Andre Fraga ngày 24 tháng 11 vào năm 2021 5 phút để đọc

ESP32 với Ethernet có dây: Sử dụng LAN8720

Cập nhật: ngày 1 tháng 3.

ESP32 đã có kết nối gốc với Internet qua Wi-Fi, nhưng không phải lúc nào chúng ta cũng có thể chỉ dựa vào hình thức kết nối này, chẳng hạn như trong các ứng dụng không có mạng Wi-Fi hoặc các ứng dụng đường dài. không thể sử dụng Wi-Fi, vì vậy để khắc phục tình trạng này, chúng tôi cần sử dụng kết nối có dây. Và đó là nơi Shield Ethernet LAN8720 được sử dụng trong các dự án của chúng tôi, cho phép chúng tôi kết nối ESP32 của mình với mạng Ethernet có dây và trong bài đăng này, chúng tôi sẽ hướng dẫn bạn cách tạo kết nối này.

Bản tóm tắt:

1 – Sơ đồ chân của Shield Ethernet LAN8720 với ESP32

Thật không may, Shield Ethernet LAN8720 không hoạt động khi chỉ kết nối trực tiếp với ESP32, để chúng tôi có thể kết nối nó, chúng tôi sẽ phải thực hiện một số thay đổi.

Theo liên kết này:

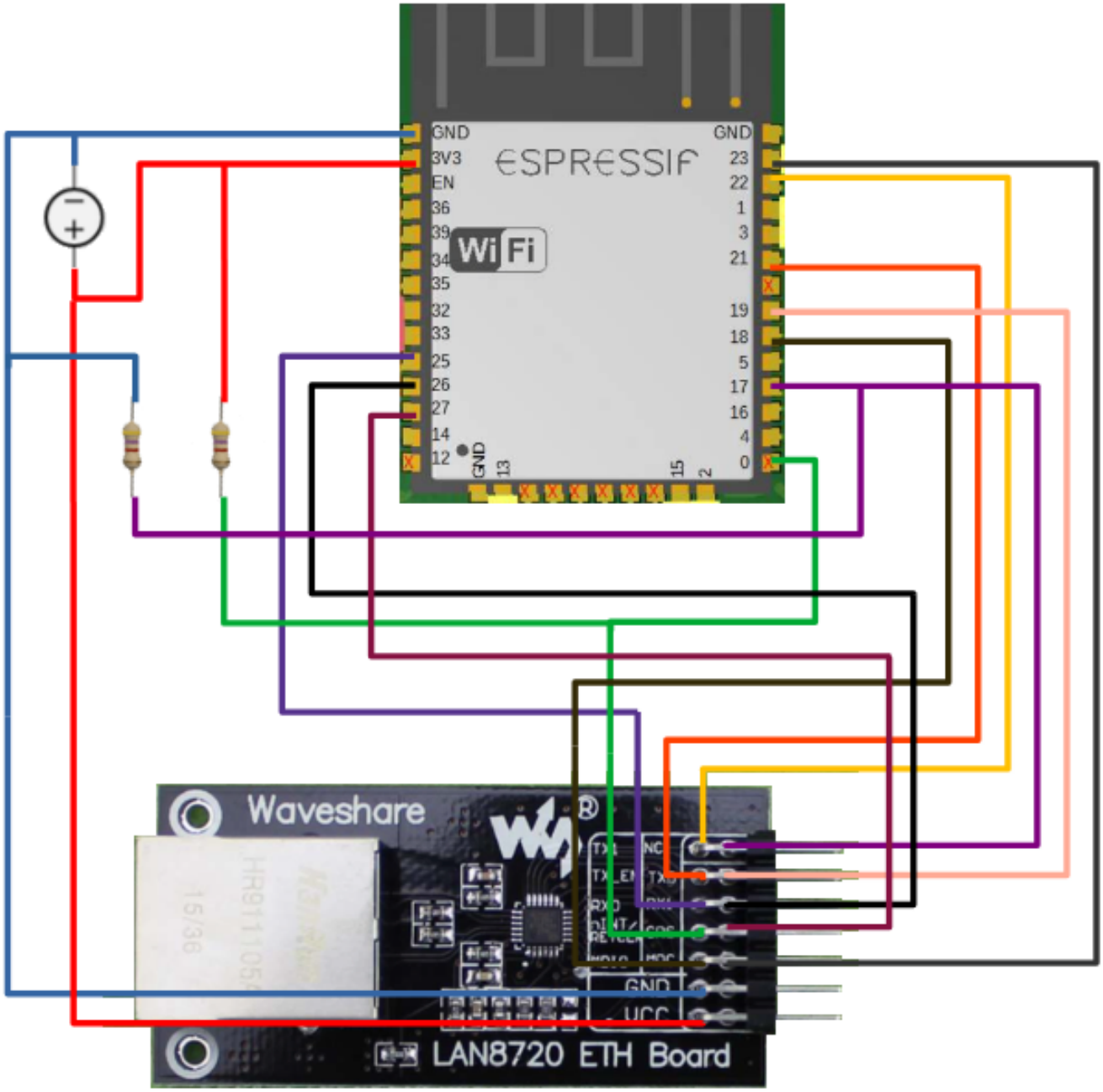
ESP32	SHIELD ETHERNET LAN8720	RESISTOR
GPIO17	PHY_POWER: **NC**	4k7Ω Pulldown
GPIO22	EMAC_TXD1: **TX1**	
GPIO19	EMAC_TXD0: **TX0**	
GPIO21	EMAC_TX_EN: **TX_EN**	
GPIO26	EMAC_RXD1: **RX1**	
GPIO25	EMAC_RXD0: **RX0**	
GPIO27	EMAC_RX_DV: **CRS**	
GPIO00	EMAC_TX_CLK: **nINT / REFCLK** (50 MHz)	4k7Ω pullup
GPIO23	SMI_MDC: **MDC**	
GPIO18	SMI_MDIO: **MDIO**	
GND	**GND**	
3.3V	**3.3V**	

