Universidad de Guanajuato
División de Ciencias e Ingenierías

Práctica 8:

MC web server

Arquitectura de Microcontroladores

Kasandra Giselle Morales Marmolejo

Dafne Geraldine Colexcua Morales

Dharma Daniela Arellano Navarro

Miguel Ángel Gomez Flores

31 Octubre 2024

Profesor: Huetzin Aaron Pérez Olivas

MC web server

Arquitectura de Microcontroladores

A. *RESUMEN*

Para conocer y comprender el funcionamiento de servidor web se utilizó un circuito conformado por un ESP32 y un sensor de temperatura LM35, al hacer uso de la programación se logró visualizar los datos del sensor en el servidor web.

I. INTRODUCCIÓN

Un servidor web es un software que forma parte del servidor y tiene como misión principal devolver información (páginas) cuando recibe peticiones por parte de los usuarios, es decir, es el software que permite que los usuarios que quieren ver una página web en su navegador puedan hacerlo. [1]

En está práctica se implementará un código para programar el ESP32 y que reciba y almacene los datos obteneidos por un sensor, en esta ocasión se utilizó un sensor de temperatura, LM35, con estos datos obtenidos después dentro del código se crea un servidor web que nos va a mostrar los datos almacenados, este caso la temperatura.

II. ESTADO DEL ARTE

A. *Contexto*

Un servidor web es una plataforma computacional muy potente que resguarda datos para ser consultados por diversos usuarios. Cuenta con un software que entrega la información solicitada por visitantes o miembros de un grupo de trabajo, a través de un dispositivo conectado a la red. Aunque también puede ser definido como un hardware y como un software y tener ambas cualidades.

Almacena los archivos que forman parte de un sitio electrónico o de una base de datos. Sirve como transmisor de esta información de acuerdo con las demandas de los usuarios. Por lo tanto, se mantiene siempre activo para recibir las solicitudes de los usuarios, procesarlas y entregar resultados sin excepción; incluso si no encuentra dicha consulta manda el código 404, que significa que no existen los recursos solicitados en su acervo.

Su función inicia cuando un usuario escribe directamente una dirección web (URL) o un dominio. Luego, el servidor web se encarga de localizar el contenido y enviarlo al usuario para que pueda verlo en su dispositivo.

Así que funciona como un lugar de almacenamiento de diversos tipos de información, la cual puede constituirse por archivos HTML, CSS, JavaScript (incluidos imágenes, textos, videos o audios), y debe ser ubicada por los programas computacionales que hacen funcionar el sistema operativo.[3]

B. *Métodos Utilizados*

ESP32

El ESP32 es un microcontrolador de 32 bits y de doble núcleo, cuenta con WIFI y BLUETOOTH, es de montaje superficial SMD, tiene 32 pines.

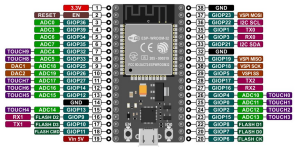

Fig. 1. ESP32 datasheet

El ESP32 con la librería Wifi.h nos posibilita trabajar de varias maneras diferentes. No solo la típica conexión al AP (punto de acceso) de tu router (como cualquier PC o móvil), sino que también permite una conexión directa entre ESP32 y los clientes sin pasar por el router, ni por su punto de acceso Wifi. [2]

Fig. 2. Ejemplo de una red doméstica típica con un ESP32 y diversos dispositivos

LM35

El LM35 es un circuito electrónico sensor que puede medir temperatura. Su salida es analógica, es decir, te proporciona un voltaje proporcional a la temperatura, para convertir el voltaje a la temperatura, proporciona 10mV por cada grado centígrado. Tiene un rango

desde 55°C a 150°C.

Este sensor no necesita de ningún circuito adicional para ser usado. Se alimenta directamente con una fuente de 5V y entrega una salida analógica entre 0V a 1.5V. Este voltaje analógico puede ser leído por el ADC de un microcontrolador o tarjeta de desarrollo como Arduino, ESP32, Raspberry Pi Pico, PICs y más.

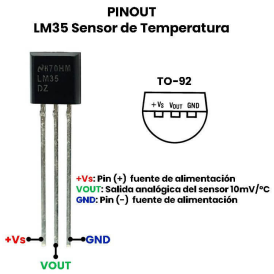

Fig. 3. LM35 datasheet

C. Aplicaciones

Crear un servidor web con un ESP32 tiene muchas aplicaciones útiles desde el hogar inteligente, donde permite controlar luces, termostatos y otros aparatos, hasta entornos industriales, donde facilita la monitorización y control de procesos, sus posibilidades son casi ilimitadas. Además, es una herramienta ideal para la creación de prototipos rápidos, la recopilación de datos de sensores y la construcción de interfaces de usuario personalizadas.

