logistiek systeem
Machine

1. Doel

Stroomlijnen van de ophaling en het versturen van stalen aan de hand van led lichten.

2. Principe

Als er met led lichtjes buiten het labo pathologie aangegeven kan worden of er stalen binnen opgehaald moeten worden, voorkomt dit veel zinloos verloop voor de logistieke dienst en de koeriers van het centraal labo.

3. Toepassingsgebied

Labo pathologie en de gang net buiten het labo pathologie.

4. Reagentia en materialen

Nvt

5. Apparatuur

Het systeem maakt gebruik van ESP 32 computers, leds, breadboards, push buttons, transistors en bedrading voor het maken van circuits. Daarnaast moet het systeem toegang hebben tot wifi.

The following list shows a summary of the ESP32 DEVKIT V1 DOIT board features and specifications:
Number of cores 2 (dual core)
Wi-Fi 2.4 GHz up to 150 Mbits/s
Bluetooth BLE (Bluetooth Low Energy) and legacy Bluetooth
Architecture 32 bits
Clock frequency Up to 240 MHz
RAM 512 KB
Pins 30
Peripherals
Capacitive touch, ADC (analog to digital converter), DAC (digital to analog converter), I2C (Inter-Integrated Circuit), UART (universal asynchronous receiver/transmitter), CAN 2.0 (Controller Area Netwokr), SPI (Serial Peripheral Interface), I2S (Integrated Inter-IC

Sound), RMII (Reduced
Media-Independent Interface), PWM
(pulse width modulation), and more.
Built-in buttons RESET and BOOT buttons
Built-in LEDs
built-in blue LED connected to GPIO2;
built-in red LED that shows the board is being powered
USB to UART
bridge CP2102

It comes with a microUSB interface that you can use to connect the board to your computer to upload code or apply power.

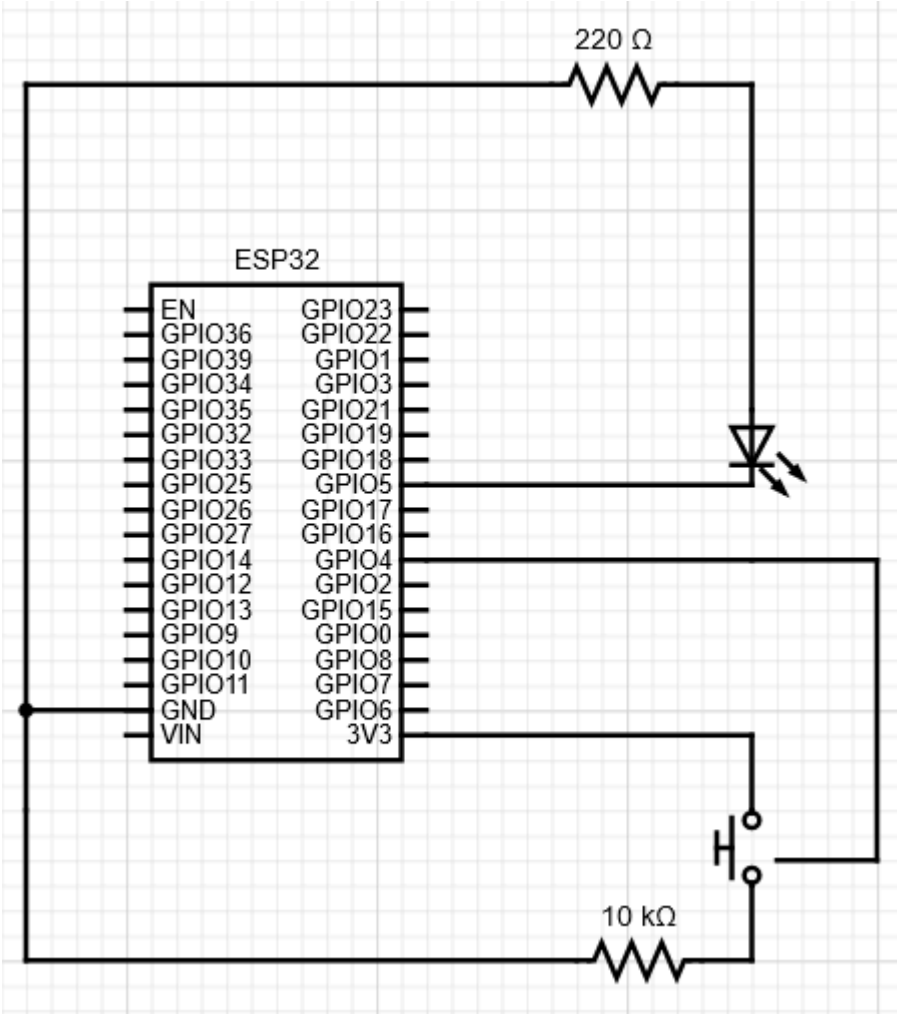
It uses the CP2102 chip (USB to UART) to communicate with your computer via a COM port using a serial interface. Another popular chip is the CH340. Check what's the USB to UART chip converter on your board because you'll need to install the required drivers so that your computer can communicate with the board (more information about this later in this guide).

