Below I show the code relative to the first example, commented in order to understand the purpose of each part.

Note: When I refer to “rise ups”, I mean that when the pulse from the flowmeter goes from LOW to HIGH, each “rise up” it's accounted and stored in a variable.

There is some libraries not so common that I describe below:
→ **EEPROM.h** allows the use of read/write functions of the EEPROM in the NodeMcu.
→ **ESP8266Wifi.h** gives the possibility of using the wifi functionality (from the ESP8266 module that comes embebbed in the NodeMcu). With this library it's possible to connect through wifi to the local network, it's very usefull in a project where the main objectiv is to have a sensor connected to the network, and remotely accessing it.
→ **ESP8266HTTPCLIENT.H** open the door for the possibility of making requests to the HTTP servers.

```
#include <Arduino.h>
#include <EEPROM.h>
#define USE_SERIAL Serial
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>

  // Variable init
  const int  buttonPin = D2; // variable for D2 pin
  int contagem = 0;   // variable to store the “rise ups” from the flowmeter pulses
  int litros = 0;
  char thingspeak_string[200]; //string used to send info to the server ThingSpeak
  char litros_string[10]="0";
  int addr = 0; //endereço eeprom

  //SSID and PASSWORD for the AP (swap the XXXXX for real ssid and password )
  const char* ssid = "XXXXX";
  const char* password = "XXXXX";

  //HTTP client init
  HTTPClient http;

  //Webserver init
  WiFiServer server(80);

   //Interrupt function, so that the counting of pulse “rise ups” dont interfere with the rest of the code (attachInterrupt)
  void pin_ISR()
  {
     contagem++;
  }

void setup() {
   // Serial Comunication init
   Serial.begin(115200);
   delay(10);

   // EEPROM access init
   EEPROM.begin(1);
   litros=EEPROM.read(addr);
   // Initialization of the variable “buttonPin” as INPUT (D2 pin)
   pinMode(buttonPin, INPUT);

   // Wifi connection init
   Serial.println();
   Serial.print("A iniciar ligação...");
   Serial.println();
   WiFi.begin(ssid, password);

  //Waiting for the connection to be established
   Serial.print("Waiting for the connection...");
   while (WiFi.status() != WL_CONNECTED)
    {
      delay(2000);
      Serial.print(".");

      if(WiFi.status() == WL_CONNECTED)
      {
        Serial.println();
        Serial.printf("Connect to the SSID: %s",ssid);
      }
    }

    /**********************/
    // Starting Webserver
    server.begin();
    Serial.println();
    Serial.println();
    Serial.println("Server started");

    // Print the IP address
    Serial.print("Use this URL to connect: ");
    Serial.print("http://");
    Serial.print(WiFi.localIP());
    Serial.println("/");
    Serial.println();
    Serial.print("A iniciar contagem dos litros...");

  // Attach an interrupt to the ISR vector
  attachInterrupt(digitalPinToInterrupt(buttonPin), pin_ISR, RISING);

  Serial.println();
  Serial.print("Waiting for client....");
  Serial.println();
}
void loop() {

  // Verify if a clients is connected to the server
  WiFiClient client = server.available();
```

```
 // Reply from the local http server, and constrution of the page “on the fly”
    client.println("HTTP/1.1 200 OK");
    client.println("Content-Type: text/html");
    client.println("Connection: keep-alive");
    client.println(""); //  do not forget this one
    client.println("<!DOCTYPE HTML>");
    client.println("<html>");
    client.println("<head><meta http-equiv=\"refresh\" content=\"10\" >");
    client.println("<script type='text/javascript'>");
    client.println("function loadDoc() {");
    client.println("var xhttp = new XMLHttpRequest();");
    client.println("xhttp.onreadystatechange = function() {");
    client.println("if (xhttp.readyState == 4 && xhttp.status == 200) {");
    client.printf("document.getElementById('id_litros').innerHTML = %d",litros);
    client.println("}");
    client.println("};");
    client.println("xhttp.open('GET', '', true);");
    client.println("xhttp.send();");
    client.println("}");
    client.println("</script></head>");
    client.println("<body onload='setInterval(loadDoc, 5000);'>");
    client.println("<br/><br/>");
    client.printf("<div id='id_litros'>Est&atilde;o contados %d litros!</div>",litros);
            client.println("<iframe width=\"450\" height=\"260\" style=\"border: 1px solid #cccccc;\"
src=\"https://thingspeak.com/channels/120470/charts/1?bgcolor=%23ffffff&color=
%23d62020&dynamic=true&results=60&title=Contagem+de+Litros&type=line\"></iframe>");
    client.println("</body>");
    client.println("</html>");
    client.stop();

    delay(1);


 // If the counting of transitions (Low to High, “rise ups”) it's higher than 440, count one litre more. Then do
the rest of the functions (update to EEPROM variable, loca  webserver and ThingSpeak)
 //pulse per litre +/- 450 "www.hobbytronics.co.uk/yf-s201-water-flow-meter"

 if(contagem > 440 )
 {
    litros++;
    Serial.println();
    Serial.print("Litros: ");
    Serial.print(litros);

    //Write the new litres value to the EEPROM and put “contagem” variable to zero
    EEPROM.write(addr, litros);
    EEPROM.commit();
    contagem = 0;
```

```
      //The value of litres is sent to the ThingSpeak server. It is needed to have an account in ThingSpeak
server before using this funcionality. You will have to copy the link given (something like this:
https://api.thingspeak.com/update.json?api_key=XXX&field1=XXX), example below.
     dtostrf(litros, 4, 2, litros_string);
    sprintf(thingspeak_string,"https://api.thingspeak.com/update.json?api_key=UI9DXIOZPFW2NCST&field1=
%s", litros_string);
      //String sent to ThingSpeak server.
      http.begin(thingspeak_string);

     //Send HTTP Header
      int httpCode = http.GET();

      // httpCode_code will be a negative number if there is an error
      if(httpCode > 0) {
          // file found at server
          if(httpCode == HTTP_CODE_OK) {
              String payload = http.getString();
              Serial.print(" ");
              Serial.println(payload);
          }
      } else {
          Serial.printf("[HTTP] GET... failed, error: %s\n", http.errorToString(httpCode).c_str());
      }
      http.end();

  }//stop counting

  delay(500);
}
```