Juego 2D con interacción mediante gestos con MPU-6050 en Raspberry Pi

Universidad Industrial de Santander

Daniel Alejandro Vega Nieves 2130545
Kevin Johan Oviedo Rueda 2130522

I. INTRODUCCIÓN

En el presente proyecto se desarrollo un sistema que acoplara un sistema de hardware a un sistema de software. El desarrollo se hizo para raspberry pi, ya que esta tarjeta de desarrollo tiene un sistema operativo que administra todo el hardware. Un sistema operativo facilita mucho la interacción entre el usuario y el hardware de la tarjeta, por esto el usuario no debe preocuparse por la arquitectura del sistema. Esto puede cambiar al momento de implementar periféricos, como usb o dispositivos que se puedan conectar a los gpio. La configuración en el sistema operativo de la tarjeta solo puede controlar los elementos agregados en fabrica. Llegado el momento de implementar elementos externos es necesario asegurar que la tarjeta pueda interactuar con dicho elemento e interpretar los datos de la comunicación.

Para lograr una comunicación entre un periférico y una raspberry es necesario configurar los protocolos y la comunicación a un bajo nivel. Es decir que el hardware de la rapsberry debe interactuar con el hardware de los periféricos. Este trabajo es facilitado por el sistema operativo ya que nos permite hacer configuraciones a ese nivel con el lenguaje de programación C. El sistema operativo tiene dos partes importantes que son el espacio del kernel y el espacio del usuario. Para el desarrollo del controlador nos centraremos en el espacio del kernel que es la parte que interactúa con el hardware de la raspberry. Una vez es cargado el controlador en el kernel se puede modificar e interactuar con los registros del procesador. Los registros son memorias en las que se pueden cagar diferentes tipos de datos, estos datos pueden ser modos de configuración para las funcionalidades que tenga el procesador de fabrica. Siendo mas específicos, los registros permiten configurar el funcionamiento de los gpio.

Dentro de las funciones que tiene el procesador para interactuar con los gpio existe un modo que permite usar dos de los gpio como un i2c. Se puede configurar este protocolo de comunicación en dichos pines, los registros a través de una serie de banderas le retorna al sistema algunas señales para alertar que la comunicación a finalizado, o hubo errores de transferencia, o que las memorias de recepción o envió de datos están vacías o llenas.

II. SENSOR MPU-6050

El sensor MPU-6050 contiene un giroscopio y acelerómetro MEMS [1] en un solo integrado. El sensor ofrece una conversion analógica a digital 16-bit para cada canal de sensado, capturando datos en los tres ejes al mismo tiempo.

El chip permite la configuración del sensor y la adquisición de datos por medio de un mapa de registros a lo cuales se puede acceder por medio del protocolo I2C.

El acelerómetro posee un rango de $\pm$ 2g de forma genérica, con la posibilidad de aumentarlo pero perdiendo precision. El giroscopio tiene un rango de $\pm$ 250º\s, con el mismo canje entre rango y precision. Para este proyecto es necesaria la mejor precision posible, y teniendo en cuenta los rangos por defecto, no es necesario realizar un cambio.

La PCB utilizada en este proyecto posee el siguiente breakout:

- VCC - Entrada de alimentación de 3.3V.
- GND - Común.
- SCL - Señal de reloj.
- SDA - Señal de datos.
- XDA - Señal de datos para dispositivo externo.
- XCL - Señal de datos para dispositivo externo.
- ADO - Configuración de dirección I2C del sensor.
- INT - Señal de interrupción del sensor.

Puesto que para el proyecto no era necesario un dispositivo externo al sensor, solo se utilizaron los pines de alimentación y de I2C.

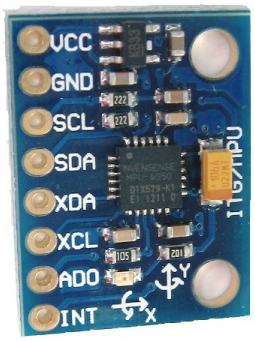

Figura 1. MPU-6050 Breakout

III. DRIVER

Para el desarrollo del proyecto se inició con la implementación de un Linux Driver para el sensor MPU-6050 en la distribución Raspbian OS para una Raspberry Pi 3. Esto con el fin de obtener los datos del sensor desde el nivel mas bajo de hardware y software, que ofrece una mayor rapidez

[1] Microelectromechanical system

en la adquisición de datos.

La Raspberry Pi 3 posee un procesador BCM2837, que ofrece soporte para multiples periféricos, el control de estos periféricos se realiza mediante la configuración de multiples registros dentro del procesador. Durante el desarrollo del proyecto no se encontraba disponible el datasheet para los periféricos del BCM2837, y se utilizó el del procesador BCM2835. El uso de de este datasheet solo implica un cambio en las dirección base de los periféricos, siendo las direcciones relativas de los registros las mismas.

