CS3262
Embedded Computer Networks
Lab 03 - SPI Communication
210113L - Akindu Induwara

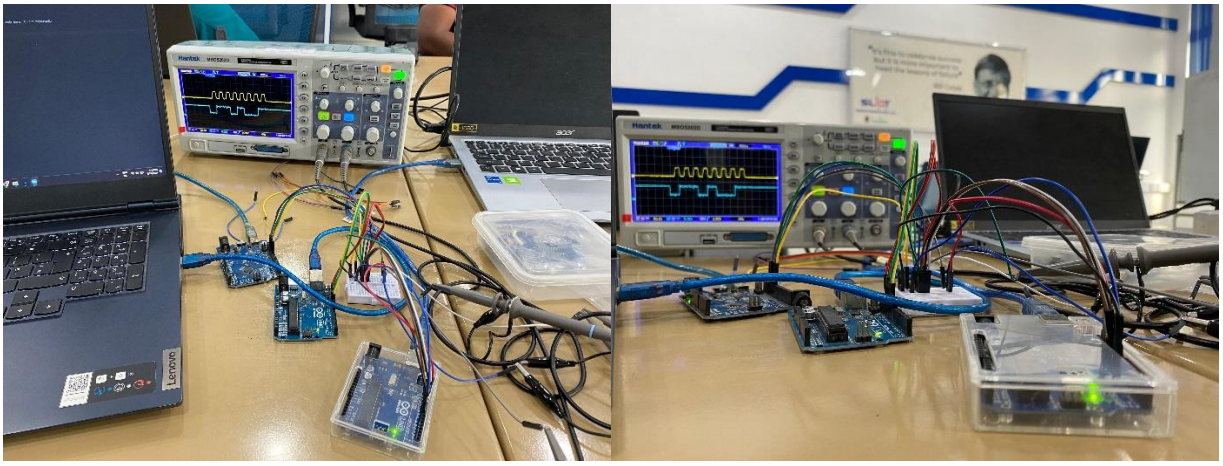
Group Members

Akindu Induwara

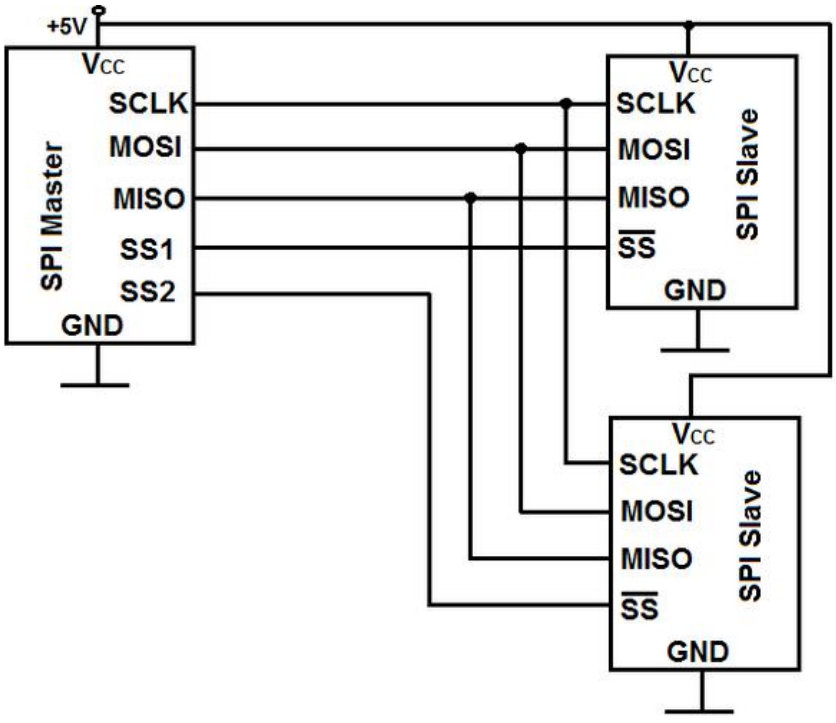
Nimeth Menuka

Tharusha Wijewardhana

SPI Operation



This laboratory activity involves connecting three Arduino boards using the Serial Peripheral Interface (SPI) protocol.
One board will act as the master, controlling the communication flow and data exchange, while the remaining two boards will function as slaves, responding to the master's requests and providing or receiving data as instructed.



The Serial Peripheral Interface (SPI) protocol utilizes four dedicated pins for data transmission and slave selection in Arduino.

Pin	Description	Default Pin Number in Arduino
SCLK	Clock Signal	13
MOSI	Master output Slave input	12
MISO	Master input Slave output	11
SS	Slave Select	10

Additional pins can be used for slave select when connecting multiple slaves, as defined in the program code.
The Serial Peripheral Interface (SPI) protocol employs a dedicated register called the Serial Peripheral Data Register (SPDR) for data management. This register serves a dual purpose.