This board also comes with a RESET button (may be labeled EN) to restart the board and a BOOT button to put the board in flashing mode (available to receive code).

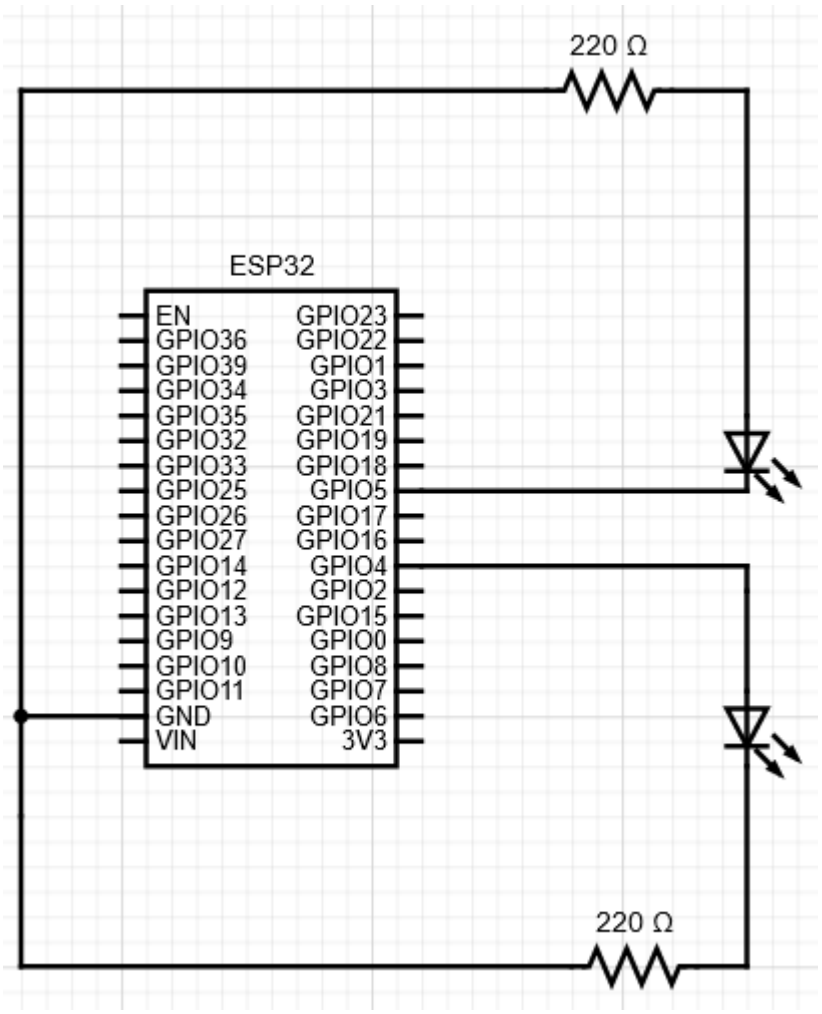
It also comes with a built-in blue LED that is internally connected to GPIO 2. This LED is useful for debugging to give some sort of visual physical output. There's also a red LED that lights up when you provide power to the board.

