INSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERIA MECANICA Y ELECTRICA

UNIDAD ZACATENCO

PROYECTO

MICROPROCESADORES

“Sensor atmosférico con ubicación GPS usando UDP”

Profesor:

GALICIA GALICIA ROBERTO

Alumno:

SALAZAR CERVANTES EDGAR YAEL

Introducción:

Introducción del Proyecto " Sensor atmosférico con ubicación GPS usando UDP " (códigos: main, node e index)

El proyecto " Sensor atmosférico con ubicación GPS usando UDP " es una aplicación que combina la lectura de datos de un sensor BME280, que proporciona información sobre la temperatura, presión y humedad ambientales, con la recepción de datos de un módulo GPS, que brinda la ubicación geográfica en términos de latitud y longitud. Estos datos se transmiten de forma inalámbrica a través de una conexión UDP y se visualizan en tiempo real en una interfaz web.

El código principal del proyecto se encuentra en el archivo "main.py". Este código se ejecuta en un microcontrolador y se encarga de leer los valores del sensor BME280, obtener los datos del módulo GPS y enviarlos a través de una conexión UDP a una dirección IP y puerto específicos. Utiliza las bibliotecas "machine", "Pin", "I2C", "UART", "time", "bme280", "socket", "network" y "micropyGPS" para interactuar con los periféricos y establecer la comunicación.

El código "node.js" en el archivo "node.js" actúa como un servidor Node.js que recibe los datos UDP del microcontrolador y los envía a los clientes conectados a través de WebSockets. Utiliza las bibliotecas "dgram", "http", "fs" y "socket.io" para manejar la comunicación UDP y la comunicación en tiempo real basada en WebSocket.

La interfaz web se encuentra en el archivo "index.html". Esta página HTML muestra los datos de temperatura, presión, humedad, latitud y longitud en tiempo real. Utiliza la biblioteca "socket.io" en el cliente para recibir los datos del servidor y actualizar la interfaz de usuario en consecuencia.

Para ejecutar el proyecto, se debe cargar el código "main.py" en el microcontrolador, iniciar el servidor Node.js ejecutando el código "node.js" y acceder a la interfaz web abriendo el archivo "index.html" en un navegador web compatible.

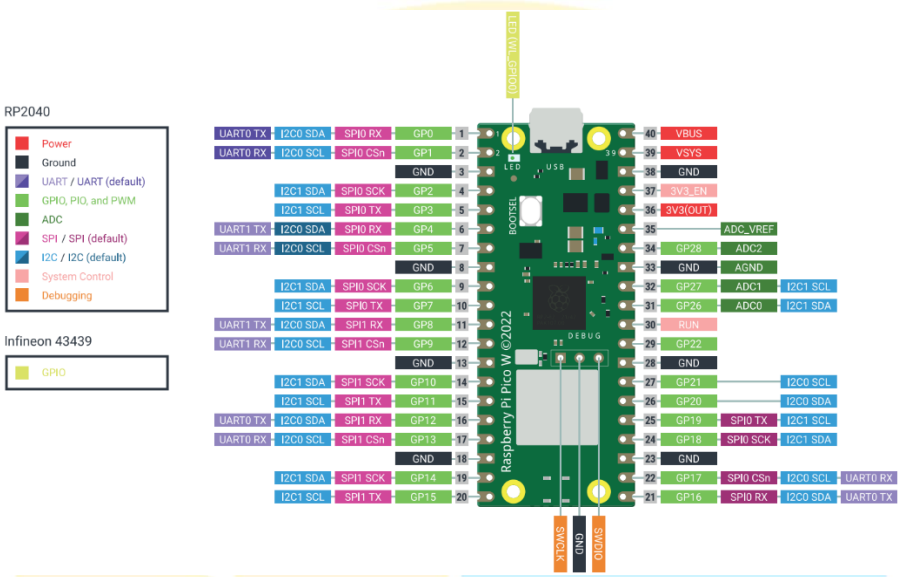
El proyecto " Sensor atmosférico con ubicación GPS usando UDP " tiene diversas aplicaciones, como el monitoreo ambiental, seguimiento de vehículos y análisis geoespacial. Proporciona una solución escalable y flexible para obtener datos ambientales y geográficos en tiempo real.

