INGENIERÍA MECATRÓNICA

IMEC

Universidad Católica Boliviana "San Pablo"

Departamento de Ciencias de la Tecnología e Innovación – Tarija

Práctica 04 – Comunicación Digital: I²C y SPI

Asignatura:	(IMT-222 - Sistemas Embebidos I)
Docente:	(Elmer Alan Cornejo Quito)
Correo:	(ecornejo@ucb.edu.bo)
Lugar / Fecha:	Tarija, 17 de noviembre de 2025

Dificultad: Baja **Requisito:** Sin supervisión obligatoria

Formato UCB – Versión base (2025)

Índice

1. Objetivos

Implementar la comunicación digital entre un microcontrolador y un sensor mediante los protocolos I^2C o SPI. Los objetivos específicos son:

1. Configurar la interfaz de comunicación seleccionada (I^2C o SPI).
2. Leer y mostrar los datos obtenidos desde un sensor digital en el monitor serial.
3. Comprender el flujo de transmisión y recepción de datos a nivel de bit.
4. Documentar y explicar detalladamente el funcionamiento del protocolo implementado.

2. Materiales y Equipos

A continuación se enumeran los materiales, instrumentos y herramientas de software necesarios para el desarrollo de la práctica de comunicación digital mediante el protocolo I^2C.

2.1. Hardware

1. Arduino UNO R3 (microcontrolador principal para la implementación del protocolo I^2C).
2. Modulo digital BH1750 (sensor de intensidad lumínica con interfaz I^2C y resistencia de pull-up integradas).
3. Protoboard (breadboard) para montaje de conexiones temporales.
4. Cables Dupont (macho-macho) para la conexión eléctrica entre el sensor y el microcontrolador.
5. Fuente de alimentación de 5V (proporcionada por el Arduino UNO vía USB).
6. Cable USB tipo A-B para alimentación y comunicación con el ordenador.
7. Multímetro digital (para verificación de continuidad y voltajes).

2.2. Software

1. Arduino IDE (entorno digital de programación y carga del firmware).
2. Librería BH1750 para Arduino (para facilitar la comunicación del sensor).
3. Librería Wire.h (Arduino I^2C library), utilizada para gestionar la comunicación I^2C con el sensor BH1750.
4. Herramienta de monitoreo serial integrada en Arduino IDE (Serial Monitor o Serial Plotter).
5. Drivers USB del Arduino UNO.
6. Repositorio GitHub personal para subir el código fuente.

3. Metodología

La práctica implementa la comunicación digital mediante el protocolo I^2C entre un microcontrolador Arduino UNO y el sensor de iluminación BH1750. El procedimiento se organiza en tres fases, tal como indica la guía de laboratorio (Configuración del entorno, Implementación de la comunicación y Presentación/Análisis), y sigue el flujo definido por la Máquina de Estados (FSM) mostrada en la Figura **??**: *Configuración*, *Lectura*, *Visualización* y *Error*.

3.1. Fase 1 — Configuración del entorno (Estado *Configuración*)

En esta fase se prepara el hardware y el software necesarios para iniciar la comunicación I^2C. Todas las acciones descritas aquí corresponden al estado *Configuración* de la FSM.

1. Identificación del tipo de comunicación

1. Consultar la hoja de datos del BH1750 y confirmar que utiliza el protocolo I^2C.
2. Identificar las líneas necesarias: SDA (datos), SCL (reloj), VCC y GND.

2. Conexión del sensor al Arduino UNO

1. Disponer Arduino UNO, el módulo BH1750 y la protoboard sobre una superficie estable.
2. Realizar las conexiones según el bus I^2C:

Arduino UNO	BH1750
5V	VCC
GND	GND
A4	SDA
A5	SCL
ADDR	GND (dirección 0x23)

3. Verificar con el multímetro la continuidad de las conexiones y la ausencia de cortocircuitos.
4. Energizar la placa mediante el cable USB y comprobar:
   - Tensión en VCC del BH1750 ≈ 5 V.
   - Tensión en SDA y SCL en reposo cercana a VCC (pull–up del módulo).

3. Configuración del entorno de desarrollo y detección del dispositivo

1. Abrir el Arduino IDE y seleccionar:
   - **Placa**: Arduino UNO.
   - **Puerto**: COM asignado a la placa.
2. Incluir las librerías:

- `Wire.h` (incluida por defecto).
- Librería `BH1750` desde el Gestor de Librerías.

3. Cargar y ejecutar un *I^2C Scanner* para verificar la presencia del sensor en el bus.
4. Observar en el Monitor Serial la dirección detectada:
   - 0x23 cuando ADDR = LOW (configuración utilizada).
   - 0x5C en caso de que el pin ADDR se conecte a VCC.