III. RESULTADOS

Objetivo:
Crear un circuito utilizando ESP32 y un LM35 para programar un servidor web que nos muestre la temeperatura registrada por

Práctica experimental:
Teniendo en cuenta la configuración de ESP32 se armo el siguiente circuito, tomando en cuenta que el sensor se necesito conectar en un GPIO que sea un convertidor de analógico a digital.

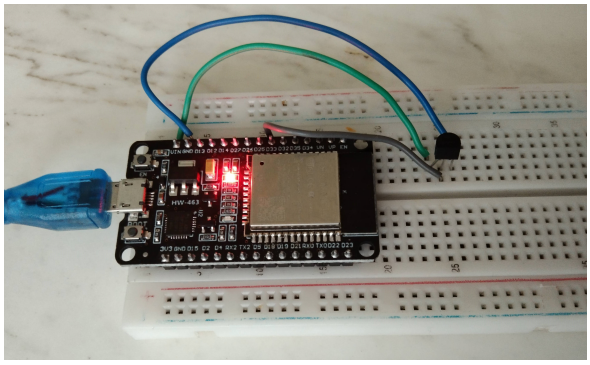

Fig. 4. Circuito

El código utilizado se puede desglosar en:

- Parte inicial
  Como se muestra en la Fig. 5 se incluye la libería WiFi la cual es necesaria para que el ESP32 se conecte a una red Wi-Fi. Se crea después un objeto de tipo WiFiServer y lo asigna al puerto 80, que es el puerto estándar para los servidores HTTP. Este objeto se utilizará para escuchar las conexiones entrantes y servir páginas web. Para después definir las credenciales de la red Wi-Fi a la que se conectará como lo es el nombre y la contraseña de la red
  Luego se define cual es el pin que se concetará la salida del sensor y después se crea una función para leer la temperatura del sesnor en grados Celsius.

```
#include <WiFi.h>

// Configuración del servidor Web en el puerto 80
WiFiServer server(80);

// Credenciales de WiFi
const char* ssid     = "INFINITUMF1AE";
const char* password = "EcFGaGG9ka";

// Configuración del sensor LM35
const int LM35_SENSOR_PIN = 33;

// Variables globales
int contconexion = 0;
String header; // Variable para guardar el HTTP request

// Función para leer la temperatura en grados Celsius
float leerTemperatura() {
  int lecturaADC = analogRead(LM35_SENSOR_PIN); // Lee el valor ADC del LM35
  float voltaje = lecturaADC * (3.3 / 4095.0);  // Convierte la lectura ADC a voltaje (3.3V para el ESP32)
  float temperatura = voltaje * 100.0;          // Convierte el voltaje a temperatura (10mV por grado Celsius)
  return temperatura;
}
```

Fig. 5. Código parte 1

Continua el código como se muestra en la Fig. 6 con la función que genera la página HTML para una página web simple. La página muestra la temperatura actual leída del sensor y denttro de este mismo se inidica que la página se recargue automáticamente cada 5 segundos para mostrar la temperatura actualizada.

```
// Página HTML
String generarPaginaHTML(float temperatura) {
  String pagina = "<!DOCTYPE html>"
                  "<html>"
                  "<head>"
                  "<meta charset='utf-8' />"
                  "<meta http-equiv='refresh' content='5' />"  // Actualiza la página cada 5 segundos
                  "<title>Servidor Web ESP32 - Temperatura</title>"
                  "</head>"
                  "<body>"
                  "<center>"
                  "<h1>Servidor Web ESP32</h1>"
                  "<p>Temperatura Actual: " + String(temperatura, 2) + " °C</p>"
                  "</center>"
                  "</body>"
                  "</html>";
  return pagina;
}
```

Fig. 6. Código parte 2

- Void setup
  En esta parte del código como se muestra en la Fig. 7 se configura el pin al que está conectado el sensor LM35 como una entrada digital. Se inicia también la conexión Wi-Fi ya que el esp32 se conecta a la red especificada anteriormente con el nombre y la contraseña.
  Se espera a que se estrablezca la conexión mediante un ciclo while que se ejecuta mientras la conexión no esta establecida.
  Si la conexión Wi-Fi se establecio se ejecita el código dentro de if e inicia el servidor web o manda un mensahe de error de conexión si no se ha establecido.

```
void setup() {
  Serial.begin(115200);

  // Configurar el pin del sensor LM35 como entrada
  pinMode(LM35_SENSOR_PIN, INPUT);

  // Conexión a la red WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED && contconexion < 50) {
    ++contconexion;
    delay(500);
    Serial.print(".");
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\nWiFi conectado");
    Serial.println(WiFi.localIP()); // Imprime la dirección IP del ESP32
    server.begin();                 // Inicia el servidor
  } else {
    Serial.println("\nError de conexion");
  }
}
```