¡Con este proyecto, podrás obtener y visualizar datos ambientales y de ubicación de manera efectiva y en tiempo real!

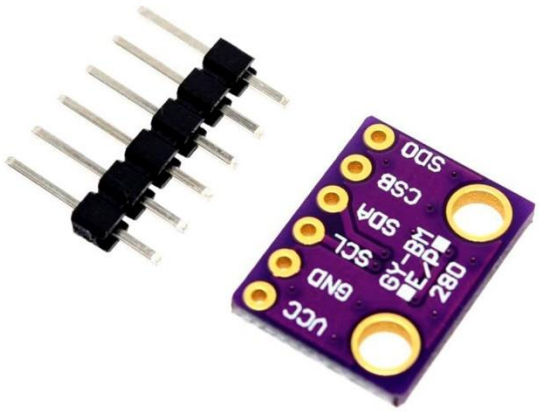
A continuación, pasaremos con la presentación de los materiales utilizados en el proyecto, así como el código correspondiente:

Para el código main los materiales son:

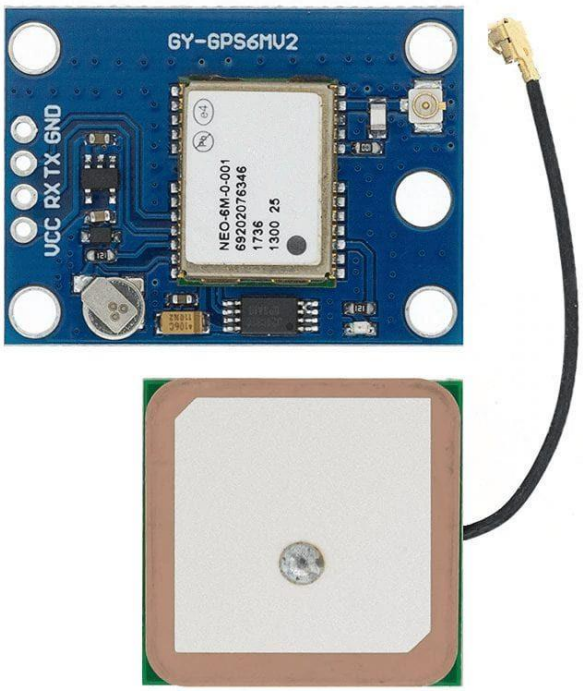
- Raspberry Pi Pico W



- BME280 BMP280 Sensor Digital de humedad temperatura módulo con Sensor de presión barométrica I2C



- GY-NEO6MV2 NEO-6M Módulo controlador de vuelo GPS 3V-5V con antena de cerámica súper fuerte



Y el código correspondiente al main es el siguiente:

"""

main.py

Este programa realiza la lectura de datos de un sensor BME280 y la información de GPS. Luego, envía los datos a través de UDP a una dirección IP y puerto específicos.

Requiere las siguientes bibliotecas y módulos:

- machine: módulo para controlar periféricos en microcontroladores.
- Pin: módulo para controlar pines de E/S en microcontroladores.
- I2C: módulo para comunicación I2C en microcontroladores.
- UART: módulo para comunicación UART en microcontroladores.
- time: módulo para funciones relacionadas con el tiempo.
- bme280: módulo para interactuar con el sensor BME280.