ESP32 DEVKIT V1
5 mm LED
220 Ohm resistor
Pushbutton
10k Ohm resistor
Breadboard
Jumper wires

Logistics Log/Logistics Lab:



Logistics Hallway:

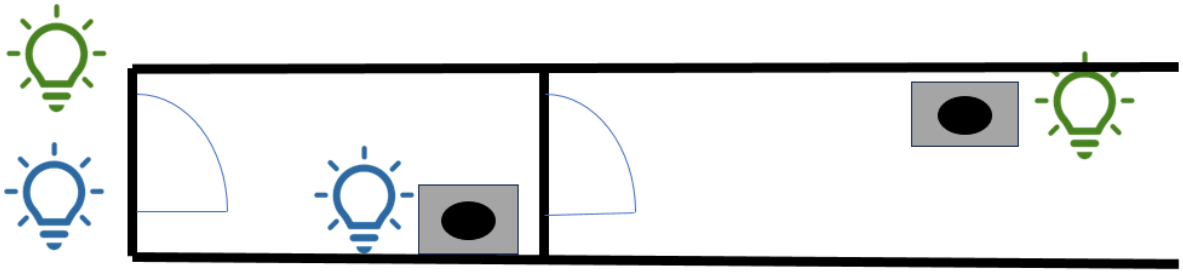


6. Veiligheid

Er mag geen brandbaar materiaal dicht bij de elektronica geplaatst worden. Koppel het systeem los van het net voordat er aan het circuit gewerkt wordt!

7. Werkwijze

7.1. Schema



7.2. Werking

Als de logistieke dienst een item moet komen ophalen, moet de ledlamp in de grote SAS branden. Dit doe je door 1 maal op de knop te drukken. Het lampje gaat direct aan. Het lampje buiten op de gang is verbonden met dit lampje via wifi. Dit lampje gaat aan als het binnen aangaat en uit als het binnen uit gaat. Het kan wel enkele seconden duren voordat het lampje buiten het signaal opgepikt heeft.

Hetzelfde systeem wordt gebruikt voor het ophalen van stalen door het centraal labo in de kleine SAS.

Wanneer er een stroomonderbreking plaatsvindt, gaat het systeem vanzelf weer aan zodra het stroom heeft. Alle lampjes zullen wel uit zijn.

Wanneer er een wifi onderbreking is, zullen de lampjes buiten aan gaan.

7.3. Werking van de code

De ESP 32's hebben een constante stroomtoevoer van 3.3V nodig voor het uitvoeren van hun functie. Deze wordt voorzien via een Micro-B USB kabel. De code wordt ook op deze manier geüpload. De kabel die hiervoor gebruikt wordt moet data kunnen versturen. De code moet gecompileerd worden voor deze geüpload kan worden. Hiervoor wordt best de Arduino IDE software gebruikt. De code is geschreven in een combinatie van C/C++ die "Arduino programming language" heet.

Elke locatie heeft zijn eigen ESP 32 computer met zijn eigen code.
De code in de grote Sas: Logistics Log
De code in de kleine Sas: Logistics Lab
De code in de gang: Logistics Hallway

Bij het uploaden van de code op de ESP 32 van de grote Sas, moet er op de "Boot" knop gedrukt worden. Als dit niet gebeurt, zal de code niet geüpload worden en zal men een error krijgen.

Logistics Log/Logistics Lab

De code van Logistics Log en Logistics Lab zijn bijna identiek. Zij hebben dezelfde werking maar een ander IP adres en PIN nummers (relevant voor het verbinden van kabels).