Vì các chân được đánh dấu màu xanh lam là các chân cố định, không thể kết nối với các GPIO của ESP32 khác.

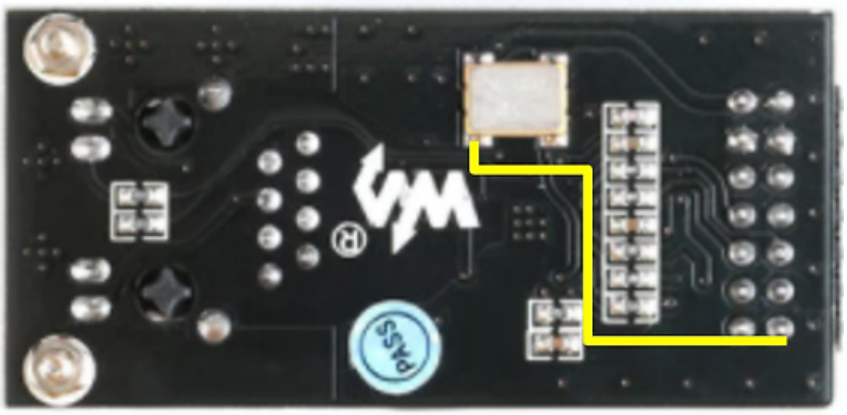
Chân GPIO0 không thể ở mức THẤP trong quá trình khởi tạo, nếu không bộ tải khởi động sẽ chờ lập trình nối tiếp và để điều này không xảy ra, chân phải được giữ ở mức CAO trong quá trình khởi tạo. Vì vậy, chúng tôi sử dụng điện trở 4k7Ω để thực hiện pullup trên đầu vào. Kết nối này được thực hiện vì GPIO0 cũng là đầu vào đồng hồ cho khối chức năng ESP32 EMAC.

Và điều cần thiết là 50MHz của REFCLK được cung cấp ngay trước khi LAN8720 được khởi tạo. Đối với điều này, phải sử dụng chân "Bật", là chân kích hoạt bộ tạo dao động, nếu nó được giữ ở mức THẤP, đầu ra sẽ bị tắt và chân GPIO17 được xác định là PHY_POWER trong mã, là đầu vào khởi tạo mà sau khi bắt đầu được cấu hình lại làm đầu ra và được xác định là CAO, thì kết nối này được thực hiện bằng cách kết nối chân NC của Tấm chắn với "Bật" của bộ tạo dao động của Tấm chắn và với chân GPIO17, để đảm bảo rằng chân kích hoạt của bộ tạo dao động ở mức THẤP mức, một điện trở 4k7Ω thực hiện kéo xuống.

