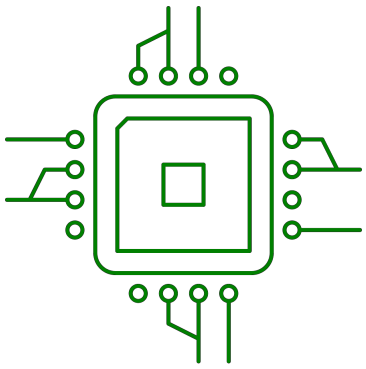

Manual Técnico: Sistema de Identificador de Cables a través de WiFi.

August 17, 2023

1 Introducción

El Sistema de IDENTIFICADOR DE CABLES a través de WiFi es una solución basada en Arduino y tecnología WiFi (ESP32) que permite controlar el encendido y apagado de switches de manera remota mediante una interfaz web. Este sistema se implementa utilizando un punto de acceso WiFi creado por el dispositivo Arduino y un servidor web que permite la interacción con las salidas a través de solicitudes HTTP.

2 Componentes Principales

2.1 Hardware

- Dispositivo Arduino compatible con WiFi (ESP32)
- Interface de potencia aislada para las salidas
- Cables de conexión

2.2 Software

- IDE de Arduino
- Librerías requeridas:
  - `WiFi.h`: Para gestionar la conectividad WiFi.
  - `DNSServer.h`: Para la configuración del servidor DNS.
  - `data.h` y `data1.h`: Archivos que contienen funciones y datos necesarios para la interfaz web.

3 Configuración del Hardware

Conecta los componentes siguiendo estas instrucciones:

1. Conecta el dispositivo Arduino (ESP32) a través de un cable USB a tu computadora.
2. Conecta la interfaz de potencia aislada a las salidas del sistema y asegúrate de conectar los pines a los puertos de potencia correctamente.
3. Verifica que los pines de las Salidas estén conectados según el arreglo `LEDPins` en el código.

4 Instalación del Software

Sigue estos pasos para instalar el software en el dispositivo Arduino:

1. Abre la IDE de Arduino en tu computadora.
2. Copia y pega el código proporcionado en una nueva ventana en la IDE.
3. Selecciona el modelo de dispositivo correcto en la pestaña `Herramientas` > `Placa`.
4. Selecciona el puerto COM correspondiente en la opción `Herramientas` > `Puerto`. O si usas Linux, muy probable este en el puerto /dev/ttyUSB0.
5. Haz clic en el botón `Subir` (ícono de flecha) para cargar el código en el dispositivo.

5 Configuración de Red

El sistema crea un punto de acceso WiFi con el nombre "KAME_HOUSE". Sigue estos pasos para conectarte:

1. En tu dispositivo (teléfono, tableta o computadora), busca las redes WiFi disponibles.
2. Selecciona y conéctate a la red WiFi "KAME_HOUSE".
3. Al conéctate a la red WiFi "KAME_HOUSE" te aparecera una ventana emergente, oprimela y te reedigira a la interface, o sino una vez conectado a la red vuelve a seleccionar la red misma para que te reedirecione.

6 Uso y Funcionalidad

Una vez configurado, el sistema te permitirá controlar las salidas a través de una interfaz web. Accede a la interfaz tocando el nombre de la red o tocando la pequeña ventana que te aparece cuando te conectas.

En la interfaz web podrás:

- Encender y apagar cada salida de manera individual.
- Controlar las salidas a distancia mediante solicitudes HTTP.

7 Solución de Problemas

Si encuentras problemas durante la instalación o uso del sistema, considera las siguientes acciones:

- Verifica las conexiones físicas de los componentes.
- Utiliza el Monitor Serie en la IDE de Arduino para depurar y observar mensajes.
- Asegúrate de estar conectado a la red WiFi "KAME_HOUSE".
- Consulta la documentación de las librerías utilizadas y recursos en línea.
- Reinicia el dispositivo Arduino usando el boton BOOT.

8 Cierre y Mantenimiento

Cuando hayas terminado de utilizar el Sistema de IDENTIFICADOR DE CABLES, simplemente desconecta el dispositivo de la fuente de alimentación. Para realizar mantenimiento, asegúrate de mantener las conexiones en buen estado y revisa periódicamente el funcionamiento de las salidas y la conectividad WiFi.

9 Notas Finales

Este manual técnico proporciona una guía básica para la instalación y configuración del Sistema de IDENTIFICADOR DE CABLES a través de WiFi. Para obtener más detalles técnicos y ajustar la configuración según tus necesidades, consulta la documentación de la IDE de Arduino, las librerías utilizadas y la especificación de tu dispositivo (ESP32).

Nota: Este manual es una referencia general y puede variar según las versiones de software y hardware utilizadas. Asegúrate de consultar la documentación y recursos actualizados según tu entorno.

