



**UNIVERSIDAD AUTÓNOMA
DE AGUASCALIENTES**

**CENTRO DE CIENCIAS BÁSICAS
DEPARTAMENTO DE SISTEMÁS
ELECTRÓNICOS INGENIERÍA EN
ELECTRÓNICA**

INTERNET DE LAS COSAS

**PROPIUESTA
ALCOHOLIMETRO WEARABLE**

Integrantes:

**ALVARADO LAGUNES MARÍA JOSÉ
FUANTOS DÍAZ JORGE EDUARDO
HIDALGO RODRÍGUEZ MELISSA
LLAMAS LLAMAS REYNALDO SEBASTIÁN
LÓPEZ ARRIAGA JUAN ANTONIO
CHAVIRA RAMOS JOSUE ALBERTO**

DOCENTE: LUIS ANTONIO RAYGOZA PEREZ

FECHA: 19 de junio de 2025

Esquemático

Para este proyecto que es el Alcoholímetro digital y que se implementó como wearable, se utilizó los siguientes materiales:

- ESP32 DEV KIT V1: Tarjeta de desarrollo y servidor de la página web
 - MAX30102: Pulsímetro y oxímetro, conectado mediante I2C.
 - MQ3: Sensor de gas, conectado mediante ADC.
 - Batería de Litio de 250 mAh: Dota al sistema de independencia para convertirlo en wearable.
 - Switch: Botón de encendido y apagado del sistema.
 - TP4056: Circuito integrado que servirá para cargar la batería

En la Figura 1 se muestra las conexiones de los componentes previamente mencionados.