De ESP 32 start automatisch op zodra het de correcte hoeveelheid stroom ontvangt. Het systeem begint met het elektrisch signaal van bepaalde pinnen te koppelen aan de corresponderende elementen. De led zal niet branden bij het opstarten.

Daarna zal de computer verbinding proberen te maken met het netwerk opgegeven netwerk met paswoord . Het zal een kleine webpagina aanmaken op zijn eigen statisch IP adres. Hierop zal de ESP 32 weergeven of de led aan staat of niet.

Daarna wacht de computer continu of er een signaal van de drukknop komt. Wanneer dat gebeurt, verandert het toestel de status van de led (van aan naar uit of omgekeerd). Dit wordt dan ook direct geüpload op de ESP32's IP adres.

Indien de ESP 32 geen verbinding kan maken met het internet zal de kleine interne let blauw knipperen. Indien de ESP 32 zijn statisch IP adres niet kan verkrijgen van het netwerk zal de kleine interne blauwe led branden.

Logistics Hallway

De ESP 32 start automatisch op zodra het de correcte hoeveelheid stroom ontvangt. Het systeem begint met het elektrisch signaal van bepaalde pinnen te koppelen aan de corresponderende elementen. De led zal niet branden bij het opstarten.

Daarna zal de computer verbinding proberen te maken met het netwerk opgegeven netwerk met paswoord. Als dit lukt, zal het eerst zoeken naar het webadres (IP adres) van de ene computer. Hier zal het de informatie van de webpagina uitlezen en controleren of de led aan staat. Indien dit zo is, gaat de corresponderende led ook aan. Zoniet, dan gaat de led uit. Hierna zal het de verbinding met dit webadres verbreken en proberen verbinding te maken met het webadres van de andere ESP 32. Hier wordt dezelfde methode herhaald. Deze cyclus wordt constant herhaald.

Als de led geen verbinding kan maken met 1 (of beide) van de webpagina's gaat de corresponderende (of beide) led gaan branden.

8. Onderhoud

Het systeem wordt niet geüpdatet. Het systeem is enkel afhankelijk van netstroom en een wifi-verbinding. Wanneer de wifi in AZ Zeno verandert van naam/paswoord moet dit aangepast worden in de code.