10 Código Fuente y Descripción

A continuación se muestra el código fuente del proyecto junto con una breve descripción de su funcionamiento en los comentarios del mismo codigo.

```
#include <WiFi.h>
#include <DNSServer.h>
#include "data.h"
#include "data1.h"

// Configuracion del puerto DNS
const byte DNS_PORT = 53;

// Direccion IP del DNS predeterminado de Android
IPAddress apIP(8, 8, 4, 4);

// Objeto para manejar el servidor DNS
DNSServer dnsServer;

// Objeto para manejar el servidor WiFi
WiFiServer server(80);

// Pines de los Salidas
const int numLEDs = 7;
static unsigned short int LEDPins[numLEDs] = {2, 0, 4, 5,
    18, 19, 3}; // Definir pines de los Salidas

// Variables para manejar la comunicacion con el cliente
char linebuf[80]; // Bufer para almacenar caracteres
   recibidos
int charcount = 0; // Contador de caracteres recibidos

void setup() {
  Serial.begin(115200); // Iniciar la comunicacion serial

  // Configurar pines de los LEDs como salidas
  for (int i = 0; i < numLEDs; i++) {
    pinMode(LEDPins[i], OUTPUT);
  }

  WiFi.mode(WIFI_AP); // Configurar WiFi en modo de punto
      de acceso
  WiFi.softAP("KAME_HOUSE"); // Establecer el nombre de
     la red WiFi
```

```
36   WiFi.softAPConfig(apIP, apIP, IPAddress(255, 255, 255,
       0)); // Configurar la direccion IP del punto de
       acceso
37
38   dnsServer.start(DNS_PORT, "*", apIP); // Iniciar el
       servidor DNS
39   server.begin(); // Iniciar el servidor web en el puerto
        80
40 }
41
42 // Funcion para procesar solicitudes de control de Salida
     .
43 void processLEDRequest(const char* request, int ledIndex)
     {
44   if (strstr(linebuf, request) > 0) {
45     if (strstr(request, "on") != nullptr) {
46       digitalWrite(LEDPins[ledIndex], HIGH); // Encender
           el LED correspondiente
47       Serial.printf("LED %d ON\n", ledIndex + 1); //
           Imprimir en la consola
48     } else if (strstr(request, "off") != nullptr) {
49       digitalWrite(LEDPins[ledIndex], LOW); // Apagar el
           LED correspondiente
50       Serial.printf("LED %d OFF\n", ledIndex + 1); //
           Imprimir en la consola
51     }
52   }
53 }
54
55 // Funcion para manipular el estado de las Salidas segun
     las solicitudes HTTP/1.1
56 void manipulacionLed() {
57   for (int i = 1; i <= numLEDs; i++) {
58     char requestOn[10];
59     char requestOff[10];
60     sprintf(requestOn, "GET /on%d", i);
61     sprintf(requestOff, "GET /off%d", i);
62     processLEDRequest(requestOn, i - 1);
63     processLEDRequest(requestOff, i - 1);
64   }
65 }
66
67 void loop() {
```

```
68  dnsServer.processNextRequest(); // Procesar solicitudes
        DNS
69  WiFiClient client = server.available(); // Verificar si
        hay clientes conectados al servidor web
70
71  if (client) {
72    String currentLine = "";
73    boolean currentLineIsBlank = true;
74
75    while (client.connected()) {
76      if (client.available()) {
77        char c = client.read(); // Leer el siguiente
              caracter del cliente
78        Serial.write(c); // Enviar el caracter a traves
              de la comunicacion serial
79        linebuf[charcount] = c; // Almacenar el caracter
              en el bufer
80
81        if (charcount < sizeof(linebuf) - 1) charcount++;
              // Incrementar el contador de caracteres
82
83        if (c == '\n') { // Si se recibe un salto de
              linea
84          currentLineIsBlank = true;
85          if (currentLine.length() == 0) { // Si la linea
                actual esta en blanco
86            web(client); // Llamar a la funcion web desde
                  data.h
87            break; // Salir del bucle
88          } else {
89            currentLine = "";
90          }
91          currentLineIsBlank = true;
92
93          // Manipulacion de los LEDs HTTP/1.1
94          manipulacionLed();
95          memset(linebuf, 0, sizeof(linebuf)); // Borrar
                el bufer
96          charcount = 0; // Reiniciar el contador de
                caracteres
97        } else if (c != '\r') {
98          currentLine += c;
99          currentLineIsBlank = false;
```

```
        }
      }
    }
    delay(1);
    client.stop();
    Serial.println("Cliente desconectado");
  }
}
```

