INFORME DE LABORATORIO 5:

Mapeando el nivel de ruido

Autores: *Santiago Giraldo Tabares, Ana María Velasco Montenegro*

Laboratorio de Electrónica Digital 3
Departamento de Ingeniería Electrónica y de Telecomunicaciones
Universidad de Antioquia

Resumen

En este documento se describe el desarrollo de un sistema integrado diseñado para la medición y geolocalización del ruido ambiental, proporcionando una herramienta eficaz para el monitoreo de la contaminación acústica. El dispositivo cuenta con un diseño ergonómico y es alimentado por batería, lo que permite su uso en diferentes entornos sin preocupaciones por conexiones eléctricas. Al encender el dispositivo, un LED verde indica la disponibilidad del sistema para operar, mientras que un pulsador activa el proceso de medición del ruido, que es captado por un micrófono durante un periodo de 10 segundos. Durante este tiempo, el módulo GPS adquiere simultáneamente la ubicación geográfica de la medición, permitiendo al microcontrolador etiquetar cada registro de sonido con su correspondiente posición geográfica.

A lo largo de la operación, varios LEDs indican el estado del dispositivo: verde para listo, amarillo durante la medición y análisis, naranja al almacenar datos exitosamente, y rojo para alertar de errores o falta de señal de GPS. Además, el sistema cuenta con una interfaz USB para la descarga de datos y se incorporan estrategias de reducción del consumo energético mediante el uso de modos SLEEP o DORMANT en el microcontrolador.

Este sistema no solo ofrece una solución tecnológica para el monitoreo ambiental, sino que también se plantea como un proyecto en la Universidad de Antioquia, donde se podrá visualizar los datos recogidos en las porterias del campus.

Palabras clave: *Raspberry Pi Pico, Almacenamiento EEPROM I2C, Detección de sonido, LEDS*

Introducción

En el ámbito de la sostenibilidad ambiental y el control de la contaminación, el monitoreo de la contaminación acústica se ha convertido en un aspecto crucial, especialmente en zonas urbanas densamente pobladas. La exposición prolongada al ruido puede tener efectos adversos significativos sobre la salud humana y el bienestar gene-

ral de los ecosistemas. Para abordar esta problemática, se ha diseñado un dispositivo integrado de medición y geolocalización del ruido ambiental, utilizando como núcleo central la Raspberry Pi Pico. Este microcontrolador ofrece una solución efectiva y eficiente para la recolección y procesamiento de datos ambientales en tiempo real.

El dispositivo está compuesto por un conjunto de módulos tecnológicamente avanzados, incluyendo un micrófono con amplificador para la detección de sonido, un módulo GPS para la geolocalización precisa, memoria no volátil para el almacenamiento de datos, junto con varios LEDs de colores y un pulsador para la interacción del usuario. La elección de la Raspberry Pi Pico como microcontrolador central es clave debido a su capacidad de procesamiento de 32 bits, bajo consumo de energía y flexibilidad en la programación y conexión de periféricos.

Al encender el dispositivo, un LED verde señala que el sistema está operativo y listo para la captura de datos. El usuario puede iniciar la medición de ruido ambiental mediante un pulsador, activando el micrófono para registrar el sonido durante un período determinado. Simultáneamente, el módulo GPS adquiere la ubicación exacta de la medición, permitiendo que la Raspberry Pi Pico procese y etiquete cada muestra de sonido con su correspondiente coordenada geográfica. Este enfoque integrado no solo mejora la precisión y relevancia de los datos recogidos, sino que también facilita un análisis detallado y geoespacial de la contaminación acústica para futuras intervenciones y políticas ambientales.

Marco teórico

Para el desarrollo de un dispositivo de medición y geolocalización del ruido ambiental utilizando una Raspberry Pi Pico, es fundamental comprender el papel y la funcionalidad de cada componente dentro del sistema. La Raspberry Pi Pico, actuando como el cerebro del dispositivo, juega un papel crucial en la coordinación de los módulos y en la gestión de datos. A continuación, se profundiza en cada uno de los componentes y tecnologías implicados en este proyecto.