9. Gerelateerde documenten

10. Wijzigingen

11. Literatuur

12. Scripts

```
// Project: Logistics system
// Function: Streamlining the service the logistics department provides
with an IoT-systems based on led signals.
// This application is developed for the pathology labo of AZ Zeno.
// Name: Yoram Vandenhouwe
// Start of project: 13/02/2024
// Implementation: 29/05/2024
// Version: 1
// Location: Logistics Hallway

// Load Wi-Fi library
#include <WiFi.h>
#include <HTTPClient.h>

// set pin numbers
const int ledLog = 4;  // the number of the pushbutton pin for
logistics department
const int ledlab =  5;    // the number of the LED pin for logistics
department


// Auxiliar variables to store the current output state
String ledlabState = "/lab/off";
String ledLogState = "/log/off";
char* result;

// Network credentials
char* ssid = "AZ_Zeno_Gast";
```

```
char* password = "";



// Set web server port number to 80
WiFiServer server(80);

// Variable to store the HTTP request
String header;

// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;
//Your Domain name with URL path or IP address with path
String serverNameLog = "http://172.27.4.159/";
String serverNameLab = "http://172.27.4.92/";

// The following variables are unsigned longs because the time,
measured in
// milliseconds, will quickly become a bigger number than can be stored
in an int.
unsigned long lastTime = 0;
// Timer set to 10 minutes (600000)
//unsigned long timerDelay = 600000;
// Set timer to 5 seconds (5000)
unsigned long timerDelay = 5000;



void setup() {
   // Serial port for debugging purposes
  Serial.begin(115200);

  // Initialize the LED pin as an output for logistics department
```

```
  pinMode(ledLog, OUTPUT);
  pinMode(ledlab, OUTPUT);

   digitalWrite(ledLog, LOW);
   digitalWrite(ledlab, LOW);

  // Connect to Wi-Fi network with SSID and password
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
       digitalWrite(ledLog, HIGH);
   digitalWrite(ledlab, HIGH);
    delay(500);
    Serial.print(".");
  }
  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}

void loop() {

  if ((millis() - lastTime) > timerDelay) {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
      HTTPClient http;

      String serverPathLog = serverNameLog;

      // Your Domain name with URL path or IP address with path
      http.begin(serverPathLog.c_str());
```

```
      // Send HTTP GET request
      int httpResponseCode = http.GET();
      // if there is a response:
      if (httpResponseCode>0) {
        Serial.print("HTTP Response code: ");
        Serial.println(httpResponseCode);
        String payload = http.getString(); // Read the webpage.
        Serial.println(payload);
      // Convert the payload and word to lowercase for case-insensitive
comparison
        payload.toLowerCase();
      ledLogState.toLowerCase();
        // Find the string present in the payload, keep the string form
the point were a match was found
        char* resultLog = strstr(payload.c_str(), ledLogState.c_str());
      if (resultLog != nullptr) { // if a match was found, set led to
"off"
    digitalWrite(ledLog, LOW);
    } else {
    digitalWrite(ledLog, HIGH);
    }
    }
      else { // if website was found, set led to "on" and print error
message
        Serial.print("Error code: ");
        Serial.println(httpResponseCode);
        digitalWrite(ledLog, HIGH);
      }
http.end();



      String serverPathLab = serverNameLab;
```

```
      // Your Domain name with URL path or IP address with path
      http.begin(serverPathLab.c_str());


      // Send HTTP GET request
      httpResponseCode = http.GET();
      // if there is a response:
      if (httpResponseCode>0) {
        Serial.print("HTTP Response code: ");
        Serial.println(httpResponseCode);
        String payload = http.getString();  // Read the webpage.
        Serial.println(payload);
      // Convert the payload and word to lowercase for case-insensitive
comparison
        payload.toLowerCase();
        ledlabState.toLowerCase();
         // Find the string present in the payload, keep the string
form the point were a match was found
        char* resultLab = strstr(payload.c_str(), ledlabState.c_str());
        Serial.print(resultLab);
      if (resultLab != nullptr) { // if a match was found, set led to
"off"
    digitalWrite(ledlab, LOW);
    } else {
    digitalWrite(ledlab, HIGH);
    }

      }
      else { // if website was found, set led to "on" and print error
message
        Serial.print("Error code: ");
        Serial.println(httpResponseCode);
        digitalWrite(ledlab, HIGH);
      }
```

```
      // Free resources
      http.end();
    }
    else {
      Serial.println("WiFi Disconnected");
    }
