Introducción

El dispositivo M5-Stack-FIRE, es una plataforma de desarrollo integral que redefine la experiencia de prototipado y creación de proyectos electrónicos. Equipado con un potente microcontrolador ESP32, pantalla táctil a color, altavoz, batería recargable y una amplia gama de sensores, M5-Stack-FIRE ofrece un ecosistema versátil y compacto para diseñadores, ingenieros y entusiastas de la electrónica.
de un magnetómetro utilizando el sensor BMM150 en conjunto con la plataforma de desarrollo M5Stack y el entorno de programación Arduino. El magnetómetro se utiliza para medir campos magnéticos en tres dimensiones (x, y, z), y para garantizar la precisión.
Para la realización de la practica utilizaremos las bibliotecas disponibles (preferentemente las bibliotecas oficiales proporcionadas por M5Stack) que facilitan el uso de los periféricos, sensores y transductores integrados.

Objetivos

Implementación de un detector de metales basado en el geo magnetómetro BMM150 que integra el M5Stack-Fire, cumpliendo con las siguientes especificaciones:
• El visualizador LCD mostrara el valor de campo magnético compensado en Z, el valor de referencia minimo del campo magnético en Z, el valor de referencia máximo del campo magnético en Z y el valor diferencial del campo magnético en Z.
• Al pulsar el botón A se entrara en modo calibración durante 10 s. La calibración implementara tanto la compensación de la distorsión *hard iron* (*offset*) como de la distorsión *soft iron* (*scale*).
• Al pulsar el botón B se tomara como valor de referencia minimo del campo magnético en Z, el valor actual del campo magnético compensado en Z **sin ningún metal próximo**.
• Al pulsar el botón C se tomara como valor de referencia máximo del campo magnético en Z, el valor actual del campo magnético compensado en Z **con un metal muy próximo**.
• El altavoz reproducirá un tono cuya frecuencia variara entre 1 kHz y 4 kHz en función de la distancia al metal detectado (valor diferencial del campo magnético en Z). El

altavoz permanecerá inactivo si el valor diferencial del campo magnético en Z no supera en un 5% el valor de referencia minimo. La frecuencia máxima del tono se corresponderá con un valor diferencial del campo magnético en Z superior al 95% del valor de referencia máximo.
Tras añadir a la calibración **bmm150_calibrate** la compensación de la distorsión *soft iron* (*scale*), debe modificar las funciones **bmm150_offset_save** y **bmm150_offset_load** para que también almacenen/restauren el valor de escala.

metal_detector_average

Tenemos que modificar el programa anterior para añadir un promediado de los valores del campo magnético en X, Y, Z configurable entre 1 (sin promediado) y 10 valores. Utilice los valores del campo magnético promediados en lugar de los instantáneos. Aumentar la frecuencia de muestro disminuyendo el retardo del bucle principal de 100 ms a 10 ms.
Para implementar el promediado utilizaremos un buffer circular.

Desarrollo

 # Inicialización de bibliotecas y variables globales

Incluir bibliotecas necesarias

Definir constantes y estructuras

Inicializar `Preferences prefs`

Inicializar estructuras del magnetómetro:

```
struct bmm150_dev dev;
```

, circular_buffer y otras variables globales que necesitaremos.

Crear objetos para el manejo de buffers y datos como:

```
circular_buffer bufferX;
circular_buffer bufferY;
circular_buffer bufferZ;
```

 # Función para limpiar el buffer

Función value_clear(circular_buffer *buf):

Establece la cabeza y la cola del buffer a 0

Inicializa los valores del buffer a 0

 # Función para encolar un valor en el buffer

Función `value_queue(circular_buffer *buf, float value)`:

Almacenamos el valor en la posición de la cabeza del buffer

Actualizamos la cabeza (circular)

Ajustamos la cola si la cabeza alcanza la cola

 # Función para calcular el promedio de un buffer

Función `value_average(const circular_buffer *buf)`:

Inicializa suma y contador a 0

Inicializa un índice al valor de la cola del buffer

Mientras el índice no alcance la cabeza del buffer:

Suma el valor en la posición del índice a la suma

Incrementa el índice (circular)

Incrementa el contador

Calcula y devolve el promedio si el contador es mayor que 0, de lo contrario, devuelve 0.