0.0.1. Raspberry Pi Pico:

La Raspberry Pi Pico es un microcontrolador que utiliza el chip RP2040, diseñado por la Raspberry Pi Foundation. Este microcontrolador destaca por su bajo costo, versatilidad y rendimiento, lo que lo hace ideal para proyectos de electrónica y sistemas embebidos. La Pi Pico ofrece un procesamiento de 32 bits, múltiples pines de E/S, soporte para lenguajes como C++ y MicroPython, y características ideales para la gestión de dispositivos periféricos mediante sus interfaces GPIO, SPI, I2C, entre otras.

0.0.2. Módulo GPS:

El módulo GPS se utiliza para determinar la posición geográfica exacta del dispositivo durante cada medición de ruido. La integración con la Raspberry Pi Pico permite al microcontrolador etiquetar cada lectura de ruido con su ubicación precisa, facilitando análisis geoespaciales detallados de la contaminación acústica.

0.0.3. Módulo EEPROM I2C AT24C256

El módulo EEPROM I2C AT24C256 ofrece una capacidad de almacenamiento de 256 kilobytes, suficiente para guardar grandes cantidades de datos de ruido y sus coordenadas GPS correspondientes. La interfaz I2C se usa para comunicarse con la Raspberry Pi Pico, permitiendo el almacenamiento y recuperación eficiente de datos incluso cuando el dispositivo está apagado o se reinicia.

0.0.4. Módulo de detección de sonido con amplificador LM393

Este módulo capta el ruido ambiente a través de un micrófono y utiliza el amplificador operacional LM393 para amplificar la señal sonora. La señal amplificada se digitaliza y procesa por la Raspberry Pi Pico para calcular el nivel de ruido en decibelios. Este módulo es esencial para obtener mediciones precisas y confiables del entorno acústico.

0.0.5. LEDs y pulsadores

Los LEDs proporcionan una indicación visual del estado del dispositivo, mientras que los pulsadores permiten al usuario interactuar con el sistema, iniciando y deteniendo la medición según sea necesario. Estos componentes mejoran la usabilidad del dispositivo, facilitando su manejo en terreno.

0.0.6. Gestión del consumo de energía

Considerando que el dispositivo es portátil y operado por batería, la Raspberry Pi Pico ofrece características para el manejo eficiente del consumo energético. Los modos SLEEP y DORMANT pueden ser utilizados para minimizar el consumo de energía cuando el dispositivo no está activo, prolongando así la duración de la batería.

Procedimiento:

Para implementar el sistema de medición y geolocalización del ruido ambiental utilizando la Raspberry Pi Pico, se seguirán una serie de pasos metodológicos y técnicos que aseguren el correcto funcionamiento del dispositivo. Este procedimiento incluye la preparación del entorno de desarrollo, instalación de herramientas y librerías, configuración del hardware, y la integración y prueba del sistema completo.

0.0.1. Configuración del hardware y entorno de desarrollo:

Involucra la preparación y configuración de la Raspberry Pi Pico junto con todos los componentes necesarios, además se debe descargar e instalar todas las librerías necesarias para manejar los módulos de GPS, detección de sonido, memoria EEPROM, y control de LEDs y pulsador.

0.0.2. Análisis previo:

Se realiza el análisis previo del programa a resolver por medio de un diagrama del flujo el cual permite tener claridad del proceso que se debe realizar. Esto es muy importante porque así se tiene una idea también de lo que se necesita y el momento en que se necesita. Se observa en la siguiente imagen:

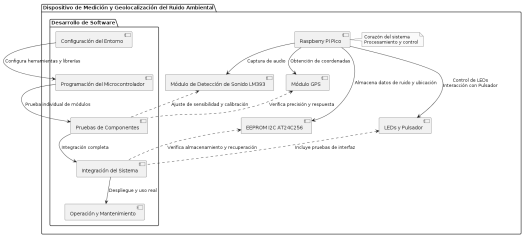

Figura **0-1**: Diagrama de flujo general

0.0.3. Módulos a usar:

- **Módulo de detección de sonido con amplificador LM393** El módulo de detección de sonido LM393 capta el ruido ambiental y utiliza un amplificador operacional para aumentar la amplitud de la señal de audio. Esta señal amplificada se envía a la Raspberry Pi Pico para su digitalización y procesamiento. La interacción se centra en la captura y envío de la señal de audio mediante la conversión analógica a digital (ADC) en la Pico.

