UNIVERSIDAD SERGIO ARBOLEDA

SISTEMAS EMBEBIDOS

INGENIERÍA ELECTRÓNICA

WENDY JOHANNA CHAPARRO – CRISTOPHER RAMÍREZ RAMÍREZ

INFORME DE LABORATORIO #5

En este laboratorio se diseñó e implementó un protocolo de transmisión con el objetivo de enviar los datos de las RPM (revoluciones por minuto) para su visualización gráfica a través de python. Para esta tarea, se utilizó la placa de desarrollo STM32F411 y el entorno de desarrollo STM32CubeIDE, herramientas que permitieron la configuración, envío y recepción de los datos con este protocolo personalizado.

En este proceso se busca cómo implementar protocolos de comunicación en sistemas embebidos, muy utilizados para aplicaciones más avanzadas en el control y monitoreo de sistemas mecánicos y electrónicos.

**SENSORES Y MODULOS IMPLEMENTADOS:**

* **Sensor infrarrojo de herradura KLH512:** El sensor de herradura KLH512 es un sensor fotoeléctrico que responde al cambio en la intensidad de la luz. Estos sensores requieren de un componente emisor que genera la luz, y un componente receptor que percibe la luz generada por el emisor. Están diseñados especialmente para la detección, clasificación y posicionado de objetos; la detección de formas, colores y diferencias de superficie, incluso bajo condiciones ambientales extremas. Los sensores de luz se usan para detectar el nivel de luz y producir una señal de salida representativa respecto a la cantidad de luz detectada. [1]
* **Modulo puente H TB6612FNG:** Es una pequeña placa integrada por un circuito integrado TB6612FNG que permite tener la manipulación de motores de DC. El controlador TB6612FNG es un driver de motores que puede controlar hasta dos motores DC a una corriente constante de 1.2 Amp .  Las dos salidas de motor (A y B) pueden controlarse por separado, la velocidad de cada motor se controla con una señal de entrada PWM de hasta 100 kHz. Este controlador integra en su placa dos señales de entradas IN1 y IN2, que se utilizan para controlar el motor en uno de los cuatro modos, como CW, CCW, corto freno y modo de parada. [2]

**LIBRERÍAS IMPLEMENTADAS:**

1. **usb\_device.h**

Esta librería es la interfaz general para la configuración y gestión del dispositivo USB en la placa STM32. Se genera automáticamente cuando se habilita el dispositivo USB en STM32CubeIDE.

* **Inicialización del USB:** Proporciona las funciones necesarias para inicializar la pila USB (USB stack), que se encarga de gestionar la comunicación USB entre el microcontrolador y el host (por ejemplo, una PC).
* **Configuración de periféricos USB:** Esta biblioteca gestiona la configuración de los endpoints, el tipo de dispositivo (como HID, CDC, MSC), y otros parámetros necesarios para el funcionamiento USB.
* **Eventos y manejo de interrupciones:** Gestiona los eventos del USB, como la conexión y desconexión, o los paquetes de datos recibidos.

1. **usbd\_cdc\_if.h**

Esta librería está relacionada con el uso del USB en modo CDC (Communications Device Class), un protocolo que permite que el dispositivo STM32 actúe como una interfaz de comunicación serial (similar a un puerto COM virtual en la computadora).

* **CDC (Class Device Class):** Permite que el microcontrolador emule un puerto serial sobre USB, lo que facilita la transmisión y recepción de datos como si fuera una comunicación UART.
* **Interfaz de funciones:** Esta biblioteca ofrece funciones específicas para enviar y recibir datos a través del USB en modo CDC, como CDC\_Transmit\_FS, que permite transmitir datos desde el STM32 hacia el host (PC) a través del puerto USB.
* **Callback functions:** Proporciona puntos de anclaje para gestionar eventos de comunicación, como la recepción de datos a través del USB (CDC\_Receive\_FS), lo que facilita la integración de la pila USB con la lógica de la aplicación.

El protocolo de comunicación ha sido diseñado para enviar  datos relacionados con las RPM (revoluciones por minuto) y el tiempo de respuesta en la que el motor se demora en llegar a las revoluciones máximas. El protocolo sigue un formato estructurado y secuencial para garantizar la recepción de la información durante la transmisión.

El proceso de comunicación comienza con el envío de un byte de inicio, cuyo valor es 0xF, que indica el comienzo de un nuevo paquete de datos. A continuación, se envía el tamaño total del paquete en 2 bytes, permitiendo así una mayor flexibilidad en la cantidad de datos que se pueden transmitir.

