```
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include <LEAmDNS.h>

#ifndef STASSID
#define STASSID "your-ssid"
#define STAPSK "your-password"
#endif

const char* ssid = STASSID;
const char* password = STAPSK;

WebServer server(80);

const int led = LED_BUILTIN;

void handleRoot() {
  digitalWrite(led, 1);
  server.send(200, "text/plain", "hello from pico w!\r\n");
  digitalWrite(led, 0);
}

void handleNotFound() {
  digitalWrite(led, 1);
  String message = "File Not Found\n\n";
  message += "URI: ";
  message += server.uri();
  message += "\nMethod: ";
  message += (server.method() == HTTP_GET) ? "GET" : "POST";
  message += "\nArguments: ";
  message += server.args();
  message += "\n";
  for (uint8_t i = 0; i < server.args(); i++) {
    message += " " + server.argName(i) + ": " + server.arg(i) + "\n";
  }
  server.send(404, "text/plain", message);
  digitalWrite(led, 0);
}

void setup(void) {
  pinMode(led, OUTPUT);
  digitalWrite(led, 0);
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  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());

  if (MDNS.begin("picow")) {
    Serial.println("MDNS responder started");
  }

  server.on("/", handleRoot);

  server.on("/inline", []() {
    server.send(200, "text/plain", "this works as well");
  });

  server.on("/gif", []() {
    static const uint8_t gif[] = {
      0x47, 0x49, 0x46, 0x38, 0x37, 0x61, 0x10, 0x00, 0x10, 0x00, 0x80, 0x01,
      0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0x2c, 0x00, 0x00, 0x00, 0x00,
      0x10, 0x00, 0x10, 0x00, 0x00, 0x02, 0x19, 0x8c, 0x8f, 0xa9, 0xcb, 0x9d,
      0x00, 0x5f, 0x74, 0xb4, 0x56, 0xb0, 0xb0, 0xd2, 0xf2, 0x35, 0x1e, 0x4c,
      0x0c, 0x24, 0x5a, 0xe6, 0x89, 0xa6, 0x4d, 0x01, 0x00, 0x3b
    };
    char gif_colored[sizeof(gif)];
    memcpy_P(gif_colored, gif, sizeof(gif));
    // Set the background to a random set of colors
    gif_colored[16] = millis() % 256;
    gif_colored[17] = millis() % 256;
    gif_colored[18] = millis() % 256;
    server.send(200, "image/gif", gif_colored, sizeof(gif_colored));
  });

  server.onNotFound(handleNotFound);

  /////////////////////////////////////////////////////
  // Hook examples

  server.addHook([](const String & method, const String & url, WiFiClient *
client, WebServer::ContentTypeFunction contentType) {
    (void)method;       // GET, PUT, ...
    (void)url;          // example: /root/myfile.html
    (void)client;       // the webserver tcp client connection
    (void)contentType;  // contentType(".html") => "text/html"
    Serial.printf("A useless web hook has passed\n");
    return WebServer::CLIENT_REQUEST_CAN_CONTINUE;
  });

  server.addHook([](const String&, const String & url, WiFiClient*,
WebServer::ContentTypeFunction) {
    if (url.startsWith("/fail")) {
```

```
      Serial.printf("An always failing web hook has been triggered\n");
      return WebServer::CLIENT_MUST_STOP;
    }
    return WebServer::CLIENT_REQUEST_CAN_CONTINUE;
  });

  server.addHook([](const String&, const String & url, WiFiClient * client,
WebServer::ContentTypeFunction) {
    if (url.startsWith("/dump")) {
      Serial.printf("The dumper web hook is on the run\n");

      // Here the request is not interpreted, so we cannot for sure
      // swallow the exact amount matching the full request+content,
      // hence the tcp connection cannot be handled anymore by the
      auto last = millis();
      while ((millis() - last) < 500) {
        char buf[32];
        size_t len = client->read((uint8_t*)buf, sizeof(buf));
        if (len > 0) {
          Serial.printf("(<%d> chars)", (int)len);
          Serial.write(buf, len);
          last = millis();
        }
      }
      // Two choices: return MUST STOP and webserver will close it
      //                       (we already have the example with '/fail' hook)
      // or                  IS GIVEN and webserver will forget it
      // trying with IS GIVEN and storing it on a dumb WiFiClient.
      // check the client connection: it should not immediately be closed
      // (make another '/dump' one to close the first)
      Serial.printf("\nTelling server to forget this connection\n");
      static WiFiClient forgetme = *client;  // stop previous one if present and
transfer client refcounter
      return WebServer::CLIENT_IS_GIVEN;
    }
    return WebServer::CLIENT_REQUEST_CAN_CONTINUE;
  });

  // Hook examples
  ///////////////////////////////////////////////////////

  server.begin();
  Serial.println("HTTP server started");
}

void loop(void) {
  server.handleClient();
  MDNS.update();
}
```