Data Transmission

When the master device initiates communication, the data intended to be sent to the slave is first loaded into the SPDR. Upon transmission, the data is shifted out bit-by-bit from the SPDR via the MOSI (Master Out Slave In) line.

Data Reception

During a communication cycle, the slave device also utilizes the SPDR. Simultaneously as the master transmits data, the slave receives incoming data bit-by-bit through the MISO (Master In Slave Out) line. This received data is then temporarily stored in the SPDR until the slave program retrieves and processes it.

Bit	7	6	5	4	3	2	1	0	
0x2E (0x4E)	**MSB**							**LSB**	**SPDR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

Master to Slave

his image depicts an SPI (Serial Peripheral Interface) communication scenario where a master Arduino board transmits the character 'I' to a slave device.

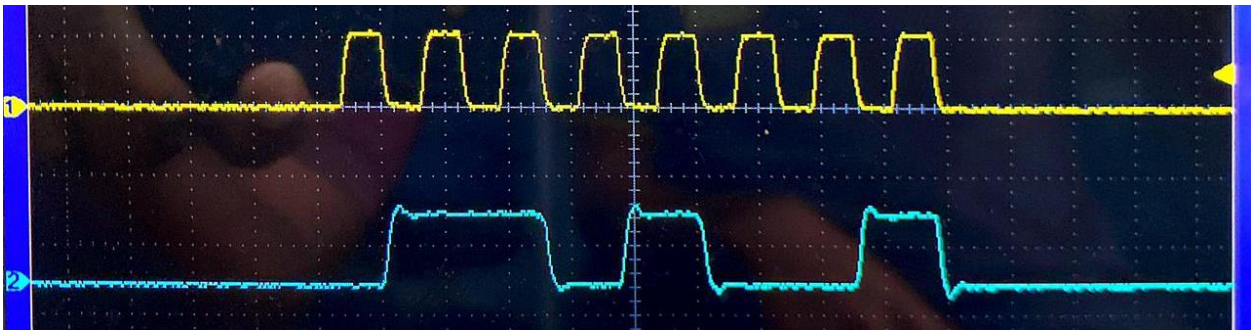
Key Observations:

- MOSI Pin: Set to LOW, indicating data transmission mode for the master.
- MISO Pin: Set to HIGH, likely due to the slave's internal configuration or lack of current data to send.
- Blue Line (Data): Represents the transferred data bits. Decoding reveals the binary representation of 'I': 01101001.
- Yellow Line (Clock): Represents the SPI clock signal, synchronizing the bit-by-bit transmission.

Explanation:

- The master initiates communication and sets the MOSI pin to LOW to signal data transmission.
- The master loads the binary representation of 'I' (01101001) into its SPI Data Register (SPDR).
- The master asserts the clock signal (yellow line), triggering the transfer of each data bit from its MOSI pin (blue line) to the slave's MISO pin.
- Simultaneously, the slave receives the incoming bits and stores them sequentially in its SPDR.

Slave to Master



This image illustrates a data transfer scenario using the Serial Peripheral Interface (SPI) protocol, where a slave Arduino board sends the integer value 105 to the master.

Key Points:

- MISO Pin: Set to LOW, indicating data transmission mode for the slave.
- MOSI Pin: Set to HIGH, likely due to the master's internal configuration or lack of data to send at this moment.
- Blue Line (Data): Represents the transferred data bits. Decoding reveals the binary representation of the integer 105: 01101001.
- Yellow Line (Clock): Represents the SPI clock signal, synchronizing the bit-by-bit transmission.

Explanation:

- The slave prepares the data to be sent, converting the integer 105 into its binary representation (01101001) and loading it into its SPI Data Register (SPDR).
- The slave asserts the MISO pin to LOW to signal data transmission.
- The master, upon detecting the low state on the MISO pin, synchronizes with the clock signal (yellow line).
- The slave transmits each bit of the data from its SPDR sequentially, sending them out through the MISO pin (blue line).
- Simultaneously, the master receives the incoming bits and stores them in its SPDR.

I2C vs SPI

Feature	I2C (Inter-Integrated Circuit)	SPI (Serial Peripheral Interface)
Type of Communication	Multi-master, Multi-slave	Single Master, Multi-slave
Number of Data Lines	Two: SCL (clock), SDA (data)	Four: SCK (clock), MOSI (master out), MISO (master in), (slave select)
Speed	Slower (typically 100-400 KHz, max 5MHz)	Faster (up to 80 MHz or higher)
Complexity	Simpler	More complex
Full Duplex:	No	Can be in some implementations
Addressing	Each device has a unique 7-bit address	Master selects specific slave using an additional Slave Select (SS) line for each slave
Wire Length	Can be longer due to lower speed	Limited due to higher speed and signal integrity requirements
Error Detection	Yes: Uses checksums for error detection	Depends on implementation, may not have built-in error detection
Power Consumption	Lower	Higher due to higher speed
Typical Applications	Low-speed peripherals (e.g., temperature sensors, RTCs)	High-speed peripherals (e.g., displays, memory cards)