2 - Sơ đồ kết nối



Hình 01: Kết nối lá chắn với ESP32.

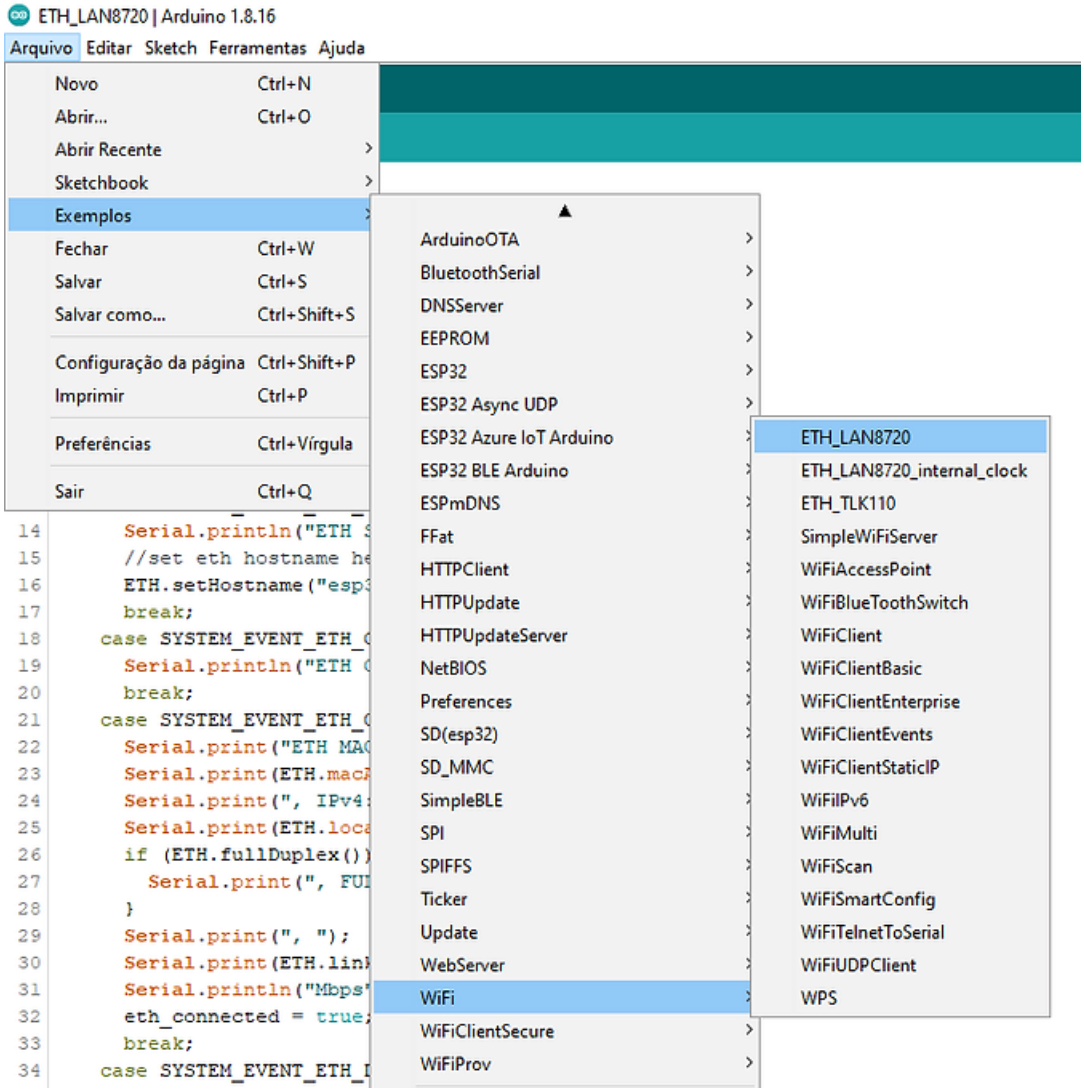


Hình 02: Shield Oscillator Kích hoạt kết nối.

