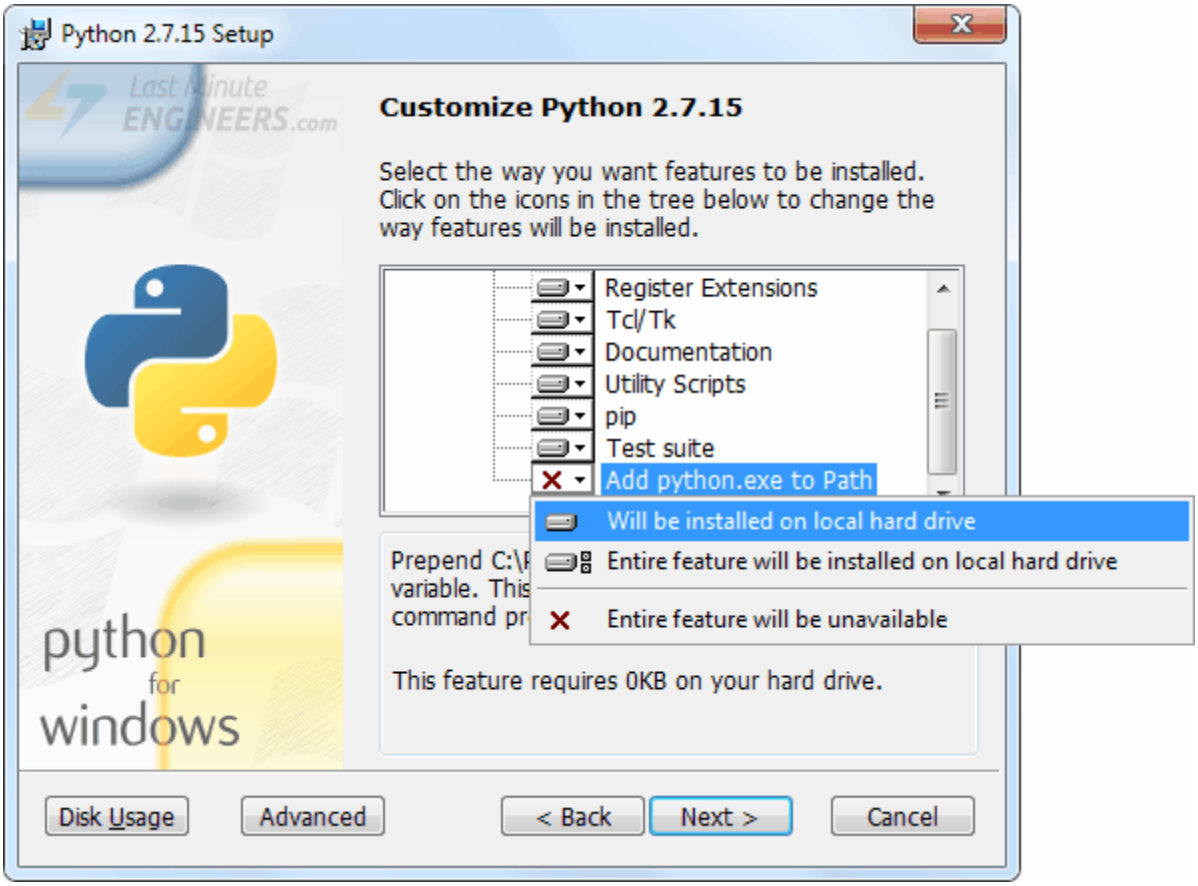
To use OTA functionality, you must first install Python 2.7.x, if it is not already installed on your machine.

Download Python 2.7.x for Windows (MSI installer) from the official Python website.



Launch the installer and proceed through the installation wizard.

Make sure the “Add python.exe to Path” option is enabled in the Customize Python 2.7.X section.