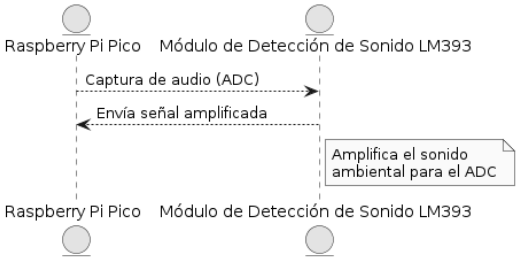

Figura **0-2**: Diagrama de flujo de detección de Sonido

- **Módulo de GPS** El módulo GPS se comunica con la Raspberry Pi Pico a través de una conexión UART. La Pico solicita y recibe datos de ubicación del GPS, los cuales son esenciales para geolocalizar las mediciones de ruido. Este módulo asegura que cada registro de sonido esté acompañado de su correspondiente posición geográfica.

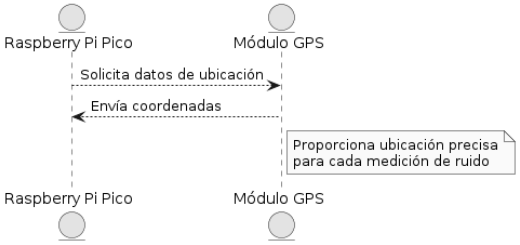

Figura **0-3**: Diagrama de flujo GPS

- **Memoria EEPROM I2C AT24C256** La EEPROM I2C AT24C256 es utilizada para almacenar los datos de ruido y ubicación de manera no volátil. La Raspberry Pi Pico envía los datos a la EEPROM a través del protocolo I2C, y la EEPROM confirma el correcto almacenamiento. Esto permite preservar los datos incluso cuando el dispositivo se apaga.

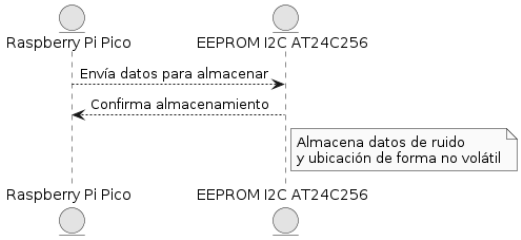

Figura **0-4**: Diagrama de flujo Memoria EEPROM I2C AT24C256

- **LEDs y Pulsador** Los LEDs y el pulsador proporcionan una interfaz de usuario directa para el dispositivo. Los LEDs indican el estado operativo del dispositivo (listo, en medición, error, etc.), mientras que el pulsador permite al usuario iniciar o cancelar procesos de medición. La Raspberry Pi Pico controla los LEDs y recibe señales del pulsador para gestionar las operaciones del sistema.

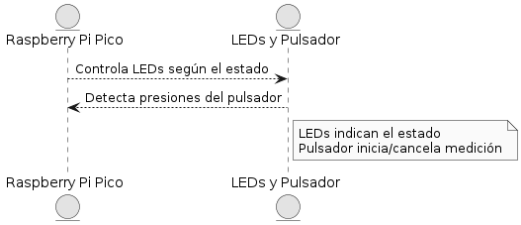

Figura **0-5**: Diagrama de flujo Modulo LEDs y Pulsador

0.0.4. Circuito implementado:

Se puede observar el circuito completo implementado con todos los elementos como el módulo de detección de sonido con amplificador LM393, módulo de GPS, memoria EEPROM I2C AT24C256, leds, pulsadores y la Raspberry Pi Pico.