3 - Mã

Đối với các liên kết, mã yêu cầu một số thay đổi để chạy trực tiếp trên ESP32. Bắt đầu từ mã ví dụ có sẵn trong Arduino IDE, có thể tìm thấy trong IDE tại:

Tệp > Ví dụ > WiFi > ETH_LAN8720



Hình 03: Mã mẫu.

Chúng tôi cần đưa vào phần đầu của mã định nghĩa về các chân mà chúng tôi đã liên kết trước đó và cách LAN8720 trên bảng Waveshare được khởi tạo để sử dụng địa chỉ I2C 1, chúng tôi cũng cần định cấu hình điều này trong mã, bên dưới chúng tôi trình bày các thay đổi :

```
/*
    This sketch shows the Ethernet event usage
*/
#include <ETH.h>

// Pino do sinal de habilitação para o oscilador de cristal externo
//(-1 para desabilitar para fonte APLL interna)
#define ETH_PHY_POWER 17
// Tipo de Ethernet PHY
#define ETH_TYPE ETH_PHY_LAN8720
// Endereço I2C de Ethernet PHY (0 ou 1 para LAN8720)
#define ETH_ADDR 1
#define ETH_PHY_ADDR 1
// Pino do sinal de relógio I2C para Ethernet PHY
#define ETH_MDC_PIN 23
// Pino do sinal I2C IO para Ethernet PHY
#define ETH_MDIO_PIN 18
// Clock
#define ETH_CLK_MODE ETH_CLOCK_GPIO0_IN

static bool eth_connected = false;

```

Hình 04: Định nghĩa chân và Địa chỉ I2C.

Và bên trong thiết lập void, chúng ta cần đưa vào “ETH.begin(;” việc khởi tạo các chân này, chúng tôi xác định:

```
void setup()
{
  Serial.begin(115200);
  WiFi.onEvent(WiFiEvent);

  ETH.begin( PHY1 , 17, 23, 18 , ETH_PHY_LAN8720 );

}
```

Hình 05: Khởi tạo.

Với những thay đổi này, mã ví dụ đã hoạt động và sẽ trả lại cho chúng tôi xác nhận quyền truy cập vào google trong màn hình nối tiếp, như sau:

```
12:30:10.569 -> ets Jun  8 2016 00:22:57
12:30:10.569 ->
12:30:10.569 -> rst:0x10 (RTCWDT_RTC_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
12:30:10.569 -> configsip: 0, SPIWP:0xee
12:30:10.615 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
12:30:10.615 -> mode:DIO, clock div:1
12:30:10.615 -> load:0x3fff0018,len:4
12:30:10.615 -> load:0x3fff001c,len:1216
12:30:10.615 -> ho 0 tail 12 room 4
12:30:10.615 -> load:0x40078000,len:10944
12:30:10.615 -> load:0x40080400,len:6388
12:30:10.615 -> entry 0x400806b4
12:30:10.848 -> ETH Started
12:30:14.837 -> ETH Connected
12:30:14.884 -> ETH MAC: 7C:9E:BD:49:53:5F, IPv4: 192.168.100.134, FULL_DUPLEX, 100Mbps
12:30:20.866 ->
12:30:20.866 -> connecting to google.com
12:30:20.958 -> HTTP/1.1 301 Moved Permanently
12:30:20.958 -> Location: http://www.google.com/
12:30:20.958 -> Content-Type: text/html; charset=UTF-8
12:30:20.958 -> Date: Wed, 24 Nov 2021 14:30:20 GMT
12:30:20.958 -> Expires: Fri, 24 Dec 2021 14:30:20 GMT
12:30:20.958 -> Cache-Control: public, max-age=2592000
12:30:20.958 -> Server: gws
12:30:20.958 -> Content-Length: 219
12:30:20.958 -> X-XSS-Protection: 0
12:30:20.958 -> X-Frame-Options: SAMEORIGIN
12:30:20.958 ->
12:30:20.958 -> <HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
12:30:21.005 -> <TITLE>301 Moved</TITLE></HEAD><BODY>
12:30:21.005 -> <H1>301 Moved</H1>
12:30:21.005 -> The document has moved
12:30:21.005 -> <A HREF="http://www.google.com/">here</A>.
12:30:21.005 -> </BODY></HTML>
12:30:21.005 -> closing connection
12:30:21.005 ->
```

Hình 06: Mã ví dụ trả về.

