

**UNIDAD ACADÉMICA MULTIDISIPLINARIA MANTE**

**PROGRAMACION DE PUERTOS E INTERFACES**

**PROYECTO**

**“Medidor de Latidos del Corazón y Temperatura Corporal con ESP32”**

**Equipo:**

**Borjas Arias Ronaldo Antonio**

**Hernández Naccud Milton Arad**

**Pesina Santander Edgar Gael**

**Ramos Antonio Saqueo**

**Vázquez Romo Marco Antonio**

**Profesor:**

**López Piña Daniel**

**Cd. Mante, Tamaulipas Abril,2025**

**Medidor de Latidos del Corazón y Temperatura Corporal con ESP32**

El presente proyecto tiene como finalidad desarrollar un sistema capaz de medir la **frecuencia cardíaca** y la **temperatura corporal** utilizando una placa **ESP32**, sensores biométricos, y una pantalla para la visualización de datos. El objetivo es proporcionar una solución accesible y eficiente para el monitoreo de signos vitales en tiempo real, con posibilidad de extender sus funcionalidades a la nube o almacenamiento local.

**2. Objetivos del Proyecto**

**Objetivo General**

Diseñar e implementar un dispositivo portátil que mida y muestre la frecuencia cardíaca y la temperatura corporal en tiempo real utilizando un microcontrolador ESP32.

**Objetivos Específicos**

* Medir la frecuencia cardíaca mediante el sensor KY-039.
* Medir la temperatura corporal con el sensor digital DS18B20.
* Visualizar los datos recolectados a través de una pantalla conectada al ESP32.
* Explorar posibles mejoras como notificaciones, almacenamiento y conectividad.

**3. Componentes Utilizados**

| **Componente** | **Función Principal** |
| --- | --- |
| **ESP32** | Microcontrolador principal del sistema |
| **KY-039** | Sensor para la medición de los latidos del corazón |
| **DS18B20** | Sensor digital para medir la temperatura corporal |
| **Pantalla OLED** | Visualización de BPM y temperatura en tiempo real |

**4. Funcionamiento General**

El sistema recoge datos en tiempo real desde los sensores biométricos conectados al ESP32. La información obtenida se muestra en una pantalla, permitiendo observar tanto los **latidos por minuto (BPM)** como la **temperatura corporal actual**.

**5. Funcionalidades Adicionales (Opcionales)**

* **Alertas inteligentes:** Notificaciones si los valores están fuera del rango normal.
* **Registro de datos:** Almacenamiento de datos en una base de datos local o en la nube.
* **Modo offline:** Permite guardar los datos en la memoria interna del ESP32 cuando no hay conexión, y subirlos una vez restablecida.

**6. Aplicaciones del Proyecto**

* Monitoreo personal de salud en el hogar.
* Evaluación rápida en entornos clínicos o deportivos.
* Proyectos educativos sobre IoT y sensores biomédicos.

**7. Conclusiones**

Este proyecto representa una herramienta accesible y funcional para el monitoreo básico de la salud. Su implementación con ESP32 lo hace ideal para entornos domésticos o de aprendizaje. La posibilidad de ampliarlo con conectividad a la nube y funcionalidades de alerta lo posiciona como una base excelente para desarrollos más avanzados en el campo de la telemedicina y el IoT biomédico.

**8.Codigo**

#include <OneWire.h>

#include <DallasTemperature.h>

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

// --- Configuración pantalla OLED ---

#define SCREEN\_WIDTH 128

#define SCREEN\_HEIGHT 64

#define OLED\_RESET -1

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RESET);

// --- Pines sensores ---

#define HEARTBEAT\_SENSOR\_PIN 2  // KY-039 (mejor A0 o GPIO34)

#define ONE\_WIRE\_BUS 4          // DS18B20

OneWire oneWire(ONE\_WIRE\_BUS);

DallasTemperature sensors(&oneWire);

#define samp\_siz 4

float reads[samp\_siz], sum;

float last, reader, start;

float before = 0;

int n = 0;

long int ptr = 0;

long int last\_beat = 0;   // Último latido detectado

float bpm = 0.0;          // BPM calculado

int threshold = 500;      // Umbral mínimo para detectar un latido

// Filtro pasa bajos

float factor = 0.75;

float maximo = 0.0;

int minimoEntreLatidos = 300;

float valorAnterior = 500;

int latidos = 0;

void setup() {

  Serial.begin(115200);

  sensors.begin();

  pinMode(13, OUTPUT);

  for (int i = 0; i < samp\_siz; i++) reads[i] = 0;

  sum = 0;

  ptr = 0;

  Serial.println("Iniciando mediciones");

  // --- Inicializar pantalla OLED ---

  if (!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C)) {

    Serial.println("No se encontró pantalla OLED");

    while (true); // Detener ejecución si falla OLED

  }

  display.clearDisplay();

  display.setTextSize(1);

  display.setTextColor(SSD1306\_WHITE);

  display.setCursor(0, 0);

  display.println("Iniciando...");

  display.display();

  delay(1000);

}

void loop() {

  n = 0;

  start = millis();

  reader = 0.;

  do {

    reader += analogRead(HEARTBEAT\_SENSOR\_PIN);

    n++;

  } while (millis() < start + 20);

  reader /= n;

  sum -= reads[ptr];

  sum += reader;

  reads[ptr] = reader;

  last = sum / samp\_siz;

  float valorFiltrado = factor \* valorAnterior + (1 - factor) \* reader;

  float cambio = valorFiltrado - valorAnterior;

  valorAnterior = valorFiltrado;

  if ((cambio >= maximo) && (millis() > last\_beat + minimoEntreLatidos)) {

    maximo = cambio;

    digitalWrite(13, HIGH);

    last\_beat = millis();

    latidos++;

  } else {

    digitalWrite(13, LOW);

  }

  maximo = maximo \* 0.97;

  if (millis() >= last\_beat + 1000) {

    bpm = latidos \* 4;

    latidos = 0;

    last\_beat = millis();

  }

  sensors.requestTemperatures();

  float tempC = sensors.getTempCByIndex(0);

  Serial.print("Ritmo cardíaco: ");

  Serial.print(last);

  Serial.print(" BPM: ");

  Serial.print(60000 / last);

  Serial.print(" Temperatura: ");

  Serial.println(tempC);

  // --- Mostrar en pantalla OLED ---

  display.clearDisplay();

  display.setTextSize(2);

  display.setCursor(0, 0);

  display.print("BPM:");

  display.println((int)60000/last);

  display.setTextSize(2);

  display.setCursor(0, 30);

  display.print("T:");

  display.print(tempC, 1);

  display.cp437(true);

  display.write(167); // símbolo °

  display.print("C");

  display.display();

  delay(1000);

}

**9.Evidencias**