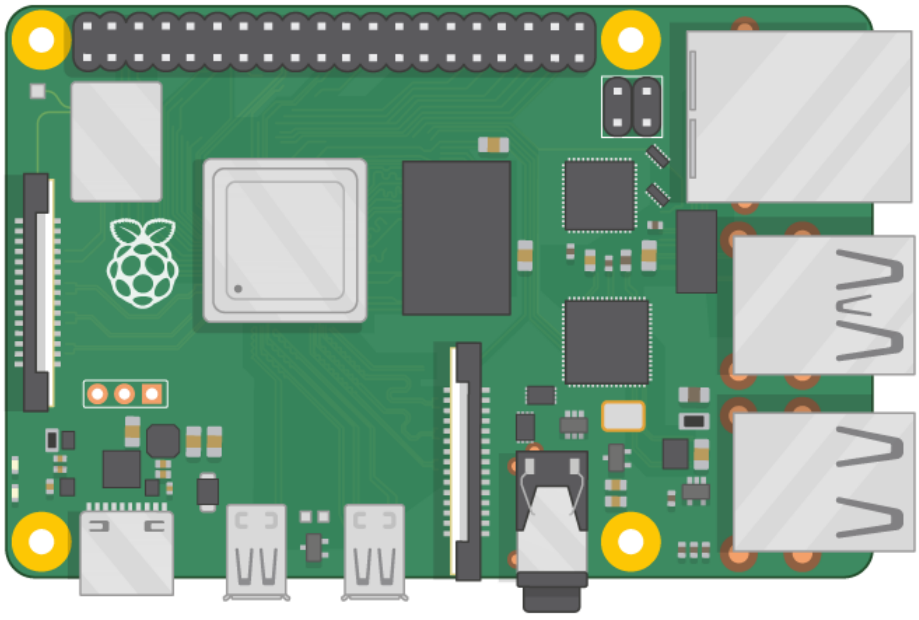
```
- socket: módulo para comunicación de red a través de sockets.
- network: módulo para configuración y administración de redes.
- micropyGPS: módulo para decodificar datos NMEA de un módulo GPS.

"""

from machine import Timer
from machine import Pin, I2C, UART
import time
import bme280
import socket
import network
from micropyGPS import MicropyGPS

# Configura los detalles de la conexión UDP
UDP_IP = '0.0.0.0'  # Dirección IP de destino para enviar los datos por UDP
UDP_PORT = 5005          # Puerto UDP de destino

# Configura los detalles de la red Wi-Fi
SSID = 'NOMBRE_DE_RED'    # Nombre de la red Wi-Fi (SSID)
PASSWORD = 'CONTRASEÑA'   # Contraseña de la red Wi-Fi

# Inicializa el adaptador Wi-Fi
wifi = network.WLAN(network.STA_IF)
wifi.active(True)

# Conéctate a la red Wi-Fi
wifi.connect(SSID, PASSWORD)

# Espera a que se establezca la conexión Wi-Fi
while not wifi.isconnected():
    pass
```

```
# Muestra la dirección IP asignada
print("Conexión Wi-Fi establecida")
print("Dirección IP:", wifi.ifconfig()[0])

# Crea un socket UDP
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Configuración del bus I2C
i2c = I2C(0, sda=Pin(0), scl=Pin(1), freq=400000)
bme = bme280.BME280(i2c=i2c)

# Configuración del GPS
gps_module = UART(1, baudrate=9600, tx=Pin(4), rx=Pin(5))
my_gps = MicropyGPS(5)  # Ajusta el valor de TIMEZONE según tu ubicación

# Valores iniciales de latitud y longitud (valores constantes)
initial_latitude = "0.000000"
initial_longitude = "0.000000"

def convert(parts):
    """
    Convierte las partes de coordenadas GPS (grados, minutos, dirección) a un valor decimal.

    Args:
        parts (tuple): Tupla con las partes de la coordenada (grados, minutos, dirección).

    Returns:
        str: Coordenada convertida a valor decimal con 6 decimales.

    """
```

```
    if parts[0] == 0:
        return None

    data = parts[0] + (parts[1] / 60.0)
    # parts[2] contiene 'E' o 'W' o 'N' o 'S'
    if parts[2] == 'S':
        data = -data
    if parts[2] == 'W':
        data = -data

    data = '{0:.6f}'.format(data)  # con 6 decimales
    return str(data)

def leer_sensor():
    """
    Lee los valores del sensor BME280 y envía los datos junto con la
información del GPS por UDP.

    """
    temp = bme.values[0]
    pres = bme.values[1]
    hum = bme.values[2]

    # Leer datos del GPS
    length = gps_module.any()
    if length > 0:
        b = gps_module.read(length)
        for x in b:
            msg = my_gps.update(chr(x))

    # Obtener latitud y longitud
    latitude = convert(my_gps.latitude)
    longitude = convert(my_gps.longitude)
```

```
    if latitude is None or longitude is None:
        # Usar los valores iniciales si no hay datos del GPS
        latitude = initial_latitude
        longitude = initial_longitude

    message = f"Temperatura: {temp}  Presion: {pres}  Humedad:
{hum}  Latitud: {latitude}  Longitud: {longitude}"
    print(message)
    sock.sendto(bytes(message, "utf-8"), (UDP_IP, UDP_PORT))

while True:
    leer_sensor()
    time.sleep(1)
```