 # Función para guardar la configuración del magnetómetro

Función `bmm150_offset_save()`:

Inicializar preferencias con el nombre "bmm150"

Guardar los datos de offset y scale en las preferencias

Finalizar preferencias

 # Función para cargar la configuración del magnetómetro

Función `bmm150_offset_load()`:

Si es posible comenzar las preferencias con el nombre "bmm150":

Cargar los datos de offset y scale desde las preferencias

Finalizar preferencias

Imprimir mensaje de carga exitosa o fallida

 # Función para reproducir un tono según la diferencia Z

Función `reproducirTono(float diferenciaZ):`

Definir porcentajes mínimo y máximo, incrementado un 5% al minimo y restándole un 5% al máximo

Si la diferenciaZ está dentro del rango permitido:

Mapear la diferenciaZ a una frecuencia entre 1000 y 4000 Hz, dentro del rango mínimo y máximo

Reproducir un tono con la frecuencia mapeada durante 1000 ms

De lo contrario, silenciar el altavoz

 # Configuración inicial

Iniciar M5Stack y otras configuraciones iniciales..

Configurar el objeto TFT_eSprite y otros componentes.

Cargar la configuración guardada

Limpiar los buffers de datos del circular_buffer y ponerlos a 0.

Iniciar la calibración del magnetómetro si llamamos a la función `void bmm150_calibrate(uint32_t calibrate_time), en esta función añadimos los cálculos necesarios para compensar el valor de Z del ruido provocado por el hard & soft iron. También guardaremos los datos calibrados gracias a la función offset_saved() que más tarde recuperaremos.`

 # Bucle principal

Actualizar el estado de M5Stack y leer datos del magnetómetro pedidos.

Calcular el valor Z compensado del *hard y soft iron* y actualizamos los buffers de datos llamando a la función `value_queue(&bufferZ, z);`

Calculamos el promedio de los datos con la variable y llamando a la función:

```
float promedioZ = value_average(&bufferZ)
```

y otras operaciones como el cálculo de la diferenciaZ:

```
float diferenciaZ = (promedioZ - minimo).
```

Mostrar información en la pantalla.

Procesar acciones de los botones A, B y C:

Si presionamos el botón A:

Iniciamos la calibración

Si presionamos el botón B:

Establecemos el valor mínimo del promedioZ (sin metal cerca)

Si presionamos el botón C:

Establecemos el valor máximo de promedioZ (con metal muy cerca)

Llamamos a la función Reproducir tono según la diferencia Z y solo si ingresamos un valor a las variables mínimo y máximo. Así evitamos pitidos indeseados desde el principio.

Esperar 10 ms para una frecuencia de muestreo.

Conclusiones

Concluyendo, el código proporciona una inmersión fascinante en el universo de la magnetometría y el desarrollo de proyectos con el M5Stack-FIRE. Este último se revela como una herramienta poderosa y versátil, aprovechando al máximo su conjunto de sensores y capacidades multimedia. Desde la calibración inteligente del magnetómetro hasta la reproducción dinámica de tonos acústicos, cada linea de código refleja la sinergia entre hardware y software, demostrando como M5Stack-FIRE puede transformar las mediciones del campo magnético.

En segundo plano, el código también resalta la importancia de la calibración necesaria para mejorar la confiabilidad de las mediciones. La capacidad de almacenar y cargar valores de compensación proporciona una solución elegante para adaptarse a diversas condiciones de uso. El uso del buffer circular es una interesante técnica que agrega muchos valores y hace un promedio de ellos, muy útil para este caso en concreto y en un futuro. Además, la incorporación de funciones alternativas mediante los botones del M5Stack-FIRE añade una agradecida funcionabilidad a esta.

En resumen, este código no solo representa una aplicación práctica del M5Stack-FIRE en el ámbito de la magnetometría, sino también un testimonio de cómo la creatividad y la funcionabilidad convergen en esta plataforma de desarrollo emocionante y robusta.