1: Código Fuente de Pagina Web

Se muestra parte del codigo de la pagina Web que se encuentra en el archivo "data.h", y sirve para la interface grafica del proyecto. Cabe señalar que el codigo web esta dividido en varias partes para la optimización de mememoria de la placa que contiene el ESP32. Cada parte de la pagina se declara como constante y como string. En este caso esta es la primera parte de la pagina web (interface de usuario), en el cual esta la cabecera del codigo html, y y el estilo de la pagina (CSS, "Cascading Style Sheets" o Hojas de Estilo en Cascada).

```
const String pagina =  R"====(
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="utf-8">
    <title>Identificador</title>
    <meta name="viewport" content="width=device-width,
       initial-scale=1.0">
    <style>
     body {
            background-image: linear-gradient(to top,#5
               a07f0, #d17bff);
            margin: 1%;
            padding: 9.5%;
            font-family: Impact, Haettenschweiler, 'Arial
                Narrow Bold', sans-serif;
        }
        input {
            color: white;
            font-weight: bold;
            margin-bottom: 20%;
            width: 100%;
        }
```

```
        p {
            color: black;
            font-weight: bold;
        }
        h1 {
            text-align: center;
            font-size: 24px;
            color: white;
            padding: 20px 0;
        }
        .container {
            color: white;
            width: 80%;
            height: inherit;
            display: flex;
            flex-wrap: wrap;
            justify-content: center;
            align-items: center;
            padding: 0%;
            margin: 12%;
            background-color: white;
            position: relative;
            border: 2px solid;
            border-radius: 20px;
            box-shadow: rgb(150,150,150) 5px 5px 20px;
        }
        .box {
            padding-left: 10px;
            text-align: center;
        }
        .box input {
                background-image: linear-gradient(to top
                   ,#186a3b,  #2ecc71);
                display: block;
                text-decoration: none;
                color: white;
                padding: 10px;
                border-radius: 15px;
                transition: background-color 0.3s;
        }
        .box input:active{
            background-image: linear-gradient(to top,#
               adb5bd,  #f8f9fa);
```

```
            color:black;

        }
@media (max-width: 767px) {
    button:focus {
      outline: none; /* Elimina el contorno
         predeterminado */
    }
  }
        @media (min-width: 768px) {
            .container {
                max-width: 600px;
            }
        }
        @media (min-width: 992px) {
            .container {
                max-width: 800px;
            }
        }
        @media (min-width: 1200px) {
            .container {
                max-width: 1000px;
            }
        }
        </style>
      </head>
      <body>
      <h1>interface IoT</h1>
      <div class="container">)====";
```

2: Código Fuente de Pagina Web.

En esta otra parte se aprecia las secciones de la caja donde se alojan los botones. Tembien estan el codigo Java Script que nos permite hacer por medio de un metodo GET hacer peticiones HTTP y el uso de la interfaz XMLHttpRequest.

```
const String pagina1 =  R"====( <div class="box">
        <p>SALIDA #)====";

const String pagina2 =  R"====(</p>
        <input type="button" id="botonEncender" onclick="
           botonEncender)====";
const String pagina3 =  R"====(()" value="Encender">
```

```
7         <input type="button" id="botonApagar"  onclick="
             botonApagar)====";
8 const String pagina4 =  R"====(()" value="Apagar">
9         </div>)====";
10 const String pagina5 =  R"====(</div>
11     <script>)====";
12 const String pagina6 =  R"====(function botonEncender)
    ====";
13
14 const String pagina7 =  R"====(){
15           consultaGET("on)====";
16 const String pagina8 =  R"====(");
17       }
18       function botonApagar)====";
19 const String pagina9 =  R"====(){
20           consultaGET("off)====";
21 const String pagina10 =  R"====(");
22       })====";
23 const String pagina11 =  R"====(function consultaGET(
    consulta){
24           const Http = new XMLHttpRequest();
25           console.log(`Consultando  ${consulta}`)
26           Http.open("GET", consulta);
27           Http.send();
28
29           Http.onreadystatechange = (e) => {
30           console.log(Http.status );
31           console.log(Http.responseText);
32           };
33       }
34         </script>
35         </body>
36         </html>)====";
```

3: Funcion de Impresion de pagina Web
Aqui se muestra la funcion de impresion que se encuantra en el archivo "data1.h" y que nos optimiza el codigo para hacer una secuencia de impresion de 12 botones (6 pares de on y off). label

A su vez tambien crea las funciones de codigo Java Script para cada boton corrspondiente. Asu vez la impresion del codigo se intercala con el numero del contador en "for", con esto obtenemos tanto el numero de salida en la interface, como la impresion de funcion para da uno de los 12 botones.

```
//Toma un par metro de tipo referencia a "WiFiClient", es decir "WiFiClient &client".
// El "&" en "WiFiClient" & indica que "client" es una referencia al objeto "WiFiClient".
// asi ya no se tiene que hacer un nuevo objeto .
// En otras palabras, la funci n puede modificar el objeto original y los cambios se
// reflejar n fuera de la funci n.
void web(WiFiClient &client)
{
  client.println("HTTP/1.1 200 OK");
  client.println("Content-type:text/html");
  client. println();
  client.print(pagina);
  for (int n = 1; n < 7; n++)
  {
    client.print(pagina1);
    client.print(n);
    client.print(pagina2);
    client.print(n);
    client.print(pagina3);
    client.print(n);
    client.print(pagina4);
  }
  client.print(pagina5);
  for (int i = 1; i < 7; i++)
  {
    client.print(pagina6);
    client.print(i);
    client.print(pagina7);
    client.print(i);
    client.print(pagina8);
    client.print(i);
    client.print(pagina9);
    client.print(i);
    client.print(pagina10);
  }
  client.print(pagina11);
}
```