Técnicas Digitales III: Trabajo Práctico 4

Conexión del ESP32 con una IMU MPU6050 utilizando I2C

Objetivo

El objetivo de este trabajo práctico es que los estudiantes se familiaricen con la interfaz I2C al conectar un ESP32 con una IMU MPU6050. Además, realizarán la transferencia de los datos adquiridos a una PC, en una cadena específica por UART, utilizando un conversor SERIE-USB y un programa de adquisición de datos, la catedra recomienda el RealTerm.

Finalmente, los datos serán procesados y graficados en un programa para procesamiento matemático, puede ser Matlab o Scilab.

Para preparar los datos para el software matemático se recomienda el uso de Python.

Materiales Necesarios

- Placa ESP32
- Sensor IMU MPU6050
- Cables de conexión (jumpers)
- Computadora.
- Interfaz USB a serie TTL.
- ESP-IDF instalado en la computadora.
- Software RealTerm instalado en la computadora (es el recomendado, pero no obligatorio)
- Matlab instalado en la computadora. O similar.
- Python instalado en la computadora. O el software de alto nivel seleccionado por el estudiante.

Esta prohibido el uso de Arduino en cualquiera de sus formas.

Instrucciones

1. **Preparación del Entorno**
    - Asegúrese de tener instalado el entorno de desarrollo ESP-IDF en su computadora. Siga las instrucciones oficiales de instalación disponibles en el sitio web de Espressif.
    - Instale el software RealTerm, disponible en este enlace.
    - Asegúrese de tener Matlab y Python instalados en su computadora.
2. **Conexión del Hardware**
    - Conecte la ESP32 a su computadora mediante un cable USB.
    - Conecte el sensor MPU6050 a la ESP32 utilizando la interfaz I2C. Las conexiones recomendadas son:
        - VCC del MPU6050 a 3.3V en la ESP32
        - GND del MPU6050 a GND en la ESP32
        - SDA del MPU6050 a GPIO21 en la ESP32
        - SCL del MPU6050 a GPIO22 en la ESP32
3. **Configuración del Proyecto en ESP-IDF**

- Abra una terminal y navegue hasta el directorio donde desea crear su proyecto.

4. **Arquitectura del software.**
   - Debe realizar el diagrama que represente la arquitectura de software del sistema, indicando en cada bloque las funciones que deben ser desarrolladas, incluyendo los parámetros de entrada y salida.
   - La transmisión de datos debe ser a una frecuencia constante de 10Hz.
   - La cadena de salida debe tener longitud constante.
   - Con los datos anteriores debe determinar la frecuencia de transferencia de datos, el formato es 8,n,1.
5. **Implementación del Código**
   - Escriba el código en C para inicializar y leer datos del sensor MPU6050 a través de la interfaz I2C.
   - Formatee los datos adquiridos en una cadena de datos donde:
     - El primer dato es 0x1B.
     - El segundo dato es un contador de 0 a 255.
     - Los siguientes datos son las lecturas de aceleración en los ejes X, Y, Z y del giroscopio en los ejes X, Y, Z.
   - Envíe esta cadena de datos a través del puerto serie UART.
   - Asegúrese de utilizar las funciones de la librería ESP-IDF para manejar I2C y UART.
6. **Visualización y Almacenamiento de Datos con RealTerm**
   - Abra RealTerm en su computadora.
   - Configure RealTerm para leer el puerto serie al que está conectada la ESP32 (usualmente COMx en Windows).
   - Configure la velocidad de baudios en la frecuencia determinada (o la velocidad que haya configurado en su código).
   - Configure RealTerm para guardar los datos recibidos en un archivo en formato hexadecimal.
   - Observe y almacene los datos del sensor MPU6050 que se envían desde la ESP32.
7. **Procesamiento y Graficación de Datos en Matlab**
   - Utilice Python para leer el archivo guardado por RealTerm, convertir los datos y prepararlos para su procesamiento en Matlab.
   - Importe los datos a Matlab y grafíquelos para analizar las lecturas de aceleración y giroscopio.
   - Asegúrese de documentar el proceso y presentar los gráficos resultantes en un informe.

     Ejemplo de script en Python para convertir los datos:

```
python
Copiar código
import numpy as np

# Leer archivo hexadecimal
with open('data.hex', 'r') as file:
    hex_data = file.read().split()

# Convertir datos a enteros
data = [int(x, 16) for x in hex_data]
```

```
# Guardar los datos en un archivo CSV para Matlab
np.savetxt('data.csv', data, delimiter=',')
```

8. **Informe Final**
    - Documente todo el proceso en un informe. Incluya Arquitectura de software, diagramas en bloques del sistema, diagrama de conexión, scripts de Python y gráficos de Matlab.
    - Asegúrese de analizar y discutir los resultados obtenidos.

Criterios de Evaluación

- Arquitectura de software y diagrama en bloques.
- Correcta conexión y configuración del hardware.
- Funcionalidad del código desarrollado.
- Correcta visualización y almacenamiento de los datos en RealTerm.
- Eficacia del script de Python para convertir los datos.
- Calidad de los gráficos y análisis en Matlab.
- Calidad y claridad del informe presentado.
- Coloquio.