Puesto que el sensor MPU-6050 utiliza el protocolo I2C, se empleará el periférico Broadcom Serial Controller que posee ocho registros 32-bit para el soporte I2C.

La implementación del driver acarrea seguir la estructura que posee un Linux Driver para su carga en el Kernel de sistema. Para acceder a los registros I2C, solo es necesario resaltar la definición de la dirección base de los periféricos del procesador en la función de carga del driver, aparte de los procesos genéricos para la contracción de un driver, como la adjudicación de memoria.

Ya con un Linux Driver que ofrece acceso de lectura y escritura a los registros I2C se procedió a la manipulación de estos registros para la lectura y escritura de los registros dentro del sensor MPU-6050.

La lectura de los registros en el sensor MPU-6050 por medio del I2C se basa en la lectura y escritura de los registros del periférico, el datasheet ofrece una guia para este proceso.

Antes de proceder a la lectura de los datos de giroscopio y acelerómetro, es necesario iniciar el funcionamiento del sensor escribiendo sobre el registro de manejo de energía, puesto que la MPU se inicializa en modo sleep.

La escritura sobre un registro en el sensor tiene los siguientes pasos:

1. Se escribe un dos en el registro I2CDLEN, que indica el numero de bytes a ingresar al FIFO para la transmisión, uno para la dirección del registro en el MPU-6050 y otro para el byte que se va a escribir.
2. Se escribe la dirección del registro en el FIFO.
3. Se escribe la dirección I2C del sensor MPU-6050 en el registro A.
4. Se escribe el byte a transmitir en el FIFO.
5. Se inicia la transmisión en modo escritura con READ en 0 y ST en 1;
6. Se muestrea DONE para confirmar la transmisión.

Es importante enfatizar en el correcto uso de la función *iowrite32*, puesto que se pueden sobreescribir sub registros no deseados fácilmente. Por lo tanto, es necesario escribir todos los registros deseados al instante, no los que se deseen cambiar.

Tras superar la escritura, para la activación del sensor, se procede a la adquisición de datos Para esto se realiza el proceso de lectura de registro dentro del sensor, que tiene la siguiente forma:

1. Se escribe un uno en el registro I2CDLEN, que indica la direccion del registro en el sensor que se va a leer.
2. Se escribe la dirección del registro en el FIFO.
3. Se escribe la dirección I2C del sensor en el registro A.
4. Se inicia la transmisión de la dirección al sensor con READ en 0 y ST en 1;
5. Se muestrea TA hasta que la transferencia termine.
6. Se inicia la recepción del byte con READ en 1 y ST en 1;
7. Se muestrea DONE para confirmar la transmisión.

Con la anterior secuencia se creó una función de lectura con argumento la dirección a leer. Luego esta función es llamada por otra que lee los 12 registros de datos y los devuelve en un vector a la función de lectura del archivo del dispositivo. Dentro de esta función se realiza la codificación del vector transformadolo a tipo char para su envió al espacio del usuario tras una lectura del archivo del dispositivo.

Con esto, se realizaron modificaciones a otras funciones dentro del Linux Driver como el reinicio de los registros I2C en el cierre del archivo del dispositivo, la habilitación automática del sensor una vez sea cargado el driver, etc.

IV. APLICACIÓN

Con la adquisición de los datos de giroscopio y acelerómetro por parte del Linux Driver, y preparados para su lectura por el usuario en el protocolo definido, se procedió a realizar un programa que obtuviera los datos desde el archivo del dispositivo en lenguaje C. El programa abre el archivo, luego lee los doce bytes del sensor y los almacena en memoria para su decodificación , y lo cierra. Con estos doce bytes en el espacio del usuario, se obtuvieron los valores cuantizados sensados por el giroscopio y acelerómetro en los tres ejes uniendo el byte superior e inferior de cada uno. Sin embargo este no es el valor cuantizado, puesto que el dato se encuentra codificado en complemento dos, para esto se paso de una variable sin signo a una con signo. Con los datos listos para su tratamiento, se realizó el ajuste con la sensibilidad dada por el datasheet, para obtener los valores físicos medidos.

Para el manejo de juego es necesario obtener el angulo de inclinación en los ejes X y Y para dar dirección a la nave, y la aceleración en el eje Y para los gestos de saltos laterales. El angulo de inclinación se puede hallar a partir de la aceleración, utilizando la gravedad como referencia, y el giroscopio con su variación en el tiempo.

El angulo en el eje X y Y a partir de la aceleración se obtuvo con las siguientes ecuaciones:

$$AngleAccX = \arctan \frac{AcX}{\sqrt{AcY^2 + AcZ^2}}$$

$$AngleaccY = \arctan \frac{AcY}{\sqrt{AcX^2 + AcZ^2}}$$

El angulo a partir de la aceleración es muy inestable puesto que es posible que aceleraciones diferentes a la gravedad que actúen sobre el sensor produzcan cálculos erróneos.