Para el uso del Node y del index usaremos una raspberry pi 4:



Código del Node:

```
/**
 * node.js
 *
 * Este programa es un servidor Node.js que recibe datos UDP y los envía a través de WebSocket a clientes conectados.
 * También sirve un archivo HTML estático en respuesta a las solicitudes HTTP.
 *
 * Requiere las siguientes bibliotecas y módulos:
 * - dgram: módulo para la comunicación UDP en Node.js.
 * - http: módulo para crear un servidor HTTP en Node.js.
 * - fs: módulo para leer archivos del sistema de archivos en Node.js.
 * - socket.io: módulo para la comunicación en tiempo real basada en WebSocket.
 */

const dgram = require('dgram');
const http = require('http').createServer(handler);
const fs = require('fs');
const io = require('socket.io')(http);

const UDP_PORT = 5005;

// Crea un socket UDP
const server = dgram.createSocket('udp4');

// Función que se ejecuta cuando se recibe un mensaje UDP
server.on('message', (message, remote) => {
  console.log('Recibido:', message.toString());
  io.sockets.emit('data', parseData(message.toString()));
});
```

```
// Enlaza el socket UDP al puerto especificado
server.bind(UDP_PORT, () => {
  console.log('Esperando por datos UDP...');
});

/**
 * Parsea los datos recibidos en formato de cadena y los devuelve como un objeto.
 *
 * @param {string} data - Los datos recibidos en formato de cadena.
 * @returns {object} - Los datos parseados como un objeto con las siguientes propiedades: temp, pressure, humidity, latitude, longitude.
 */
function parseData(data) {
  const [temp, pressure, humidity, latitude, longitude] = data.split(' ');

  return {
    temp: temp.split(':')[1].trim(),
    pressure: pressure.split(':')[1].trim(),
    humidity: humidity.split(':')[1].trim(),
    latitude: latitude.split(':')[1].trim(),
    longitude: longitude.split(':')[1].trim(),
  };
}

/**
 * Manejador para las solicitudes HTTP. Sirve un archivo HTML estático.
 *
 * @param {object} req - El objeto de solicitud HTTP.
```

```
 * @param {object} res - El objeto de respuesta HTTP.
 */
function handler(req, res) {
  fs.readFile(__dirname + '/public/index.html', function(err, data) {
    if (err) {
      res.writeHead(404, { 'Content-Type': 'text/html' });
      return res.end("404 Not Found");
    }
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.write(data);
    return res.end();
  });
}

// Evento para cuando un cliente se conecta a través de WebSocket
io.sockets.on('connection', function(socket) {
  console.log('Cliente conectado');
});

// Inicia el servidor HTTP y escucha en el puerto 8080
http.listen(8080, function() {
  console.log('Servidor HTTP escuchando en el puerto 8080');
});
```