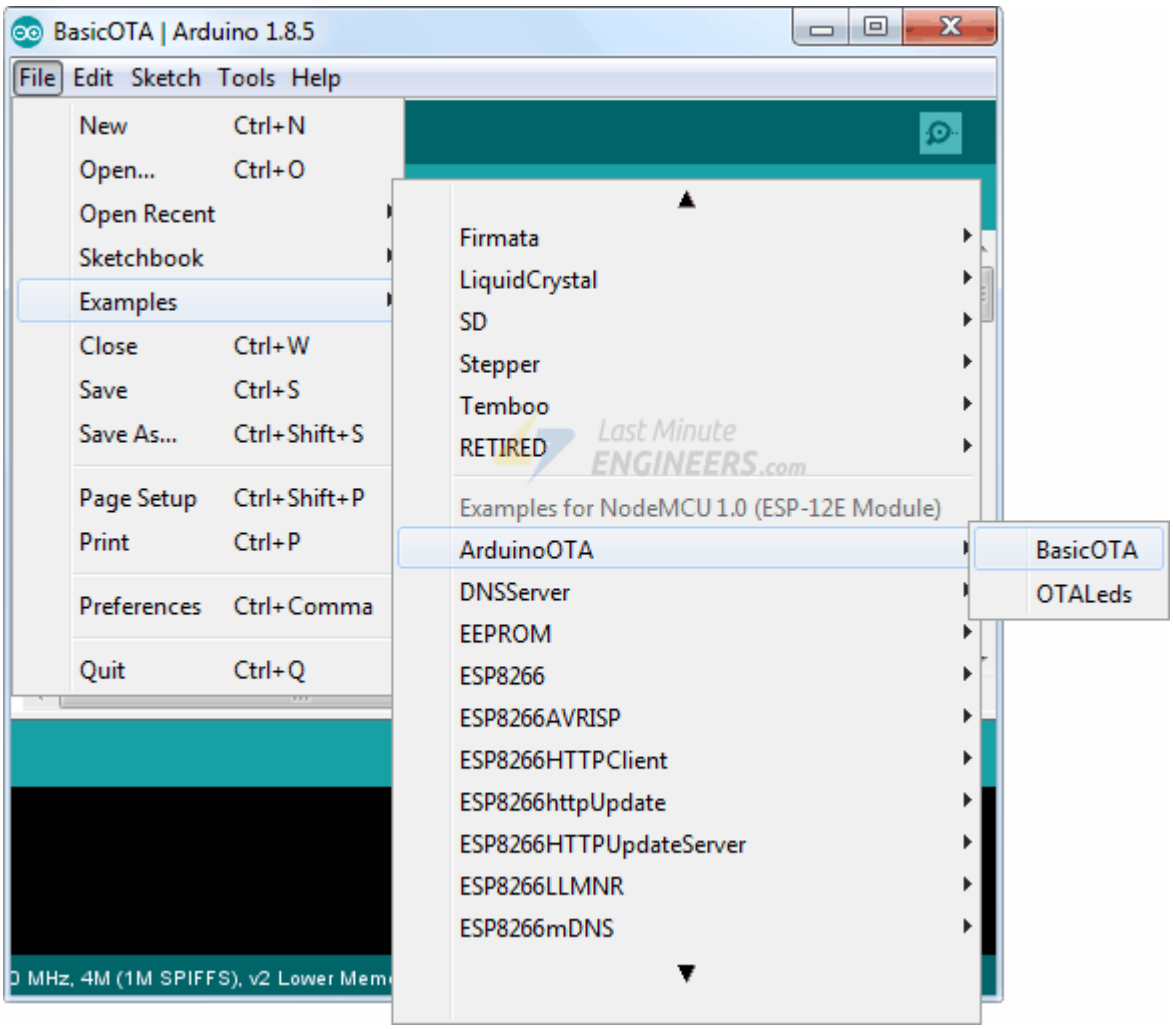


Step 2: Upload Basic OTA Firmware Serially

Because the ESP8266 's factory image lacks OTA Upgrade capability, you must first load the OTA firmware on the ESP8266 via serial interface.

It is required to first update the firmware in order to perform subsequent updates over-the-air.

The ESP8266 add-on for the Arduino IDE includes an OTA library as well as a BasicOTA example. Simply navigate to File > Examples > ArduinoOTA > BasicOTA.