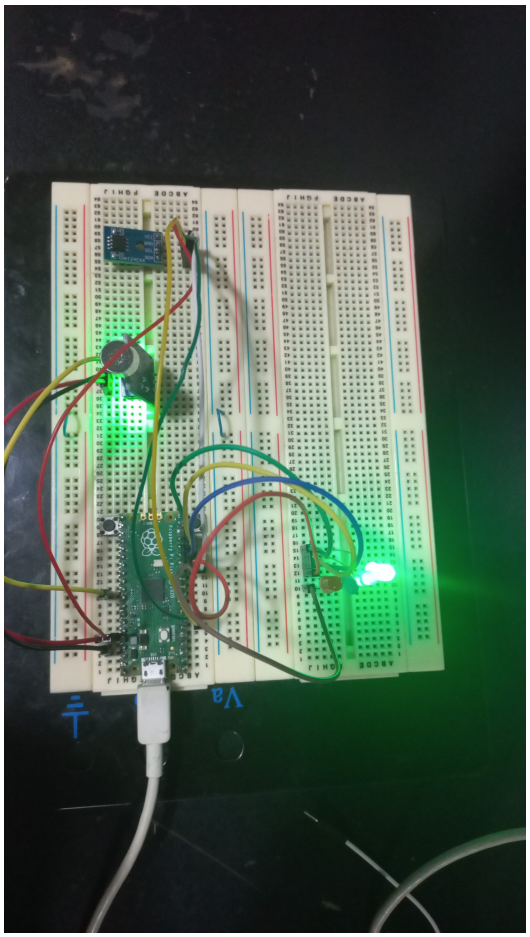

Figura **0-6**: Circuito implementado

0.0.5. Código Implementado

El código está diseñado para gestionar la captura de audio, visualización del estado mediante LEDs, entrada de usuario a través de botones, y almacenamiento de datos en una memoria EEPROM. Al ejecutarse, el programa configura inicialmente los pines de entrada/salida necesarios, incluyendo los destinados al micrófono, LEDs, y botones. Utiliza interrupciones para manejar las pulsaciones de botones, lo que permite al usuario interactuar con el sistema cambiando su estado operativo o solicitando la descarga de datos.

La captura del audio se realiza a través de un ADC que convierte la señal analógica del micrófono en valores digitales que luego son procesados para determinar el nivel de ruido en decibelios. Esto se logra calculando el valor RMS de la señal captada y aplicando la fórmula correspondiente para obtener el nivel de presión sonora. Estos datos son esenciales para evaluar la contaminación acústica en diferentes entornos.

El sistema está diseñado para operar bajo una máquina de estados, facilitando la gestión de diferentes modos como la espera (IDLE), operación (OPERATION), éxito en la captura y almacenamiento (SUCCESSFUL), y cancelación (CANCELLED). Cada estado permite realizar acciones específicas como iniciar o detener la medición del ruido, almacenar los resultados obtenidos, o cancelar operaciones en curso.

Para la persistencia de los datos, el código incluye funciones que permiten escribir y leer de la memoria EEPROM. Estas funciones manejan la asignación de direcciones dentro de la EEPROM y aseguran que los datos de las mediciones de sonido sean guardados y recuperados eficazmente. Esto es vital para análisis posteriores o revisiones de los datos capturados.

En su conjunto, el código proporciona una solución robusta para la medición y registro de niveles de ruido, con capacidades interactivas para el usuario y almacenamiento eficiente de datos, lo que lo hace ideal para aplicaciones de monitoreo ambiental o estudios relacionados con la contaminación acústica.

Se muestra algunas porciones de código implementado:

```c
void init_i2c() {
    i2c_init(I2C_PORT, 100 * 1000);
    gpio_set_function(I2C_SDA_PIN, GPIO_FUNC_I2C);
    gpio_set_function(I2C_SCL_PIN, GPIO_FUNC_I2C);
    gpio_pull_up(I2C_SDA_PIN);
    gpio_pull_up(I2C_SCL_PIN);
}

void write_to_eeprom(uint16_t address, uint8_t* data, size_t length) {
    uint8_t buffer[length + 2];
    buffer[0] = address >> 8;
    buffer[1] = address & 0xFF;
    memcpy(&buffer[2], data, length);
    i2c_write_blocking(I2C_PORT, EEPROM_ADDRESS, buffer, length + 2, false);
    sleep_ms(5);  // EEPROM write delay
}

void read_from_eeprom(uint16_t address, uint8_t* data, size_t length) {
    uint8_t address_bytes[] = {address >> 8, address & 0xFF};
    i2c_write_blocking(I2C_PORT, EEPROM_ADDRESS, address_bytes, 2, true);
    i2c_read_blocking(I2C_PORT, EEPROM_ADDRESS, data, length, false);
}

uint16_t find_next_write_address() {
    uint16_t address = 0;
    uint8_t buffer[5];  // Buffer to read 5 bytes (dB value + timestamp)
    while (address < 0x7FFF) {  // Assuming 32KB EEPROM
        read_from_eeprom(address, buffer, sizeof(buffer));
        if (buffer[0] == 0xFF) {
            break;  // Found an empty slot
        }

        address += sizeof(buffer);
    }
    return address;
}
```