En la FSM, si el escáner detecta correctamente la dirección del BH1750, se genera el evento **sensorOk**, que provoca la transición desde el estado *Configuración* al estado *Lectura*. Si no se detecta ninguna dirección válida, el grupo debe revisar cableado y alimentación antes de continuar; esta situación se considerará posteriormente como evento **sensorFail** hacia el estado *Error*.

3.2. Fase 2 — Implementación de la comunicación (Estados *Lectura*, *Visualización* y *Error*)

En esta fase se inicializa el bus I^2C en el firmware, se implementa la lectura de datos del sensor y se muestran los resultados en el Monitor Serial. Operativamente, el sistema alterna entre los estados *Lectura* y *Visualización*, y ante fallos pasa al estado *Error*, de acuerdo con la FSM.

1. Inicialización del bus de comunicación (entrada al estado *Lectura*)

1. Crear un nuevo sketch en Arduino IDE.
2. Incluir las librerías en el código:
   - `#include <Wire.h>`
   - `#include <BH1750.h>`
3. En la función `setup()`, implementar:
   a) `Wire.begin();` para habilitar el bus I^2C del ATmega328P.
   b) `Serial.begin(baudrate);` para habilitar el puerto serie.
   c) `sensor.begin();` (según la librería utilizada) para iniciar comunicación con el BH1750.
   d) Mensajes de diagnóstico por Monitor Serial indicando el resultado de la inicialización.

Si el sensor responde correctamente durante la inicialización, el sistema permanece en el estado *Lectura*. Si la librería indica que no hay respuesta del dispositivo (por ejemplo, ausencia de ACK), se considera que ocurrió el evento **sensorFail** y se transita al estado *Error*.

2. Lectura periódica de datos (estado *Lectura*)

1. En la función `loop()`, implementar un ciclo de lectura periódica del BH1750:
   a) Solicitar la medición de iluminación al sensor.
   b) Almacenar el valor leído en una variable (por ejemplo, `lux`).
   c) Verificar que el valor sea válido (no nulo, no igual a 65535, etc.).
2. Cuando se obtiene un dato válido, se considera que ocurre el evento **dataReady** y la FSM transita al estado *Visualización*.

3. Visualización de resultados (estado *Visualización*)

1. Imprimir en el Monitor Serial:
   - Mensajes de estado ("BH1750 operativo", "Leyendo datos...").
   - El valor de `lux` con su unidad correspondiente.
2. De forma opcional, indicar si el nivel de iluminación es bajo, medio o alto, únicamente con fines demostrativos.
3. Tras mostrar la información, se considera que ocurre el evento **stop**, que devuelve a la FSM al estado *Lectura* para realizar una nueva adquisición.

4. Manejo de errores (estado *Error*)

Cuando durante la inicialización o durante el ciclo de lectura se detectan anomalías (por ejemplo, ausencia de respuesta del sensor, lecturas constantes en 0 o 65535), el sistema entra en el estado *Error*, asociado al evento **sensorFail**.

1. Detener temporalmente el lazo de lectura.
2. Verificar:
   - Correcta conexión de SDA y SCL.
   - Presencia de tensión de alimentación en el BH1750.
   - Dirección I^2C configurada en el código (0x23 u otra según el hardware).
3. Una vez corregido el problema, generar un *reset* del sistema (reinicio de la placa o reinicio del programa). Este evento **reset** provoca la transición desde el estado *Error* de vuelta al estado *Configuración*, repitiendo la Fase 1 hasta recuperar la comunicación.

3.3. Fase 3 — Presentación y análisis

Finalmente, cada grupo debe preparar una presentación donde:

1. Explique la estructura del código y cómo se implementa el flujo definido por la Máquina de Estados (transiciones *Configuración* → *Lectura* → *Visualización* y manejo del estado *Error*).
2. Describa las funciones principales utilizadas del protocolo I^2C (`Wire.begin()`, funciones de lectura de la librería BH1750, etc.).
3. Compare brevemente las características de I^2C y SPI, destacando:
   - Número de líneas utilizadas.
   - Forma de direccionamiento.
   - Ventajas y desventajas en aplicaciones típicas.
4. Detalle el rol de las señales SCL y SDA en el caso de I^2C (y, de forma comparativa, SCK/MISO/MOSI/CS en SPI), relacionándolas con el intercambio de datos mostrado en la práctica.

4. Cuestionario

1. ¿Qué diferencias existen entre los protocolos I^2C y SPI en cuanto a número de líneas, velocidad y direccionamiento?