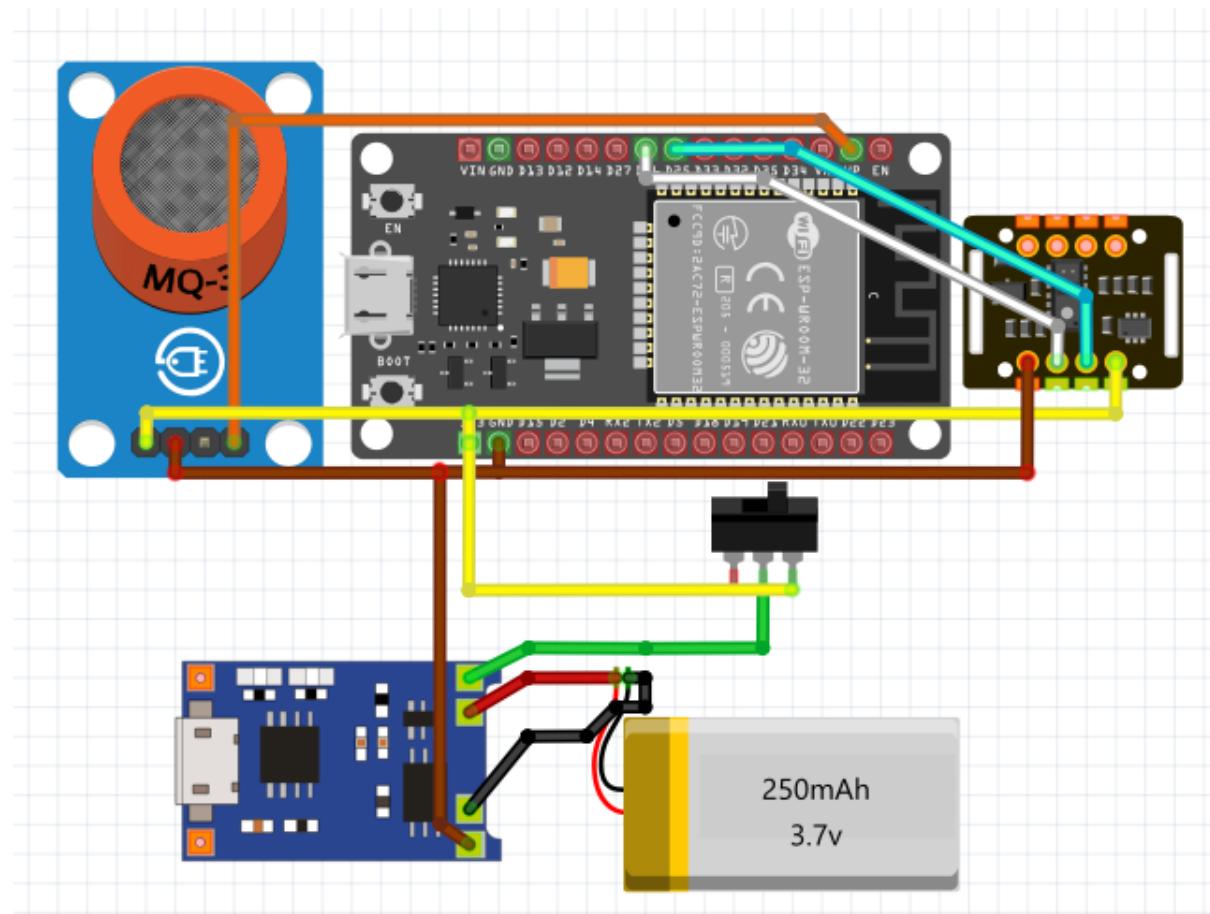


Figura 1. Conexiones de los componentes

La batería alimentará a la ESP32 con 3.3v, la cual distribuirá ese voltaje a los componentes MQ-3 y MAX30102, la batería será cargada a través del módulo TP4056. Para el consumo de energía se agregó un switch, que nos permitirá ahorrar energía cuando el wearable no se este utilizando.

Código

El código se implementó en Arduino IDE, ya que este nos proporciona librerías que facilitan el manejo de los módulos a utilizar y se ha utilizado para la implementación de prácticas pasadas en la materia. El código se encargará de transmitir los datos que captura de los sensores (MQ-3 y MAX30102) mediante WiFi a una página web, la cual mostrará la información para que el usuario pueda percibir su nivel de alcohol, pulso y oxigenación y que a su vez los datos sean guardados en la memoria caché del navegador, el protocolo de comunicación que se utilizó fue mediante un servidor asíncrono.

```
#include <WiFi.h>
#include <ESPAsyncWebServer.h>
#include <FS.h>
#include <SPIFFS.h>
#include <ArduinoJson.h>
#include <ESPmDNS.h>

#include "myMAX30102.h"
#include "myMQ3.h"

// ----- Pines -----
#ifndef ESP32C3
#define ESP32C3
#endif
const int pin_MQ3 = 0;
const int pin_SDA = 8;
const int pin_SCL = 9;
#else
const int pin_MQ3 = 36; // VP
const int pin_SDA = 25;
const int pin_SCL = 26;
#endif

// Credenciales de Wi-Fi
const char* ssid = "dlink_extension";
const char* password = "3YYHxxcCGm";
// El nombre pagina web
const char* host = "alcoholimetro"; // http://alcoholimetro.local
```

```

AsyncWebServer server(80);

void iniciarMedicionSensores();
volatile bool medicionEnCurso = false;
volatile float ultimaOxigenacion = 0.0;
volatile int ultimoPulso = 0;
volatile float ultimoAlcohol = 0.0;

// _____ setup()
void setup() {
    Serial.begin(115200);
    // _____ SPIFFS begin
    Serial.println("Verificando SPIFFS");
    if (!SPIFFS.begin()) {
        Serial.println("¡Error al montar SPIFFS!");
        //ESP.restart(); // Reinicia si SPIFFS falla
    }else Serial.println("SPIFFS montadas");
    // _____ WiFi begin
    WiFi.begin(ssid, password);
    Serial.print("Conectando a Wi-Fi");
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("✓ ¡Conectado exitosamente!");
    Serial.print("📶 IP local asignada: ");
    Serial.println(WiFi.localIP());
    // _____ mDNS
    if (!mDNS.begin(host)) {
        Serial.println("Error configurando el respondedor mDNS.");
        while(1) { // Detener si no se puede iniciar mDNS
            delay(1000);
        }
    }
    Serial.print("mDNS iniciado. Puedes acceder desde http://");
    Serial.print(host);
    Serial.println(".local");

    initMQ3();
    initMAX30102();

    // _____ Server begin (con AsyncWebServer)
}