Nhưng đối với blog này, chúng tôi muốn triển khai một đoạn mã thú vị hơn để thể hiện chức năng này.

Do đó, chúng tôi sẽ sử dụng mã để kết nối thẻ của mình với mạng và lấy địa chỉ IPv4, qua đó chúng tôi có thể truy cập một trang trong trình duyệt nơi chúng tôi có thể kích hoạt đầu ra trên ESP32.

```
#bao gồm < ETH . giờ >

// Kích hoạt chân tín hiệu cho bộ tạo dao động tinh thể bên ngoài (-1
để tắt đối với nguồn APLL bên trong)
#define ETH_PHY_POWER  17
// Loại Ethernet PHY
#define ETH_TYPE  ETH_PHY_LAN8720
// Địa chỉ I2C của Ethernet PHY (0 hoặc 1 cho LAN8720 )
#define ETH_ADDR  1
#define ETH_PHY_ADDR  1
// Chân tín hiệu đồng hồ I2C cho Ethernet PHY
#define ETH_MDC_PIN  23
```

```
// Chân tín hiệu I2C IO cho Ethernet PHY
#define ETH_MDIO_PIN  18
// Đồng hồ
#define ETH_CLK_MODE  ETH_CLOCK_GPIO0_IN

bool tĩnh eth_connected =  false ;
Máy chủ
WiFiServer ( 80 ) ;


void  WiFiEvent ( WiFiEvent_t event )
{
  switch  ( event )  {
    case  SYSTEM_EVENT_ETH_START :
      Serial . println ( "ETH đã bắt đầu" ) ;
      // đặt tên máy chủ eth tại đây
      ETH . setHostname ( "esp32-ethernet" ) ;
      phá vỡ ;
    trường hợp  SYSTEM_EVENT_ETH_CONNECTED :
      Nối tiếp . println ( "ETH đã kết nối" ) ;
      phá vỡ ;
    trường hợp SYSTEM_EVENT_ETH_GOT_IP :
      Nối tiếp . in ( "ETH MAC:" ) ;
      nối tiếp . in ( ETH . macAddress ( ) ) ;
      nối tiếp . in ( ", IPv4: " ) ;
      nối tiếp . in ( ETH . localIP ( ) ) ;
      nếu  ( ETH . fullDuplex ( ) )  {
        Nối tiếp . in( ", FULL_DUPLEX" ) ;
      }
      Nối tiếp . in ( ", " ) ;
      nối tiếp . in ( ETH . linkSpeed ( ) ) ;
      nối tiếp . println ( "Mbps" ) ;
      eth_connected =  true ;
      phá vỡ ;
    trường hợp  SYSTEM_EVENT_ETH_DISCONNECTED :
      Nối tiếp . println ( "ETH Ngắt kết nối" ) ;
      eth_connected=  sai ;
      phá vỡ ;
    trường hợp  SYSTEM_EVENT_ETH_STOP :
      Nối tiếp . println ( "ETH Đã dừng" ) ;
```

```
      eth_connected =  sai ;
      phá vỡ ;
    mặc định :
      phá vỡ ;
  }
}


 thiết lập vô hiệu ( )
{
  Nối tiếp . bắt đầu ( 115200 ) ;
  Wi-Fi . onEvent ( WiFiEvent ) ;

  ETH . bắt đầu (  PHY1  ,  17 ,  23 ,  18  ,  ETH_PHY_LAN8720  ) ;

  máy chủ . bắt đầu ( ) ;

}


 vòng lặp void ( )
{
  Máy khách WiFiClient = máy chủ . có sẵn ( ) ;

  if  ( client )  {                              // nếu bạn nhận được
client
    Serial . println ( "Khách hàng mới." ) ;           // in thông báo
qua cổng nối tiếp
    String currentLine =  "" ;                 // tạo một chuỗi để chứa
dữ liệu đầu vào của máy khách
    trong khi  ( client .connected ( ) ) { // lặp trong khi máy khách
được kết nối if ( client . available ( ) ) { // nếu có byte để đọc từ
máy khách
       char c
                    = khách hàng . đọc ( ) ;              // đọc một
byte, sau đó
       Serial . viết ( c ) ;                     // in ra màn hình nối
tiếp
       nếu  ( c ==  '\n' )  {                     // nếu byte là ký tự
xuống dòng
```

```
          // nếu dòng hiện tại trống, bạn có hai ký tự xuống dòng trên
một dòng.
          // đó là phần cuối của yêu cầu HTTP từ máy khách, vì vậy hãy
gửi phản hồi:
          if  ( currentLine . length ( )  ==  0 )  {
            // Các tiêu đề HTTP luôn bắt đầu bằng mã phản hồi (ví dụ:
HTTP/1.1 200 OK)
            // và một loại nội dung để cho khách hàng biết điều gì sắp
xảy ra, sau đó là một dòng trống:
            client . println ( "HTTP/1.1 200 OK" ) ;
            khách hàng . println ( "Kiểu nội dung:text/html" ) ;
            khách hàng . println ( ) ;

            // o conteúdo da resposta HTTP segue o cabeçalho:
            client . print ( "Nhấp vào <a href=\"/H\">đây</a> để bật
đèn LED ở chân 5.<br>" ) ;
            khách hàng . print ( "Nhấp vào <a href=\"/L\">đây</a> để
tắt đèn LED ở chân 5.<br>" ) ;

            // Phản hồi HTTP kết thúc bằng một dòng trống khác:
            client . println ( ) ;
            // thoát khỏi vòng lặp while:
            break ;
          }  else  {     // nếu bạn có một dòng mới, hãy xóa
currentLine:
            currentLine =  "" ;
          }
        }  else  if  ( c !=  '\r' )  {   // nếu bạn có ký tự nào khác
ngoài ký tự xuống dòng,
          currentLine += c ;       // thêm nó vào cuối dòng hiện tại
        }

        // Kiểm tra xem yêu cầu của máy khách là "GET /H" hay "GET /L":
        if  ( currentLine . endWith ( "GET /H" ) )  {
          digitalWrite ( 5 ,  HIGH ) ;               // GET /H bật đèn
LED
        }
        if  ( currentLine . endWith ( "GET /L" ) )  {
          digitalWrite ( 5 ,  LOW ) ;                // NHẬN /L tắt
```