Before you begin uploading the sketch, you must modify the following two variables with your network credentials so that the ESP8266 can connect to an existing network.

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

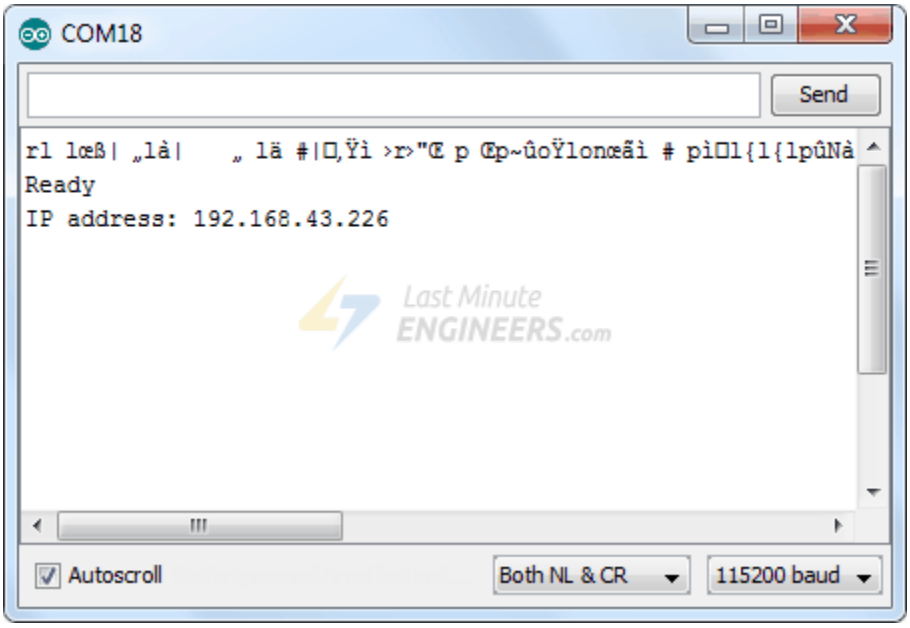
When you're finished, go ahead and upload the sketch.

```
#include <ESP8266WiFi.h>
#include <ESP8266mDNS.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 8266
  // ArduinoOTA.setPort(8266);

  // Hostname defaults to esp8266-[ChipID]
  // ArduinoOTA.setHostname("myesp8266");

  // 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);
  });
  ArduinoOTA.onEnd([]() {
```

```
    Serial.println("\nEnd");
  });
  ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {
    Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
  });
  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");
  });
  ArduinoOTA.begin();
  Serial.println("Ready");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}

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

Now, open the Serial Monitor at 115200 baud rate and press the RST button on the ESP8266. If everything is fine, you should see the dynamic IP address assigned by your router. Make a note of it.



Step 3: Upload New Sketch Over-The-Air

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

Remember that you must include the OTA code in every sketch you upload. Otherwise, you will lose OTA capability and will be unable to perform the next over-the-air upload. Therefore, it is recommended that you modify the preceding 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.