Luego se tiene giroscopio, que sensa la variación angular en el tiempo. Con esta, es posible encontrar la posición angular utilizando el tiempo entre cada muestra. La lectura del giroscopio es precisa y no es afectada por fuerzas externas. Sin embargo, debido a la integración sobre el tiempo, la medición se desplaza y no vuelve a cero al regresar a su posición original

Teniendo en cuenta estas observaciones se utilizó el filtro complementario, que emplea las dos mediciones para un calculo de posición angular preciso. Este tiene la forma:

$$Angle = 0{,}89 * (Angle + Gy * dt) + 0{,}11 * AngleAcc$$

Este filtro da un peso a cada medida que suma el angulo total, se le da mas peso a la medida del giroscopio por su precision a corto plazo y menos al acelerómetro que sostiene la posición angular sin desplazamientos.

Con los ángulos filtrados ya es posible obtener el gesto para el movimiento bidimensional en el juego, para esto se definieron umbrales que indicaban la inclinación suficiente para considerar el gesto, agregando una lógica que establece el gesto para su posterior codificación.

Para los saltos laterales se utilizó únicamente la medida del acelerómetro, teniendo en cuenta los gestos de desplazamiento, no es posible utilizar solo la aceleración en el eje Y, puesto que se obtendrían gestos erróneos debido a la gravedad. Para solucionar esto se empleo la aceleración total sobre el sistema y se estableció un umbral para descartar la la gravedad.

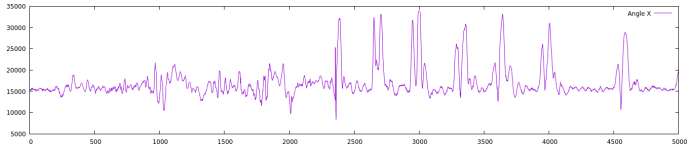

Figura 2. Aceleración total sobre el sensor durante prueba

Debido a la naturaleza del gesto, la aceleración positiva o negativa en el eje Y generaba una aceleración inversa justo después del gesto, como solución se estableció un numero de muestras donde se se tendrían en cuenta estos picos de aceleración después de ser detectado un gesto.

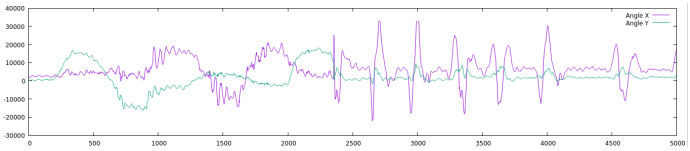

Figura 3. Aceleración en X y Y durante prueba

Una vez detectados estos gestos, se codificaron en dos caracteres, donde el primero cambiaba entre WASD, los símbolos básicos de movimiento en PC, y el segundo LR que indica izquierda o derecha. Estos caracteres se escribieron en un archivo en disco duro para su lectura por el videojuego.

V. INTERFAZ

La programación del videojuego se realizo en c++. Esta fue la aplicación en la que se aplicaron los datos obtenidos del sensor. El juego es un formato árcade y consiste en una nave que avanza a través de algunos niveles y debe destruir algunos enemigos. Se planteo esto como aplicación para hacer un poco mas dinámico el juego. Ya que la respuesta de la nave dependía de los gestos hechos por el usuario. Pudiendo ir a la derecha, izquierda, avanzar y retroceder dependiendo del angulo del sensor. Para la activación de algunas habilidades especiales, se tomaban decisiones en base a la aceleración del sensor. Y para los disparos se instalo un botón.

Para la programación del juego se uso un paquete de librerías de código abierto llamadas SFML. Este conjunto de librerías fueron desatolladas para la creación de juegos 2D en c++. Las librerías facilitan el uso de sonidos, imágenes, fuentes de letras y uso de ventanas. Puesto que c++ es un lenguaje de alto nivel orientado a objetos, cada elemento del juego se trato como un objeto, definiendo sus características y métodos en archivos cabecera. Cada objeto se relacionaba con los otros mediante punteros, facilitando mucho la implementación de los sprites en las animaciones del juego.

SFML consta de 5 módulos principales system, window, graphics, audio y network. Cada uno de estos módulos proporciona un conjunto de clases que facilita el uso de multimedia. Ademas de ser compatible con openGL. También incluye entre sus utilidades herramientas orientadas a los hilos, para la ejecución de código en paralelo. Cabe destacar que SFML es un paquete para el uso de multimedia y no un motor.

En la carpeta NaveSfml puede encontrarse el código fuente del videojuego.

VI. BIBLIOGRAFIA

- Pieter-Jan. (2013, Abril 26). *Reading a IMU Without Kalman: The Complementary Filter* [En linea]. Disponible en: http://www.pieter-jan.com/node/11
- BROADCOM. *BCM2835 ARM Pheripherials*