```

```

// Manejar el archivo raíz (index.html)
server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/index.html", "text/html");
});

// Manejar style.css
server.on("/style.css", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/style.css", "text/css");
});

// Manejar script.js
server.on("/script.js", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/script.js", "application/javascript");
});

// Ruta para INICIAR la medición
server.on("/iniciar_medicion", HTTP_GET, [](AsyncWebServerRequest *request){
    if (!medicionEnCurso) { // Solo iniciar si no hay una medición en curso
        medicionEnCurso = true;
        request->send(200, "application/json", "{\"status\":\"iniciando\"}");
    } else {
        request->send(200, "application/json", "{\"status\":\"en_curso\"}");
    }
});

// Manejar la ruta para obtener datos de los sensores
// Ruta para OBTENER el ESTADO y los DATOS de la medición
server.on("/obtener_datos", HTTP_GET, [](AsyncWebServerRequest *request){
    StaticJsonDocument<200> jsonDocument;

    if (medicionEnCurso) {
        jsonDocument["status"] = "midiendo";
    } else {
        jsonDocument["status"] = "completado";
        jsonDocument["oxigenacion"] = ultimaOxigenacion;
        jsonDocument["pulso"] = ultimoPulso;
        jsonDocument["alcohol"] = ultimoAlcohol;
    }

    String jsonString;
    serializeJson(jsonDocument, jsonString);
    request->send(200, "application/json", jsonString);
});

// Manejar rutas no encontradas
server.onNotFound([](AsyncWebServerRequest *request){

```

```

    request->send(404, "text/plain", "404: Not found");
});

// Manejar rutas no encontradas
server.onNotFound(notFound);

server.begin();
Serial.println("Servidor web asíncrono iniciado");
}

void iniciarMedicionSensores() {
    Serial.println("Medición iniciada...");
    ultimaOxigenacion = leerSp02();
    ultimoPulso = leerPulso();
    ultimoAlcohol = leerAlcohol();
    medicionEnCurso = false;
    Serial.println("Mediciones terminadas");

}

// loop()
void loop() {
    if(medicionEnCurso)
        iniciarMedicionSensores();
}

// Función para manejar rutas no encontradas
void notFound(AsyncWebServerRequest *request) {
    request->send(404, "text/plain", "404: Not found");
}

```

Capturas

En la Figura 2 se muestra el circuito en físico del wearable donde están todos los componentes acomodados e interconectados para su funcionamiento y en la Figura 3 se muestra el wearable ya con el circuito físico en su interior.