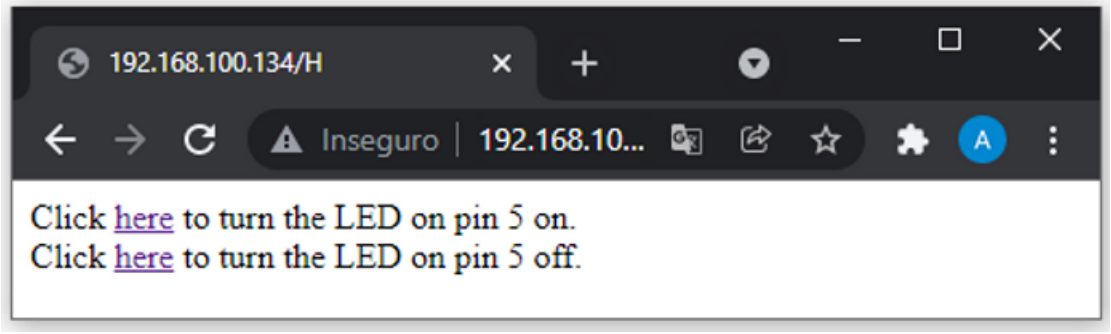
```
đèn LED
        }
      }
    }
    // đóng kết nối:
    client . dừng lại () ;
    nối tiếp . println ( "Máy khách bị ngắt kết nối." ) ;
  }

}
```