    lastTime = millis();
  }
}
```

```
// Project: Logistics system
// Function: Streamlining the service the logistics department provides
with an IoT-systems based on led signals.
// This application is developed for the pathology labo of AZ Zeno.
// Name: Yoram Vandenhouwe
// Start of project: 13/02/2024
// Implementation: 29/05/2024
// Version: 1
// Location: Logistics Lab

// Load Wi-Fi library
#include <WiFi.h>
#include <HTTPClient.h>


// set pin numbers
const int buttonLog = 4;  // the number of the pushbutton pin for
logistics department
const int ledLog =  5;    // the number of the LED pin for logistics
department
#define INTERNAL_LED 2 // The number of the internal led
// variable for storing the pushbutton status
int buttonState = 0;
```

```
// Variable to keep track of the led status
int ledLogStatus;
// To indicate the current state on the webpage
String ledSasLogState = "off";

// Network credentials
char* ssid = "AZ_Zeno_Gast";
char* password = "";


// Set web server port number to 80
WiFiServer server(80);

// Variable to store the HTTP request
String header;

// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;



IPAddress local_IP(172, 27, 4, 92); // Desired Static IP Address
http://172.27.4.92/
IPAddress subnet(255, 255, 255, 0);
IPAddress gateway(172, 27, 4, 1);


void setup() {
  // Serial port for debugging purposes
  Serial.begin(115200);
```

```
    // initialize digital pin LED_BUILTIN as an output.
  pinMode(INTERNAL_LED, OUTPUT);

  // initialize the pushbutton pin as an input for logistics department
  pinMode(buttonLog, INPUT);
  // initialize the LED pin as an output for logistics department
  pinMode(ledLog, OUTPUT);

  // Connect to Wi-Fi network with SSID and password
    WiFi.mode(WIFI_STA);
    // Configures Static IP Address
   if (!WiFi.config(local_IP, gateway, subnet))
    {
      digitalWrite(INTERNAL_LED, HIGH);   // turn the LED on (HIGH is
the voltage level)
         Serial.println("Configuration Failed!");
    }

  // Try to connect to wifi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
      digitalWrite(INTERNAL_LED, HIGH);   // turn the LED on (HIGH is
the voltage level)
        delay(1000);                       // wait for a second
        digitalWrite(INTERNAL_LED, LOW);    // turn the LED off by
making the voltage LOW
        delay(1000);
    Serial.print(".");
  }
  digitalWrite(INTERNAL_LED, LOW);    // turn the LED off by making the
voltage LOW
  server.begin();

    // Print the assigned network settings (for debugging purposes)
```

```
    Serial.println("\nConnected to the WiFi network");
    Serial.print("Local ESP32 IP: ");
    Serial.println(WiFi.localIP());
    Serial.print("Subnet Mask: " );
    Serial.println(WiFi.subnetMask());
    Serial.print("Gateway IP: ");
    Serial.println(WiFi.gatewayIP());
    Serial.print("DNS 1: ");
    Serial.println(WiFi.dnsIP(0));
    Serial.print("DNS 2: ");
    Serial.println(WiFi.dnsIP(1));



}

void loop() {


  // read the state of the pushbutton value
  buttonState = digitalRead(buttonLog);

  // check if the pushbutton is pressed for logistics department
  // if it is, the buttonState is HIGH
  if ( buttonState== HIGH){
    delay(300);                                  // delay 300
miliseconds
    if ( ledLogStatus == 1){                    // if button was of; set
value to 1. if botton was on; set value to 0
    ledLogStatus =0;
    }else {
      ledLogStatus = 1;
    }
  }
  if (ledLogStatus == HIGH) {     // if the value was 1 ( HIGH):
```

```
    // turn LED on
    digitalWrite(ledLog, HIGH);
   ledSasLogState = "on";

  } else {
    // turn LED off
    digitalWrite(ledLog, LOW);
    ledSasLogState = "off";
  }

   WiFiClient client = server.available();   // Listen for incoming
clients

  if (client) {                             // If a new client
connects,
    currentTime = millis();
    previousTime = currentTime;
    Serial.println("New Client.");          // print a message out in
the serial port
    String currentLine = "";                // make a String to hold
incoming data from the client
    while (client.connected() && currentTime - previousTime <=
timeoutTime) {  // loop while the client's connected
      currentTime = millis();
      if (client.available()) {             // if there's bytes to read
from the client,
        char c = client.read();             // read a byte, then
        Serial.write(c);                    // print it out the serial
monitor
        header += c;
        if (c == '\n') {                    // if the byte is a newline
character
          // if the current line is blank, you got two newline
characters in a row.
          // that's the end of the client HTTP request, so send a
```

```