Figura **0-7**: Código implementado

```c
void save_average_db(float average_db) {
    uint16_t address = find_next_write_address();
    if (address < 0x7FFF) {
        uint8_t data[5]; // Keep the data size to store dB value and timestamp
        int16_t db_value = (int16_t)(average_db * 100);  // Store dB value with 2 decimal places
        data[0] = db_value >> 8;
        data[1] = db_value & 0xFF;

        // Get current system time
        time_t current_time = time(NULL);
        struct tm *local_time = localtime(&current_time);

        // Store the timestamp in EEPROM
        data[2] = local_time->tm_hour;
        data[3] = local_time->tm_min;
        data[4] = local_time->tm_sec;

        write_to_eeprom(address, data, sizeof(data));
    }
}

void erase_eeprom_memory() {
    for (uint16_t address = 0; address < 0x7FFF; address += 64) { // Erase 64 bytes at a time
        uint8_t erase_data[64];
        memset(erase_data, 0xFF, sizeof(erase_data)); // Fill with 0xFF
        write_to_eeprom(address, erase_data, sizeof(erase_data));
    }
}

void dump_data_from_eeprom() {
    uint16_t address = 0;
    uint8_t buffer[5];
    int max_iterations = 1000; // Set a maximum number of iterations
    int iterations = 0;
    while (address < 0x7FFF && iterations < max_iterations) {
        read_from_eeprom(address, buffer, sizeof(buffer));
        if (buffer[0] == 0xFF) {
            break;  // No more data
        }
        int16_t db_value = (buffer[0] << 8) | buffer[1];
        float db = db_value / 100.0;

        // Extract timestamp
        struct tm tm_info;
        tm_info.tm_hour = buffer[2];
        tm_info.tm_min = buffer[3];
        tm_info.tm_sec = buffer[4];
```

Figura **0-8**: Código implementado

0.0.6. Gráfica muestras

Se observa en la siguiente imagen las mediciones de nivel de ruido:

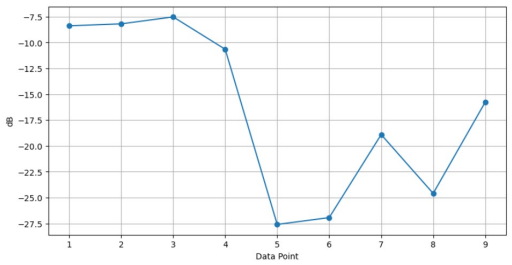

Figura **0-9**: Gráfica de nivel de Ruido

Se muestra la evolución de las mediciones de nivel de ruido (en decibelios, dB) a lo largo de nueve puntos de datos, con el eje horizontal representando las muestras de forma secuencial y el eje vertical mostrando los niveles de ruido en dB. La disposición de los puntos muestra que las mediciones fueron tomadas en el orden inverso, es decir, de derecha a izquierda en la gráfica, con el punto 9 siendo el más reciente y el punto 1 el menos reciente.

Analizando los valores y la gráfica, se observa una variabilidad significativa en los niveles de ruido. Los datos comienzan alrededor de -7.5 dB, aumentan ligeramente, y luego experimentan un descenso pronunciado hasta aproximadamente -27 dB, antes de aumentar nuevamente hasta cerca de -8 dB en los datos más recientes.

Los valores específicos extraídos del archivo de texto muestran que los niveles de ruido más bajos se registraron a las 00:01:54 y a las 00:02:10, con valores de -27.59 dB y -26.93 dB respectivamente. Estas mediciones corresponden a los puntos 5 y 6 en la gráfica, que son precisamente donde la línea muestra el descenso más pronunciado.