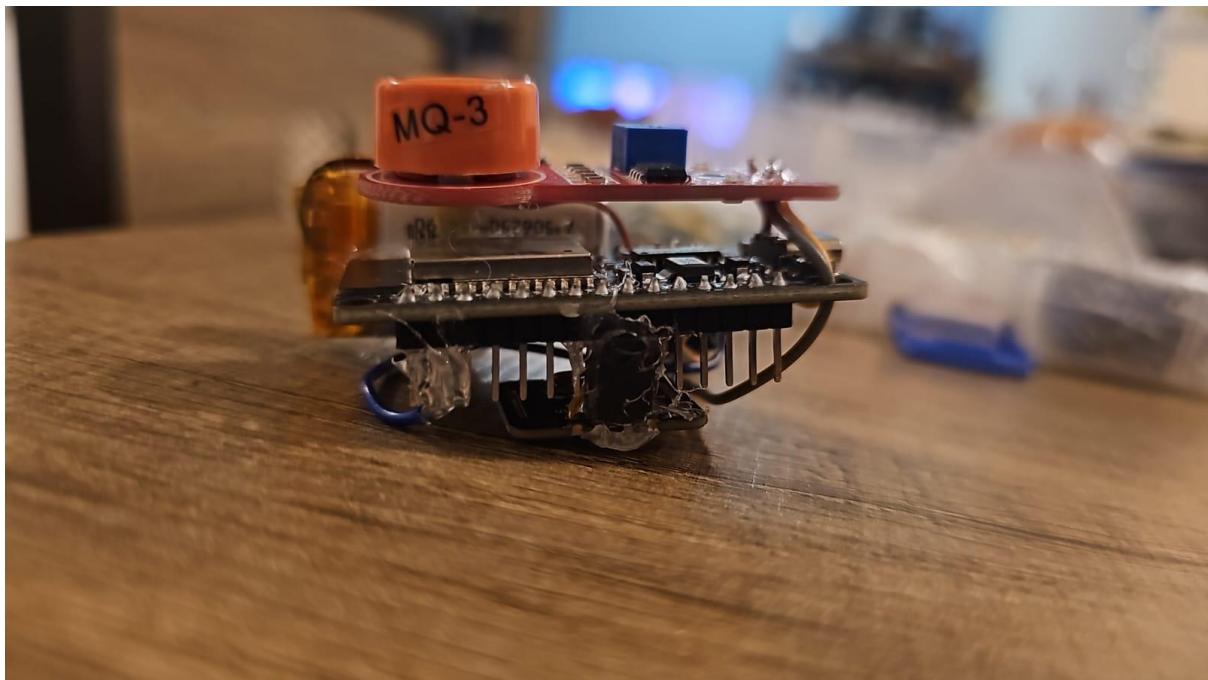


Figura 2. Circuito en físico del wearable



Figura 3. Wearable

En la Figura 3 se muestra la página web donde se podrá percibir la información de los sensores como lo es la oxigenación, pulso y nivel de alcohol, además te muestra un registro o historial de las últimas mediciones con fecha y hora.