Fig. 7. Código parte 3

- Void Loop
  En este apartado que muestra la Fig. 8 se busca que el cliente este conectado, si es asi se ejecuta el código dento del if donde se imprime un mensaje para notificar que el cliente esta conectado. También esta un ciclo while que se ejecuta si el cliente esta conectado, se encarga de leer los datos enviados por el cliente, analizarlos línea por línea y determinar cuándo se ha recibido una solicitud completa. Una vez que se ha identificado una solicitud completa, se genera la respuesta correspondiente y se envía al cliente.
  Se termina el ciclo y se cierra la conexión para después imprimir un mensaje notificando al cliente que se desconecto

```
void loop() {
  // Verifica si hay un cliente disponible
  WiFiClient client = server.available();
  if (client) { // Si se conecta un nuevo cliente
    Serial.println("Nuevo cliente conectado.");
    String currentLine = "";

    // Lee el request del cliente
    while (client.connected()) {
      if (client.available()) { // Si hay datos para leer
        char c = client.read();
        Serial.write(c); // Muestra el request en el monitor serie
        header += c;
        if (c == '\n') {
          // Si la nueva línea está en blanco, significa que es el fin
          if (currentLine.length() == 0) {
            // Lee la temperatura actual del LM35
            float temperatura = leerTemperatura();

            // Responde con los encabezados HTTP estándar
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();

            // Genera y envía la página HTML con la temperatura actual
            client.println(generarPaginaHTML(temperatura));

            // La respuesta HTTP termina con una línea en blanco
            client.println();
            break;
          } else {
            currentLine = "";
```

Fig. 8. Código parte 4

Y como resultado del código anterior se obtuvo en el monitor serial la dirección IP como se muestra en la Fig. 9

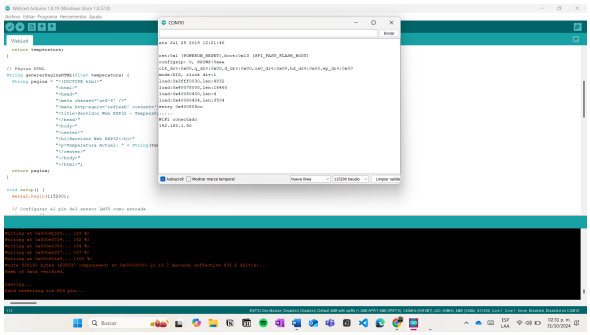

Fig. 9. Monitor serial

Al copiar la dirección IP en el navegador se visualizo la temepratura como se muestra en la Fig. 10

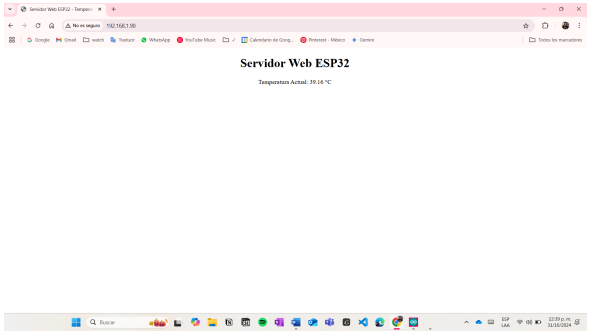

Fig. 10. Servidor Web

IV. CONCLUSIÓN

Durante el desarrollo de esta práctica se logró reforzar los conocimientos que se tenían del ESP32 de prácticas anteriores y del uso de sensores como lo fue el LM35

Se adquirio nuevo conocimiento sobre como conectar a WI-FI el ESP32, el código que se debe utilizar y como a través de este conexión poder obtener datos y visualizarlos en diferentes dispositivos solo teniendo la dirección IP, aunque el diseño del servidor web fue muy simple se cumplio el objetivo de visualizar los datos del sensor en la web.

Este proyecto ha demostrado el potencial del ESP32 como una plataforma versátil para la creación de dispositivos IoT. En futuras iteraciones, se podría explorar la integración de otros sensores como humedad, luz o movimiento, entre otros.

REFERENCES

[1] Martínez, G. (2023, August 17). ¿Qué es un servidor web y para qué sirve? - Webempresa. Webempresa. https://www.webempresa.com/hosting/que-es-servidor-web.html

[2] Pascual, C. (2022, April 5). Servidor web con ESP32. Programarfacil Arduino y Home Assistant. https://programarfacil.com/esp32/servidor-web-con-esp32/Queesunservidorwebycomofunciona

[3] Coppola, M. (2022, August 1). Qué es un servidor web, para qué sirve, cómo funciona y ejemplos. Hubspot.es. https://blog.hubspot.es/website/que-es-servidor-web