Este tipo de análisis es útil para entender las fluctuaciones en el entorno acústico evaluado, posiblemente atribuibles a cambios en las condiciones ambientales, actividades humanas en las proximidades, o variaciones en la sensibilidad del equipo de medición durante el periodo de registro. Este entendimiento es fundamental para diseñar estrategias efectivas de mitigación del ruido o para realizar estudios más detallados sobre las fuentes de ruido y su impacto en el entorno.

Obstáculos en la implementación

Durante el desarrollo y la implementación del dispositivo de medición y geolocalización del ruido ambiental, nos enfrentamos a un problema significativo con el módulo GPS. A pesar de contar con condiciones óptimas para su funcionamiento, como un espacio abierto que garantiza una clara visibilidad del cielo, el módulo GPS no logró funcionar correctamente.

Inicialmente, se sospechó de problemas relacionados con la señal; sin embargo, las pruebas exhaustivas confirmaron que la ubicación no era el factor limitante. Tras investigaciones adicionales, se descubrió que el módulo GPS utilizado podría ser una réplica de baja calidad. Estos módulos falsificados son comunes en el mercado y no ofrecen el mismo nivel de rendimiento que sus contrapartes originales.

Para diagnosticar más a fondo el problema, se empleó el software u-center, herramienta líder en el análisis y la configuración de módulos GPS. Las pruebas se realizaron en diferentes momentos del día, buscando eliminar cualquier variable temporal que pudiera afectar la recepción de la señal. Lamentablemente, los resultados fueron consistentemente negativos.

Además, se intentó mejorar la captación de señal utilizando una antena externa en lugar de la antena cerámica estándar que viene con el módulo. A pesar de este cambio, no hubo mejoría en la capacidad del módulo para detectar satélites.

Cabe destacar que la comunicación por UART entre la Raspberry Pi Pico y el módulo GPS se estableció correctamente. Las sentencias NMEA se enviaban y recibían sin problemas, lo que indica que el problema residía específicamente en la incapacidad del módulo GPS para establecer una conexión con los satélites, en lugar de un error en la comunicación digital.

Discusión

El dispositivo diseñado para medir el ruido ambiental ha funcionado efectivamente en la captura y procesamiento del sonido, aunque se enfrentaron desafíos significativos con el módulo GPS, cuya incapacidad para localizar satélites resalta la importancia de verificar la calidad de los componentes electrónicos en proyectos críticos. El uso del ADC ha sido crucial, permitiendo un análisis preciso y confiable del ruido mediante la conversión de ondas sonoras en datos digitales. Además, la implementación de la EEPROM a través de la interfaz I2C ha facilitado un almacenamiento robusto y eficiente de datos, pero presenta limitaciones para almacenar datos en formatos estructurados, requiriendo especial atención al recuperar y formatear los datos. A pesar de los problemas con el GPS, la interfaz de usuario con botones y LEDs ha funcionado eficazmente, facilitando la interacción directa y la gestión del dispositivo a través de una estructura de máquina de estados. Se identifican oportunidades significativas para mejorar el proyecto, como implementar un módulo GPS de mejor calidad y añadir funcionalidades de comunicación inalámbrica para expandir la operatividad.

La implementación final del proyecto, que incluye todos los códigos fuente, documentación, procesos los cuales son necesarios para comprender y replicar el laboratorio número 5 de la materia Electrónica Digital 3 desarrollado para la Raspberry Pi Pico, está disponible públicamente para consulta y descarga en GitHub. Este repositorio representa el trabajo en equipo y las iteraciones de desarrollo que han caracterizado este proyecto desde su inicio. El link al repositorio es el siguiente: GitHub Mapeando el nivel de ruido

Recomendaciones

Es importante verificar la autenticidad y la calidad de los componentes electrónicos antes de su integración en proyectos críticos. Para futuros proyectos, recomendamosla adquisición de módulos GPS a través de canales confiables y verificar su funcionamiento antes de proceder con la integración en el sistema.