Changes to the code are highlighted in blue.

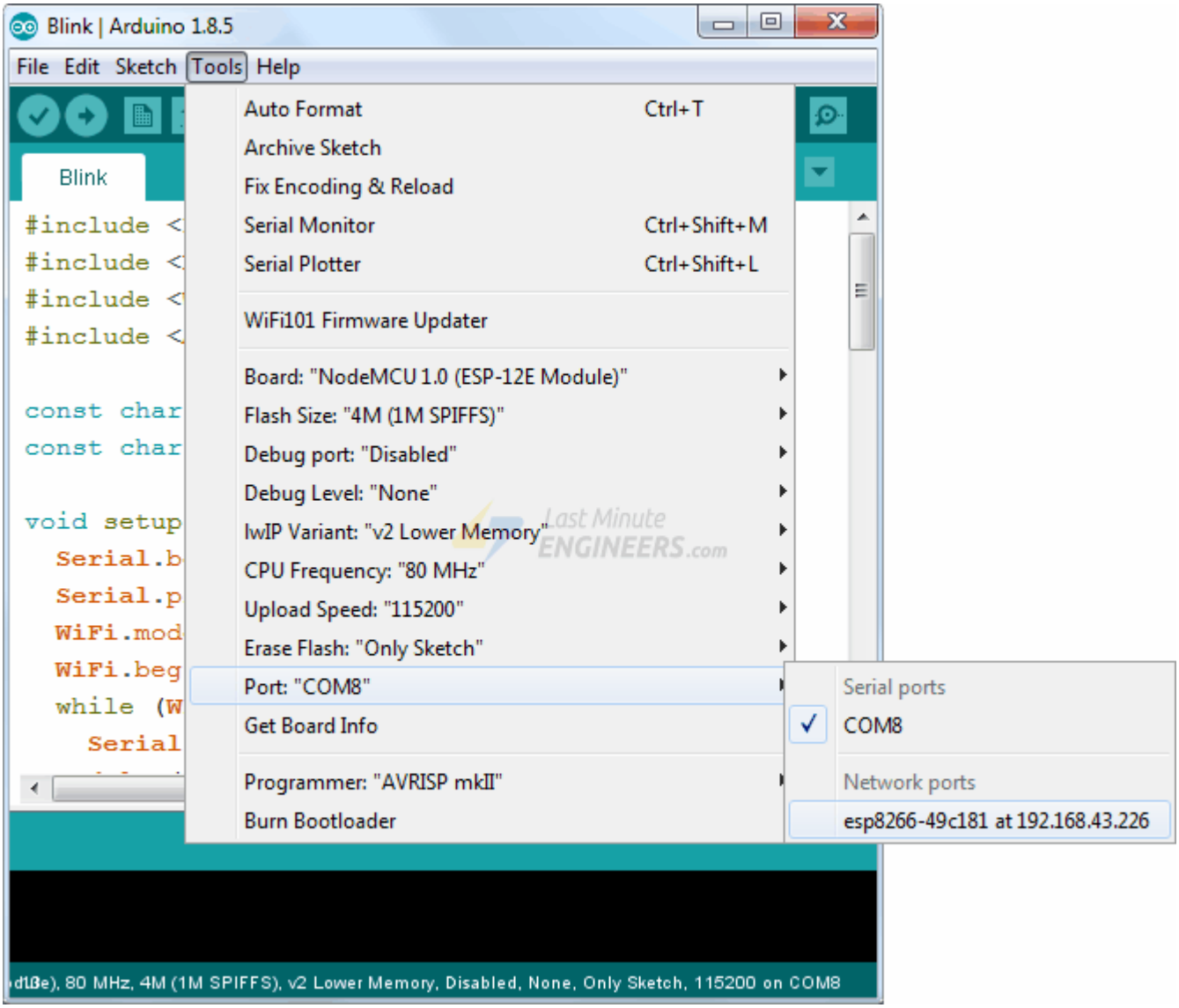
```
#include <ESP8266WiFi.h>
#include <ESP8266mDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>

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

//variabls for blinking an LED with Millis
const int led = D0; // ESP8266 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 8266
   // ArduinoOTA.setPort(8266);

   // Hostname defaults to esp8266-[ChipID]
   // ArduinoOTA.setHostname("myesp8266");

   // 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);
  });
  ArduinoOTA.onEnd([]() {
    Serial.println("\nEnd");
  });
  ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {
    Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
  });
  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");
  });
  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);
    }
}
```

Notice that we haven't used the `delay()` function to make the LED blink. This is because the `delay()` function pauses the program. If the next OTA request is generated while the ESP8266 is paused waiting for the `delay()` to complete, your program will miss that request.

After copying the above sketch to your Arduino IDE, navigate to Tools > Port option. Look for something like: esp8266-xxxxxx at your_esp_ip_address. If you are unable to locate it, you may need to restart your IDE.



Choose the port and press the Upload button. The new sketch will be uploaded in a matter of seconds. The on-board LED should start blinking.