INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

ITESO

Universidad Jesuita de Guadalajara

Práctica 2

Alumnos: Eduardo Ethandrake Castillo Pulido ie714410
Alejandro Gudiño Gallegos ie714594

Sistemas Embebidos 2

Profesor: Edgardo Serna

20 de octubre de 2020

Descripción de la Practica

En esta practica desarrollaras un modulo AHRS usando la IMU BMI160 la cual contiene un acelerómetro y un giroscopio para la medición de los parámetros necesarios, además de la tarjeta FRDM-K66F para implementar el algoritmo de fusión de sensores y enviar dichos valores obtenidos a una PC mediante una comunicación serial (UART), donde se graficarán a través de un script en Python.

Requerimientos de la practica

Para esta practica deberás diseñar un sistema que cumpla con lo siguiente:

1. Deberás diseñar un sistema con la arquitectura especificada en el documento
2. Usar la FRDM-K66F sobre la cual deberá usar FreeRTOS como capa de abstracción del sistema operativo y MCUXpreso-SDK como capa de abstracción de microcontrolador.
3. Para la capa de abstracción de hardware: diseñaras y codificaras módulos para I2C y UART
4. La comunicación entre el BMI160 y la FRDM-K66F es a través del periférico I2C, utilizando el modulo especificado en el punto 3.
5. La comunicación entre la FRDM-K66F y la PC es a través del periférico UART, utilizando el modulo indicado en el requerimiento 3.
6. Implementar la librería BMI160
7. Implementar la librería de AHRS, la cual hará la conversión de los datos obtenidos de los sensores del BMI160 hacia ángulos de navegación.
8. 5 milis y 20 milis

Decisiones claves para resolver el problema

Para poder realizar la tarea lo primero fue realizar el driver del I2C y el de la BMI160. Estos dos drivers fueron desarrollados en la tarea 6. Para poder verificar el buen funcionamiento de estos drivers se mando a imprimir los valores que se obtenían del sensor BMI160.

Después de tener esos drivers, se procedió a incluir la librería de Mahoney, la cual fue proporcionada por el maestro. Para mandar los valores a estas funciones, lo primero que se hizo fue calibrar el sensor. Para esta calibración lo que se hizo fue inicializar una tarea que se ejecuta al inicio del programa y después se suspende. Esta tarea lo que hace es tomar 100 muestras cuando el sistema esta en reposo, y de esta manera podemos determinar cuales son los valores cuando el sistema no se está moviendo.

Al final de la tarea de calibración, lo que se hizo fue crear dos nuevas tareas, una que va a estar recibiendo datos de la BMI160 y otra que va a mandar por UART. Después de crear la tarea, se suspende la tarea de calibración, lo cual deja solo dos tareas corriendo. La de tomar datos del sensor, y la de mandar datos por medio de la UART.

El tiempo en el que se toma muestra del sensor es de 5 milisegundos, mientras que se manda datos por medio de la UART cada 20 milisegundos. También a la hora de estar tomando muestras se esta obteniendo un promedio, y ese valor promedio se esta mandando a la función de Mahoney. Para poder obtener el promedio se están tomando 4 muestras, por eso mandamos por UART casa 20 milisegundos, para poder tomar 4 muestras, obtener promedio y mandarlo en la UART.

Los valores de cuantas muestras tomar para la calibración y el promedio fueron obtenidos a prueba y error, esto fue porque al inicio cuando el sensor estaba estable, el programa de Python mostraba

variaciones en la figura. Por lo tanto, se decidió calibrar y obtener promedios, de tal manera que se pudiera obtener datos más precisos y una imagen mas estable en el programa de Python.

También, se decidió configurar el BMI para que estuviera muestreando a una frecuencia de 400Hz. Esto se logro hacer escribiendo en unos registros de configuración para el acelerómetro y giroscopio.

Conclusión

En general, el haber realizado el driver de I2C y BMI160 facilito la realización de la práctica, pues se tenía ya la mayor parte de la practica realizada. También, gracias a herramientas como teletype en Atom, se pudo estar codificando desde equipos que no tienen MCU instalado. También, el poder visualizar el movimiento del sensor en el programa Python facilito el desarrollo, pues podíamos ver en forma gráfica si los métodos para poder mejorar la calidad de las muestras estaban funcionando de forma correcta o no. En general no se presentaron muchas dificultades, lo que nos causaba algunas dificultades era el integrar conceptos como semáforos para los drivers desarrollados en la tarea.

Link a repositorio

https://github.com/eecastillo/Practica2_SEII