Conclusiones

- Aunque el dispositivo diseñado para medir el ruido ambiental ha funcionado adecuadamente en términos de captura y procesamiento de sonido, se enfrentaron desafíos significativos con el módulo GPS. La incapacidad de este componente para localizar satélites, a pesar de los esfuerzos exhaustivos y las pruebas con diferentes configuraciones y antenas, subraya la importancia de verificar la autenticidad y calidad de los componentes electrónicos usados en proyectos críticos.
- El uso del convertidor analógico a digital (ADC) en este proyecto para capturar señales del módulo de micrófono ha sido crucial. Al convertir las ondas sonoras en datos digitales, el ADC permite un análisis preciso y cuantitativo del nivel de ruido ambiental. Esta traducción fidedigna de señales analógicas a formatos digitales asegura que la medición del ruido sea tanto precisa como confiable, facilitando el monitoreo efectivo de la contaminación acústica en diferentes entornos.
- La implementación de la memoria EEPROM con la interfaz de comunicación serial I2C en este proyecto ha resultado ser extremadamente eficiente y sencilla. La interfaz I2C facilita la conexión y comunicación con la EEPROM, permitiendo un manejo fluido y robusto del almacenamiento de datos.
- Dado que la señal que se obtiene con el microfono es una señal acustica dentro del rango de frecuencias audibles, no resulta demasiado complicado muestrearla como si podría ocurrir con frecuencias muy altas. Por lo tanto no se requiere un tratamiento especial del ADC, y menos si es para sacarle promedios a las muestras obtenidas.
- El uso de la memoria EEPROM en este proyecto, aunque eficaz para almacenar datos en bruto, presenta limitaciones en comparación con soluciones de almacenamiento más avanzadas como módulos SD. Aunque la EEPROM es fácil de implementar y efectiva para almacenar datos crudos mediante la interfaz I2C, no admite directamente el almacenamiento de datos en formatos estructurados como tablas de Excel o archivos de texto. Esto implica que debe prestarse especial atención al recuperar y formatear los datos, para transformarlos en formatos más útiles y manejables. En futuras implementaciones, considerar el uso de un módulo SD podría ofrecer una mayor flexibilidad y accesibilidad para el manejo de datos en formatos específicos, facilitando así la integración y análisis posterior de la información recopilada.
- El problema con los módulos GPS falsificados o de baja calidad destaca un riesgo considerable en la adquisición de componentes. Este desafío afectó la capacidad del sistema para geolocalizar las mediciones de ruido, una funcionalidad clave del proyecto. La experiencia reafirma la necesidad de adquirir componentes de proveedores confiables y realizar pruebas exhaustivas antes de su integración.
- A pesar de los problemas con el GPS, la interfaz de usuario mediante botones y LEDs funcionó eficazmente, permitiendo una interacción clara y directa. La estructura de la máquina de estados facilitó la operación del dispositivo y la gestión de diferentes condiciones de trabajo, como la medición, el almacenamiento de datos, y la cancelación de operaciones.
- Reconociendo los desafíos enfrentados, se identifican oportunidades significativas para mejorar y expandir el proyecto. Implementar un módulo GPS de mejor calidad o de una marca reconocida podría resolver los problemas actuales y permitir la plena realización de las capacidades del dispositivo. Además, la integración de funcionalidades de comunicación inalámbrica podría ofrecer una operatividad y accesibilidad mejoradas.
- Para futuras iteraciones y proyectos similares, se recomienda enfocarse en la validación detallada de cada componente antes de su incorporación en el sistema. También se sugiere la exploración de alternativas para la mejora de la eficiencia energética y la ampliación de la autonomía del dispositivo, lo que permitiría un monitoreo más extensivo y menos intrusivo en diversos entornos.

Referencias

[**1**] Monk, S. (2016). Programming the Raspberry Pi: Getting Started with Python. McGraw-Hill Education. [**2**] Norris, M. (2022). The Maker's Guide to the IoT: Raspberry Pi Pico. Packt Publishing. [**3**] Johnson, L., Gupta, A. (2020). IoT applications in environmental monitoring: A review. Environmental Technology Innovation, 18, 100777. [**4**] Lee, W., Kim, Y. (2018). Noise pollution and control within urban environments. Journal of Environmental Management, 206, 577-583.