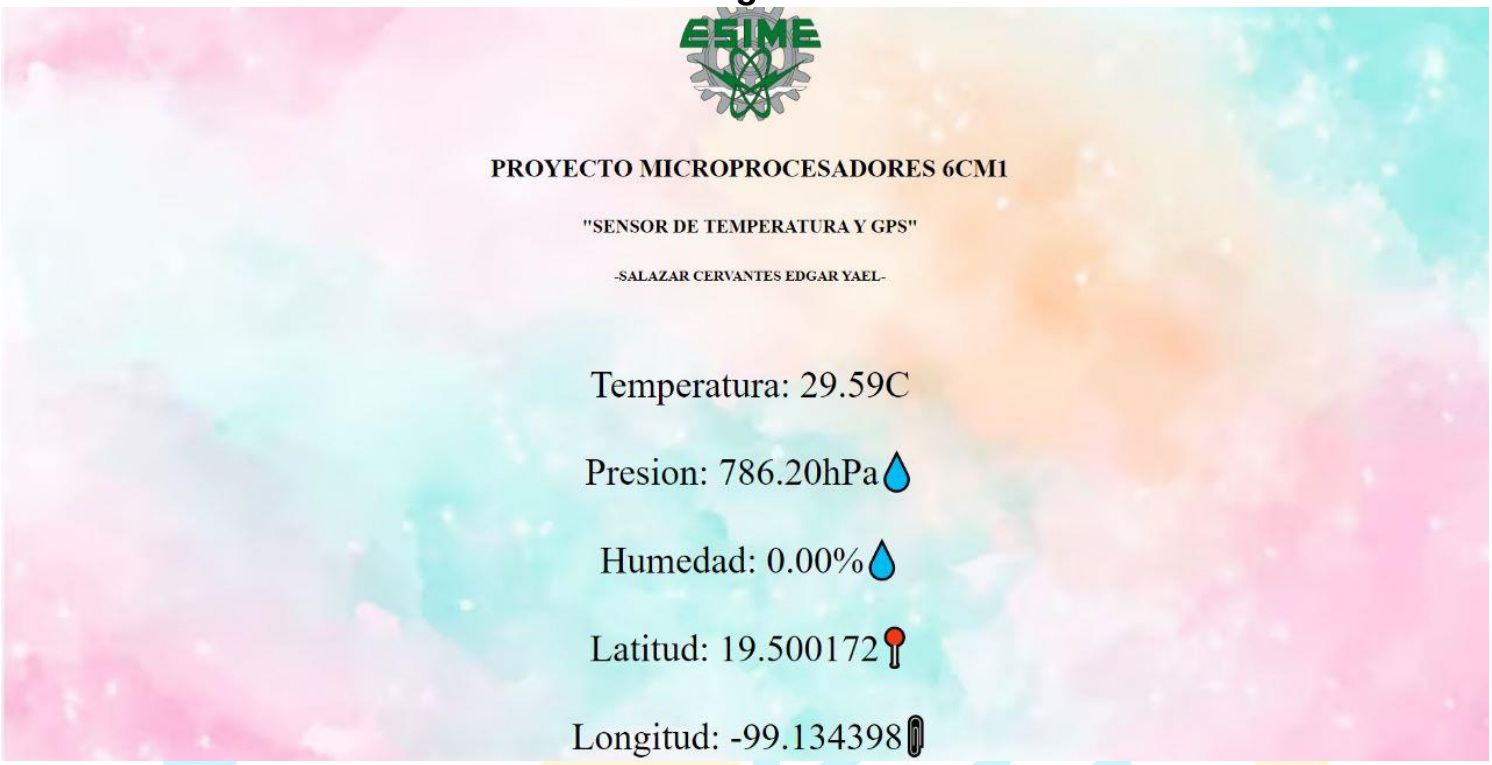
Finalmente, el código del Index:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>6CM1</title>
    <style>
        /* Estilos CSS para la página */
        body {
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            height: 100vh;
            margin: 0;
            padding: 0;
            background-image: url('https://img.freepik.com/vector-
premium/fondo-acuarela-colores_521516-1.jpg');
            background-size: cover;
            background-position: center;
            background-repeat: no-repeat;
        }

        .logo-esime {
            width: 200px;
        }

        h1, h2, h3 {
            text-align: center;
        }
```

```
        p {
            text-align: center;
        }
    </style>
</head>
<body>
    <img class="logo-esime"
src="https://www.esimetic.ipn.mx/assets/files/esimetic/img/conoc
enos/LogoEsime.png" alt="Logo ESIME">
    <h1>PROYECTO MICROPROCESADORES 6CM1</h1>
    <h2>"SENSOR DE TEMPERATURA Y GPS"</h2>
    <h3>-SALAZAR CERVANTES EDGAR YAEL-</h3>