response:
          if (currentLine.length() == 0) {
            // HTTP headers always start with a response code (e.g.
HTTP/1.1 200 OK)
            // and a content-type so the client knows what's coming,
then a blank line:
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();


            // Display the HTML web page
            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta name=\"viewport\"
content=\"width=device-width, initial-scale=1\">");
            client.println("<link rel=\"icon\" href=\"data:,\">");
            // CSS to style the on/off "button"
            // Feel free to change the background-color and font-size
attributes to fit your preferences
            client.println("<style>html { font-family: Helvetica;
display: inline-block; margin: 0px auto; text-align: center;}");
            client.println(".button { background-color: #4CAF50;
border: none; color: white; padding: 16px 40px;");
            client.println("text-decoration: none; font-size: 30px;
margin: 2px; cursor: pointer;}");
            client.println(".button2 {background-color:
#555555;}</style></head>");

            // Web Page Heading
            client.println("<body><h1>ESP32 Web Server</h1>");

            // Display current state, and ON/OFF button for logistics
led
            client.println("<p>Logistieke dienst led - State " +
```

```
ledSasLogState + "</p>");
            // If the logistics led is on, it displays the ON button
            if (ledSasLogState== "on") {
              client.println("<p><a id=\"/log/on\"><button
class=\"button\">ON</button></a></p>");
            } else {
              client.println("<p><a id=\"/log/off\"><button
class=\"button button2\">OFF</button></a></p>");
            }
            client.println("</body></html>");

            // The HTTP response ends with another blank line
            client.println();
            // Break out of the while loop
            break;
          } else { // if you got a newline, then clear currentLine
            currentLine = "";
          }
        } else if (c != '\r') {  // if you got anything else but a
carriage return character,
          currentLine += c;      // add it to the end of the
currentLine
        }
      }
    }
    // Clear the header variable
    header = "";
    // Close the connection
    client.stop();
    Serial.println("Client disconnected.");
    Serial.println("");
  }

}
```

```
// Project: Logistics system
// Function: Streamlining the service the logistics department provides
with an IoT-systems based on led signals.
// This application is developed for the pathology labo of AZ Zeno.
// Name: Yoram Vandenhouwe
// Start of project: 13/02/2024
// Implementation: 29/05/2024
// Version: 1
// Location: Logistics Log

// Load Wi-Fi library
#include <WiFi.h>
#include <HTTPClient.h>

// set pin numbers
const int buttonLog = 4;  // the number of the pushbutton pin for
logistics department
const int ledLog =  5;    // the number of the LED pin for logistics
department
#define INTERNAL_LED 2 // The number of the internal led
// variable for storing the pushbutton status
int buttonState = 0;

// Auxiliar variables to store the current output state
int ledLogStatus;

// To indicate the current state on the webpage
String ledSasLabState = "off";

// Network credentials
char* ssid = "AZ_Zeno_Gast";
char* password = "";
```

```
// Set web server port number to 80
WiFiServer server(80);

// Variable to store the HTTP request
String header;

// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;

IPAddress local_IP(172, 27, 4, 93); // Desired Static IP Address
http://172.27.4.93/
IPAddress subnet(255, 255, 255, 0);
IPAddress gateway(172, 27, 4, 1);

void setup() {
   // Serial port for debugging purposes
  Serial.begin(115200);

      // initialize digital pin LED_BUILTIN as an output.
  pinMode(INTERNAL_LED, OUTPUT);

  // initialize the pushbutton pin as an input for logistics department
  pinMode(buttonLog, INPUT);
  // initialize the LED pin as an output for logistics department
  pinMode(ledLog, OUTPUT);

 // Connect to Wi-Fi network with SSID and password
    WiFi.mode(WIFI_STA);
    // Configures Static IP Address
   if (!WiFi.config(local_IP, gateway, subnet))
```

```
    {
      digitalWrite(INTERNAL_LED, HIGH);   // turn the LED on (HIGH is
the voltage level)
         Serial.println("Configuration Failed!");
    }

  // Try to connect to wifi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
      digitalWrite(INTERNAL_LED, HIGH);   // turn the LED on (HIGH is
the voltage level)
        delay(1000);                       // wait for a second
        digitalWrite(INTERNAL_LED, LOW);    // turn the LED off by
making the voltage LOW
        delay(1000);
