 GET and POST are methods of communicating between web browser and the server.

* **GET** – Requests data from a specified resource
* **POST** – Submits data to be processed to a specified resource

The Hypertext Transfer Protocol (**HTTP**) is designed to enable communications between clients and servers.

HTTP works as a request-response protocol between a client and server. Each Hypertext Transfer Protocol (HTTP) message is either a request or a response. A server listens on a connection for a request, parses each message received, interprets the message semantics in relation to the identified request target, and responds to that request with one or more response messages. A client constructs request messages to communicate specific intentions, examines received responses to see if the intentions were carried out, and determines how to interpret the results.

First of all, we need to declare an object of the class [HTTPClient](https://github.com/esp8266/Arduino/blob/dd81336b79ddf15925876b983af13816d9d5807e/libraries/ESP8266HTTPClient/src/ESP8266HTTPClient.cpp), from which we will call various methods to prepare the headers and content of the request, send it and check for the result. We will call this object simply “http”.

|  |  |
| --- | --- |
| 1 | HTTPClient http; |

After that, we call the begin method on the http object and pass the URL that we want to connect to and make the post request. In this case, I’m sending the post request to an application running on my local network, which is why I’m sending it the format seen bellow (Host IP:Port/Path).

|  |  |
| --- | --- |
| 1 | http.begin("<http://192.168.1.88:9999/hello>"); |

Nevertheless, we can send the request to a website by specifying it’s domain name, as seen bellow (the destination [website](http://jsonplaceholder.typicode.com/) specified implements a dummy REST API for testing and prototyping).

|  |  |
| --- | --- |
| 1 | http.begin("<http://jsonplaceholder.typicode.com/users>"); |

Next, we can define headers with the addHeader method. In this case, we are specifying the content-type as “text/plain”, since we will just send a simple string in the body.

|  |  |
| --- | --- |
| 1 | http.addHeader("Content-Type", "text/plain"); |

The body of the request is specified as a parameter when calling the POST method on the HTTPClient object. In this case, we will simply send a string saying “Message from ESP8266”. The return value of this method corresponds to the HTTP response code and thus is important to check for error handling.

|  |  |
| --- | --- |
| 1 | int httpCode = http.POST("Message from ESP8266"); |

We can now get the payload by calling the getString method, which will return the response payload as a string.

|  |  |
| --- | --- |
| 1 | String payload = http.getString(); |

In the end, we need to call the end() method on the object to guarantee that the TCP connection is closed. This is very important to free the resources.

|  |  |
| --- | --- |
| 1 | http.end(); |

Just to handle any possible WiFi connection errors, we will include a validation of the connection status before making the request. For debugging purposes, we will print both the response payload and the HTTP code.

**GetRequest in ESP8266**

#include <ESP8266WiFi.h>

#include <ESP8266HTTPClient.h>

const char\* ssid = "Linkedthings709"; // The SSID (name) of the Wi-Fi network you want to connect to

const char\* password = "F4DD396D"; // The password of the Wi-Fi network

HTTPClient http;

void setup()

{

Serial.begin(115200);

Serial.println();

Serial.println(WiFi.localIP());

Serial.printf("Connecting to %s ", ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED)

{

delay(500);

Serial.print(".");

}

Serial.println(" connected");

}

void loop()

{

WiFiClient client;

http.begin("http://ct.ottomatically.com/api/v1/events/phase/1"); //Specify request destination

int httpCode = http.GET(); //Send the request

if (httpCode > 0) { //Check the returning code

String payload = http.getString(); //Get the request response payload

Serial.println(payload); //Print the response payload

}

http.end(); //Close connection

delay(5000);

}

**postRequest in ESP8266**

#include <ESP8266WiFi.h>

#include <ESP8266HTTPClient.h>

const char\* ssid = "Linkedthings709"; // The SSID (name) of the Wi-Fi network you want to connect to

const char\* password = "F4DD396D"; // The password of the Wi-Fi network

HTTPClient http;

void setup()

{

Serial.begin(115200);

Serial.println();

Serial.println(WiFi.localIP());

Serial.printf("Connecting to %s ", ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED)

{

delay(500);

Serial.print(".");

}

Serial.println(" connected");

}

void loop()

{

WiFiClient client;

HTTPClient http; //Declare object of class HTTPClient

http.begin("http://ct.ottomatically.com/api/v1/events"); //Specify request destination

http.addHeader("Content-Type", "application/json"); //Specify content-type header

String val = "\"2 1 Faiq\"";

int httpCode = http.POST(val); //Send the request

String payload = http.getString(); //Get the response payload

Serial.println(httpCode); //Print HTTP return code

Serial.println();

Serial.println(payload); //Print request response payload

http.writeToStream(&Serial);

http.end(); //Close connection

delay(5000);

}