Los datos clave de la transmisión son las RPM y el tiempo de respuesta, que se envían inmediatamente después del tamaño. Para asegurar que los datos lleguen de manera correcta y sin errores, se incorpora un byte de verificación o *checksum*. Este byte se calcula a partir de los valores de los bytes enviados con una suma, permitiendo verificar los datos en la recepción.

Finalmente, el protocolo se cierra con el envío de un byte de finalización, cuyo valor también es 0xF, señalando la conclusión de la transmisión.

**Estructura del Protocolo**:

1. **Byte de inicio**: Valor fijo 0xF.
2. **Tamaño del paquete**: 2 bytes que indican la longitud de los datos.
3. **Datos**: RPM y tiempo de respuesta.
4. **Checksum**: 1 byte para validar la integridad del paquete.
5. **Byte de finalización**: Valor fijo 0xF.

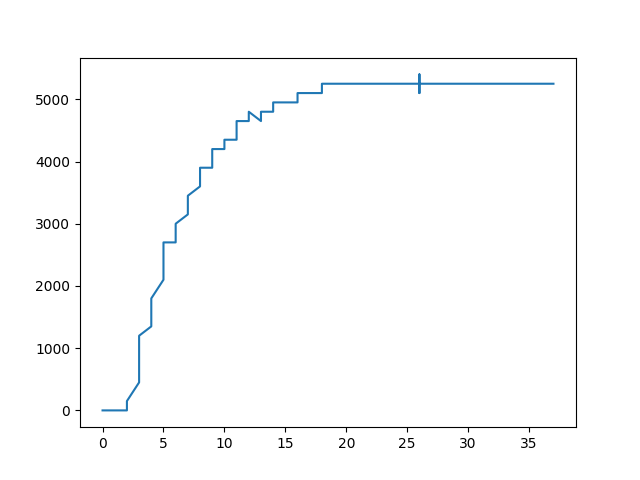
**Overhead del Protocolo**: El *overhead* de este protocolo se refiere a los bytes adicionales que no forman parte de los datos principales (RPM y tiempo de respuesta), pero que son necesarios para garantizar la integridad y sincronización de la comunicación. En este caso, el *overhead* está compuesto por:

* 1 byte de inicio.
* 2 bytes para el tamaño del paquete.
* 1 byte de *checksum*.
* 1 byte de finalización.

En total, el *overhead* del protocolo es de 5 bytes.

Lo datos transmitidos fueron enviados en formato **Little Endian**, los bytes menos significativos (los que tienen menor peso) se almacenan primero, es decir, en la dirección de memoria más baja.

**RESULTADOS:**



**Grafica de RPM en función del tiempo**

En la gráfica se obtiene que las revoluciones por minuto máximas tienen un asentamiento en 17 segundos aproximadamente y el tiempo que se demora el motor en responder a la señal de activación es de 2 segundos aproximadamente.

**CONCLUSIONES:**

1. Es importante tener presente que los datos de inicio y fin se deben diferenciar ya que asegura que los paquetes de datos sean reconocidos correctamente por el receptor, minimizando errores y mejorando la eficiencia en la transmisión.
2. La incorporación de un checksum como mecanismo de validación garantiza que los datos enviados (RPM y tiempo de respuesta) que lleguen al receptor sin alteraciones, ofreciendo un nivel adicional de seguridad y confiabilidad en la transmisión.
3. Al incluir dos bytes para definir el tamaño del paquete, el protocolo se adapta a diferentes volúmenes de datos, lo que lo hace versátil ante grandes volúmenes de datos.
4. Conocer términos como Overhead y formatos de transmisión de datos permite un óptimo desarrollo y creación de protocolos ya que son fundamentales a la hora de recibir los datos necesarios, asegurando la sincronización y corrección en caso de errores.

**BIBLIOGRAFÍA:**

[1] «Sensor de Herradura KLH512 - Moviltronics», *Moviltronics*, 3 de mayo de 2024. Disponible en: <https://moviltronics.com/tienda/sensor-klh512/>

[2] «Doble Puente H TB6612FNG», *UNIT ELECTRONICS*. Disponible en: <https://uelectronics.com/producto/doble-puente-h-tb6612fng/?srsltid=AfmBOor2_Fn69ubd_3kAXtR1Zh-oOXWBrBFq8dmfOs97bHHpi7R7ogZn>

[3] alldatasheet.com, «TB6612FNG PDF». Disponible en: <https://www.alldatasheet.com/datasheet-pdf/view/807693/TOSHIBA/TB6612FNG.html>