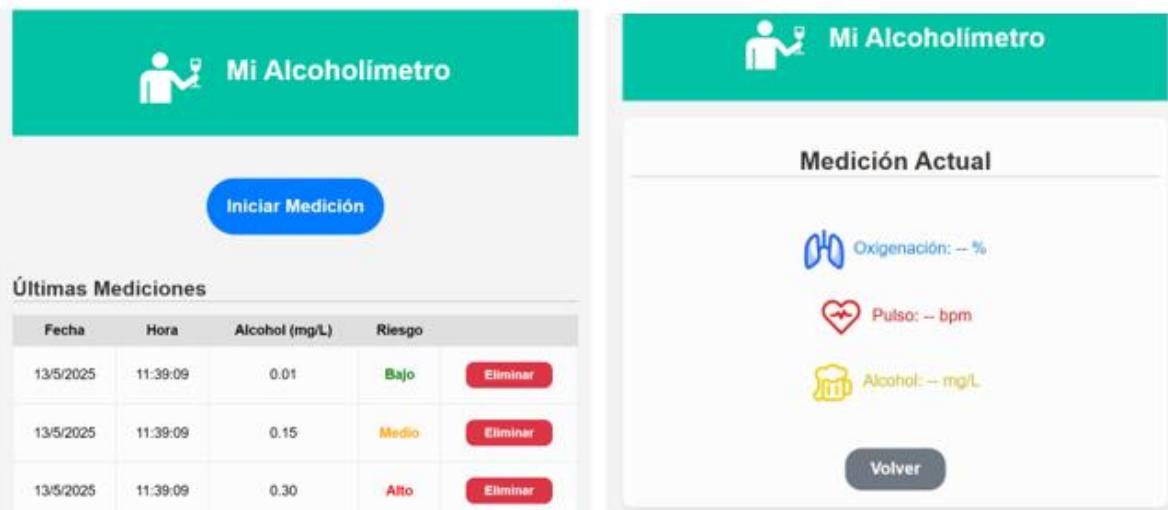


Figura 4. Página web del wearable

Conclusiones

Josue Alberto Chavira Ramos: Este proyecto nos sirvió para implementar los conocimientos adquiridos durante el semestre en la clase de internet de las cosas, ya que utilizamos herramientas como lo es la ESP32 que nos permitió transmitir información de los sensores y poder mostrarlos desde una página web a través de wifi, además de la implementación del Arduino IDE, ya que este nos facilitó el uso de las librerías para los sensores y su correcto funcionamiento. Estos conocimientos nos servirán para futuros proyectos, en los que tengamos que realizar alguna comunicación o muestra de datos por internet.

María José Alvarado Lagunes: Mediante la utilización de una ESP32 y tecnologías como el servidor web, se logró diseñar un dispositivo portátil capaz de medir niveles de alcohol, oxigenación y pulso, mostrando esta información en tiempo real a través de una página web. Este proyecto demuestra una implementación concreta de IoT en el ámbito de la salud y la seguridad personal, al permitir la supervisión no invasiva de parámetros biométricos críticos desde cualquier dispositivo conectado. Además, refuerza la importancia del diseño de sistemas embebidos energéticamente eficientes y la integración de hardware y software para aplicaciones modernas.

Reynaldo Sebastián Llamas Llamas: Gracias a las herramientas fáciles de usar en la esp32, este proyecto permitió hacer uso de IoT para la construcción de un wearable que permite obtener una lectura de la cantidad de alcohol que una persona consumió, así como la oxigenación y su pulso. Esto mediante una página web a la que se suben los resultados y la persona puede acceder a ellos en

cualquier momento. Este claro ejemplo de IoT, donde se puede acceder a la página y obtener datos obtenidos por sensores, es solo una de las aplicaciones para las que es útil el IoT. Así, se puede implementar fácilmente sistemas de monitorización e incluso de control en aplicaciones como domótica (para proyectos personales) o incluso en proyectos grandes de ciudades inteligentes.

Melissa Hidalgo Rodríguez: Este proyecto fue una excelente oportunidad para consolidar los conocimientos de IoT, el uso de la ESP32 fue esencial para establecer el servidor web asíncrono y el uso de los sensores MQ3 y MAX30102 fueron esenciales para recolectar los datos de alcohol y pulso. La implementación de un dispositivo wearable autónomo, alimentado por una batería destaca la importancia del diseño eficiente para aplicaciones IoT portátiles, además de mostrar la potencia de la tecnología IoT para proyectos centrados en monitorización de la salud y la seguridad personal.

Jorge Eduardo Fuants Díaz: La integración del Internet de las cosas y los sistemas embebidos pueden dar lugar a la elaboración de herramientas útiles para usuarios que lo requieran, en este caso, el dispositivo que se realizó fue con finalidades útiles para la seguridad y monitoreo de los signos vitales, tales como la oxigenación y la frecuencia cardiaca. Después de todo el desarrollo del proyecto podemos destacar las grandes ventajas que puede ofrecer el adquirir conocimientos sobre el internet de las cosas, el manejo de sensores y dispositivos electrónicos.

Juan Antonio López Arriaga: El desarrollo de este proyecto representó una oportunidad para aplicar conceptos fundamentales de sistemas embebidos y comunicación inalámbrica, integrando sensores biométricos con una ESP32 para el envío de datos en tiempo real a través de una red Wi-Fi. Se implementó un servidor web asíncrono capaz de mostrar lecturas de los sensores directamente en una interfaz accesible desde cualquier navegador, lo que implicó el manejo eficiente de recursos limitados y la optimización del código en el entorno del Arduino IDE. También se trabajó con protocolos de comunicación digital como I2C y UART, además de la gestión de entradas analógicas y digitales.