    Serial.print(".");
  }
  digitalWrite(INTERNAL_LED, LOW);    // turn the LED off by making the
voltage LOW
  server.begin();

    // Print the assigned network settings (for debugging purposes)
    Serial.println("\nConnected to the WiFi network");
    Serial.print("Local ESP32 IP: ");
    Serial.println(WiFi.localIP());
    Serial.print("Subnet Mask: " );
    Serial.println(WiFi.subnetMask());
    Serial.print("Gateway IP: ");
    Serial.println(WiFi.gatewayIP());
    Serial.print("DNS 1: ");
    Serial.println(WiFi.dnsIP(0));
    Serial.print("DNS 2: ");
    Serial.println(WiFi.dnsIP(1));

}
```

```
void loop() {


  // read the state of the pushbutton value
  buttonState = digitalRead(buttonLog);

  // check if the pushbutton is pressed for logistics department
  // if it is, the buttonState is HIGH
  if ( buttonState== HIGH){
    delay(300);                                  // delay 300
milliseconds
    if ( ledLogStatus == 1){                    // if button was of; set
value to 1. if botton was on; set value to 0
    ledLogStatus =0;
    }else {
      ledLogStatus = 1;
    }
  }
  if (ledLogStatus == HIGH) {     // if the value was 1 ( HIGH):
    // turn LED on
    digitalWrite(ledLog, HIGH);
   ledSasLabState = "on";

  } else {
    // turn LED off
    digitalWrite(ledLog, LOW);
    ledSasLabState = "off";
  }

   WiFiClient client = server.available();   // Listen for incoming
clients

  if (client) {                              // If a new client
connects,
```

```
    currentTime = millis();
    previousTime = currentTime;
    Serial.println("New Client.");          // print a message out in
the serial port
    String currentLine = "";                // make a String to hold
incoming data from the client
    while (client.connected() && currentTime - previousTime <=
timeoutTime) {  // loop while the client's connected
      currentTime = millis();
      if (client.available()) {             // if there's bytes to read
from the client,
        char c = client.read();             // read a byte, then
        Serial.write(c);                    // print it out the serial
monitor
        header += c;
        if (c == '\n') {                    // if the byte is a newline
character
          // if the current line is blank, you got two newline
characters in a row.
          // that's the end of the client HTTP request, so send a
response:
          if (currentLine.length() == 0) {
            // HTTP headers always start with a response code (e.g.
HTTP/1.1 200 OK)
            // and a content-type so the client knows what's coming,
then a blank line:
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();


            // Display the HTML web page
            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta name=\"viewport\"
```

```
content=\"width=device-width, initial-scale=1\">");
            client.println("<link rel=\"icon\" href=\"data:,\">");
            // CSS to style the on/off "button"
            // Feel free to change the background-color and font-size
attributes to fit your preferences
            client.println("<style>html { font-family: Helvetica;
display: inline-block; margin: 0px auto; text-align: center;}");
            client.println(".button { background-color: #4CAF50;
border: none; color: white; padding: 16px 40px;");
            client.println("text-decoration: none; font-size: 30px;
margin: 2px; cursor: pointer;}");
            client.println(".button2 {background-color:
#555555;}</style></head>");

            // Web Page Heading
            client.println("<body><h1>ESP32 Web Server</h1>");

            // Display current state, and ON/OFF button lab led
            client.println("<p> id=led Centraal labo led - State " +
ledSasLabState + "</p>");
            // If lab led is on, it displays the ON button
            if (ledSasLabState== "on") {
              client.println("<p><a id=\"/lab/on\"><button
class=\"button\">ON</button></a></p>");
            } else {
              client.println("<p><a id=\"/lab/off\"><button
class=\"button button2\">OFF</button></a></p>");
            }


            // The HTTP response ends with another blank line
            client.println();
            // Break out of the while loop
            break;
          } else { // if you got a newline, then clear currentLine
```

```
            currentLine = "";
          }
        } else if (c != '\r') {  // if you got anything else but a
carriage return character,
          currentLine += c;      // add it to the end of the
currentLine
        }
      }
    }
    // Clear the header variable
    header = "";
    // Close the connection
    client.stop();
    Serial.println("Client disconnected.");
    Serial.println("");
  }

}
```