Để có được địa chỉ IPv4 mà ESP32 đã tìm kiếm trên mạng, chúng ta phải mở màn hình nối tiếp và với ESP32 được kết nối với cáp mạng, nó sẽ in như sau:

```
12:43:00.940 -> ets Jun  8 2016 00:22:57
12:43:00.940 ->
12:43:00.940 -> rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
12:43:00.987 -> flash read err, 1000
12:43:00.987 -> ets_main.c 371
12:43:01.313 -> ets Jun  8 2016 00:22:57
12:43:01.313 ->
12:43:01.313 -> rst:0x10 (RTCWDT_RTC_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
12:43:01.313 -> configsip: 0, SPIWP:0xee
12:43:01.313 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
12:43:01.359 -> mode:DIO, clock div:1
12:43:01.359 -> load:0x3fff0018,len:4
12:43:01.359 -> load:0x3fff001c,len:1216
12:43:01.359 -> ho 0 tail 12 room 4
12:43:01.359 -> load:0x40078000,len:10944
12:43:01.359 -> load:0x40080400,len:6388
12:43:01.359 -> entry 0x400806b4
12:43:01.592 -> ETH Started
12:43:05.582 -> ETH Connected
12:43:05.629 -> ETH MAC: 7C:9E:BD:49:53:5F, IPv4: 192.168.100.134 FULL_DUPLEX, 100Mbps
```

Hình 07: Lấy địa chỉ IPv4.

Sau khi lấy được địa chỉ IPv4, chúng ta phải sao chép và dán địa chỉ đó vào trình duyệt, sau đó chúng ta sẽ có quyền truy cập vào trang nơi chúng ta sẽ kích hoạt đèn LED của mình.



Hình 08: Truy cập trang.

Khi chúng ta click kích hoạt LED thì trang sẽ gửi địa chỉ có đuôi "/H" đến ESP32 và ESP32 sẽ nhận dạng và kích hoạt LED và khi chúng ta click vào ngắt kết nối nó sẽ gửi địa chỉ có "/L" cuối cùng. Mỗi lần chúng tôi nhấp vào một trong các tùy chọn, ESP32 sẽ in trên màn hình nối tiếp xác nhận địa chỉ mà nó nhận được.

Đoạn mã này là nơi chúng tôi thực hiện việc xử lý các tình trạng sau:

```
114         // Verifique se o pedido do cliente foi "GET /H" ou "GET /L":
115         if (currentLine.endsWith("GET /H")) {
116           digitalWrite(5, HIGH);               // GET /H liga o LED
117         }
118         if (currentLine.endsWith("GET /L")) {
119           digitalWrite(5, LOW);                // GET /L desliga o LED
120         }
```

Hình 09: đoạn mã.

Xác nhận trả về các lệnh đã nhận:

```
16:09:03.497 -> New Client.
16:09:03.497 -> GET /L HTTP/1.1
16:09:03.497 -> Host: 192.168.100.134
16:09:03.497 -> Connection: keep-alive
16:09:03.497 -> Upgrade-Insecure-Requests: 1
16:09:03.497 -> User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
16:09:03.530 -> Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
16:09:03.530 -> Referer: http://192.168.100.134/H
16:09:03.530 -> Accept-Encoding: gzip, deflate
16:09:03.530 -> Accept-Language: pt-PT,pt;q=0.9,en-US;q=0.8,en;q=0.7
16:09:03.530 ->
16:09:03.530 -> Client Disconnected.
16:09:04.509 -> New Client.
16:09:04.509 -> GET /favicon.ico HTTP/1.1
16:09:04.509 -> Host: 192.168.100.134
16:09:04.509 -> Connection: keep-alive
16:09:04.509 -> User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
16:09:04.509 -> Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
16:09:04.556 -> Referer: http://192.168.100.134/L
16:09:04.556 -> Accept-Encoding: gzip, deflate
16:09:04.556 -> Accept-Language: pt-PT,pt;q=0.9,en-US;q=0.8,en;q=0.7
16:09:04.556 ->
16:09:04.556 -> Client Disconnected.
```

Hình 10: Lệnh trang.

Vì vậy, chúng tôi đã hoàn thành và bây giờ chúng tôi có thể thực hiện các dự án của mình với ESP32 yêu cầu Ethernet có dây.

4 - Tài liệu video

Chúng tôi cũng có một video về chủ đề này, xem nó ở đây!

5. Tài liệu tham khảo

https://sautter.com/blog/ethernet-on-esp32-using-lan8720/
https://github.com/espressif/arduino-esp32/issues/1938

Linkedin muốn Tác giả