   Número de líneas:

   - I^2C:
     - utiliza 2 líneas compartidas
     - SDA (datos)
     - SCL (reloj)
   - SPI:
     - utiliza 4 líneas básicas
     - MOSI
     - MISO
     - SCK
     - CS/SS (por cada esclavo)

   Velocidad:

   - I^2C: típicamente 100 kHz (Standard), 400 kHz (Fast), hasta 3.4 MHz en modos especiales.
   - SPI: velocidades mayores, frecuentemente entre 5–20 MHz, dependiendo del hardware.

Direccionamiento:

- I^2C: usa direccionamiento de 7 o 10 bits, varios dispositivos en el mismo bus con direcciones únicas.
- SPI: no tiene direccionamiento interno; la selección del esclavo se hace mediante una línea CS por dispositivo.

2. ¿Qué función cumple la señal de ACK (Acknowledge) en la comunicación I^2C?
   El ACK es un bit que envía el dispositivo receptor (esclavo o maestro) para indicar:
   - que recibió correctamente el byte transmitido,
   - que está presente y operativo en el bus,
   - y que el transmisor puede continuar enviando más datos.

   Sin ACK, la transmisión se considera fallida.

3. ¿Qué sucede si el esclavo no envía un ACK durante una transmisión I^2C?
   Si no aparece el ACK (NACK):
   - el maestro interpreta que no hay dispositivo respondiendo en esa dirección,
   - o que el esclavo no pudo procesar el byte,
   - entonces el maestro detiene la transmisión,
   - y genera una condición de STOP o reintenta la comunicación.

   En firmware esto suele detectarse como error, lectura nula o bloqueo del bus.

4. ¿Cuál es la función de la línea Chip Select (CS) en el protocolo SPI?
   La línea Chip Select (CS) o Slave Select (SS):
   - Activa el esclavo con el que se desea comunicar,
   - Permite que solo un esclavo escuche el bus y responda,
   - Se coloca en nivel bajo (0) para habilitar al dispositivo,
   - En nivel alto (1) el esclavo queda desconectado del bus.

   Sin CS no hay forma de manejar múltiples esclavos SPI.

5. ¿Qué ventajas presenta SPI respecto a I^2C en aplicaciones de alta velocidad?
   SPI ofrece:
   - Mayor velocidad de transferencia (varios MHz).
   - Transmisión full-duplex (envía y recibe simultáneamente).
   - Menor latencia, ya que no hay arbitraje ni bits de ACK.

- Estructura más simple en la capa de protocolo.

Por eso SPI es preferido en memorias flash, pantallas, ADCs rápidos, DACs de precisión, etc.

6. ¿Qué limitaciones tiene cada protocolo al conectar múltiples dispositivos?
   I^2C:

   - Requiere que todas las direcciones sean diferentes.
   - Limitado por las resistencias pull-up y la capacitancia del bus.
   - El número de dispositivos afecta la velocidad máxima.

   SPI:

   - Se necesita una línea CS por cada esclavo, lo que incrementa el cableado.
   - Puede volverse poco práctico con muchos dispositivos.
   - No tiene direccionamiento interno, todo depende del maestro.

7. ¿Cómo podría verificarse mediante osciloscopio o analizador lógico la comunicación realizada?

   Con osciloscopio:

   - Observar las señales SDA/SCL (I2C) o MOSI/MISO/SCK (SPI).
   - Verificar niveles lógicos, tiempos de subida, frecuencia del reloj.
   - Identificar condiciones de START/STOP y pulsos de ACK.

   Con analizador lógico:

   - Capturar las líneas digitales del bus.
   - Decodificar automáticamente el protocolo (I^2C/SPI).
   - Leer bytes transmitidos, direcciones, ACK/NACK.
   - Detectar errores o colisiones.

5. Rúbrica de Evaluación

- Funcionamiento del sistema: comunicación y lectura del sensor (30 %).
- Código y documentación: claridad, modularidad y comentarios (25 %).
- Presentación en clase: explicación del protocolo y demostración práctica (25 %).
- Informe y formato: estructura, redacción y coherencia (20 %).

La evaluación será grupal y se realizará sobre 100 puntos según los criterios anteriores.

6. Seguridad y Buenas Prácticas

- Verificar el tipo de lógica (3.3 V o 5 V) antes de conectar el sensor al microcontrolador.
- Usar resistencias de pull-up adecuadas para el bus I2C.
- No realizar cambios de conexión con el sistema energizado.
- Mantener un orden en el área de trabajo y cuidado del material de laboratorio.

7. Formato y Entrega

- Trabajo en grupos de máximo cuatro integrantes.
- Subir el código fuente al repositorio personal de GitHub y a la plataforma LMS.
- Entregar un informe en formato PDF con la estructura institucional.
- Incluir fotografías o capturas de la comunicación funcional.
- Preparar una breve presentación (5-7 minutos) para la exposición grupal.