    <p></p>
    <font size="7">
        <!-- Mostrar los datos recibidos -->
        <p id="temp">
            <span id="temp-symbol">&#8451;</span> <!-- Símbolo
para la temperatura -->
        </p>
        <p id="pressure">
            <span id="pressure-symbol">&#128167;</span> <!--
Símbolo para la presión -->
        </p>
        <p id="humidity">
            <span id="humidity-symbol">&#128167;</span> <!--
Símbolo para la humedad -->
        </p>
        <p id="latitude">
            <span id="latitude-symbol">&#128205;</span> <!--
Símbolo para la latitud -->
        </p>
        <p id="longitude">
```

```
            <span id="longitude-symbol">&#128206;</span> <!--
Símbolo para la longitud -->
        </p>
    </font>

    <!-- Cargar el script de socket.io -->
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/4.7.1/socket
.io.js"></script>
    <script>
        // Conectar al servidor mediante socket.io
        var socket = io();

        // Escuchar el evento 'data' desde el servidor
        socket.on('data', function (data) {
            // Mostrar los datos recibidos en la página
            document.getElementById("temp").innerHTML =
"Temperatura: " + data.temp; // Mostrar temperatura con símbolo
            document.getElementById("pressure").innerHTML =
"Presion: " + data.pressure + "<span id='pressure-
symbol'>&#128167;</span>"; // Mostrar presión con símbolo
            document.getElementById("humidity").innerHTML =
"Humedad: " + data.humidity + "<span id='humidity-
symbol'>&#128167;</span>"; // Mostrar humedad con símbolo
            document.getElementById("latitude").innerHTML =
"Latitud: " + data.latitude + "<span id='latitude-
symbol'>&#128205;</span>"; // Mostrar latitud con símbolo
            document.getElementById("longitude").innerHTML =
"Longitud: " + data.longitude + "<span id='longitude-
symbol'>&#128206;</span>"; // Mostrar longitud con símbolo
        });
    </script>
</body>
</html>
```

Demostración de los 3 códigos funcionando:



Conclusión del Proyecto " Sensor atmosférico con ubicación GPS usando UDP "

En conclusión, el proyecto " Sensor atmosférico con ubicación GPS usando UDP " ha demostrado ser una solución efectiva para obtener datos ambientales y de ubicación en tiempo real. Mediante la combinación de un sensor BME280 para medir la temperatura, presión y humedad, y un módulo GPS para obtener la latitud y longitud, hemos logrado construir un sistema capaz de brindar información valiosa sobre el entorno.

El desarrollo del proyecto nos ha permitido explorar y utilizar diferentes tecnologías. El código principal, implementado en un microcontrolador, se encarga de leer los valores del sensor BME280 y los datos del módulo GPS, estableciendo una conexión UDP para enviarlos a un servidor. A su vez, el servidor Node.js recibe estos datos y los envía a los clientes conectados a través de WebSockets, lo que permite visualizarlos en tiempo real en una interfaz web.

Además de proporcionar información precisa y actualizada, el proyecto destaca por su versatilidad y aplicabilidad en diversos campos. Puede utilizarse para monitorear el entorno en tiempo real, realizar análisis geoespaciales, rastrear vehículos y mucho más. Además, al ser una solución escalable, es posible agregar más sensores o funcionalidades según las necesidades específicas.

Durante el desarrollo del proyecto, hemos adquirido conocimientos en el manejo de sensores, comunicación en tiempo real, programación en diferentes lenguajes y configuración de redes. Estas habilidades son fundamentales en el campo de la Internet de las Cosas (IoT) y nos permiten explorar nuevas posibilidades y aplicaciones.

En resumen, el proyecto " Sensor atmosférico con ubicación GPS usando UDP " nos ha brindado una experiencia enriquecedora, combinando hardware y software para obtener y visualizar datos ambientales y de ubicación en tiempo real. Nos hemos adentrado en el mundo de la IoT y hemos aprendido cómo integrar diferentes tecnologías para crear soluciones prácticas y útiles.