# AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

**Faculty of Engineering**

# Lab Report

## Experiment # 06

**Title:** Communication between two Arduino Boards using SPI.

| | | | |
|---|---|---|---|
| **Date of Perform** | 12 March 2023 | **Date of Submission** | 19 March 2023 |
| **Course Title** | MICROPROCESSOR AND EMBEDDED SYSTEMS | | |
| **Course Code** | COE3104 | **Section** | C |
| **Semester** | Spring 2022-23 | **Degree Program** | BSc in CSE |
| **Course Teacher** | **RICHARD VICTOR BISWAS** | | |

**Declaration and Statement of Authorship:**

1. I/we hold a copy of this Assignment/Case-Study, which can be produced if the original is lost/damaged.
2. This Assignment/Case-Study is my/our original work and no part of it has been copied from any other student's work or from any other source except where due acknowledgment is made.
3. No part of this Assignment/Case-Study has been written for me/us by any other person except where such collaboration has been authorized by the concerned teacher and is clearly acknowledged in the assignment.
4. I/we have not previously submitted or currently submitting this work for any other course/unit.
5. This work may be reproduced, communicated, compared, and archived for the purpose of detecting plagiarism.
6. I/we give permission for a copy of my/our marked work to be retained by the Faculty Member for review by any internal/external examiners.
7. I/we understand that Plagiarism is the presentation of the work, idea, or creation of another person as though it is your own. It is a form of cheating and is a very serious academic offense that may lead to expulsion from the University. Plagiarized material can be drawn from, and presented in, written, graphic and visual forms, including electronic data, and oral presentations. Plagiarism occurs when the origin of the source is not appropriately cited.
8. I/we also understand that enabling plagiarism is the act of assisting or allowing another person to plagiarize or copy my/our work.

* Student(s) must complete all details except the faculty use part.
** Please submit all assignments to your course teacher or the office of the concerned teacher.

## Group # 01

| Sl No | Name | ID | Department |
|---|---|---|---|
| 1 | Zaid Amin **Rawfin** | 20-42459-1 | CSE |
| 2 | Abzana Sultan **Ira** | 20-41973-1 | CSE |
| 3 | MD. Sanjid Bin Karim **Sezan** | 19-41702-3 | CSE |
| 4 | Abdur Rahman **Swapnil** | 18-38950-3 | CSE |

| ***Faculty use only*** | | |
|---|---|---|
| FACULTY COMMENTS | **Marks Obtained** | |
| | **Total Marks** | |

# Table of Contents

# Experiment Title

Communication between two Arduino Boards using SPI.

# Abstract

In this experiment, we will explore the SPI protocol and how to use it with Arduino. We will use SPI to establish communication between two Arduino boards. One Arduino will act as a master, while the other will be the slave. We will connect two LEDs and push buttons to each Arduino. To demonstrate SPI communication, we will control the master-side LED with the push button on the slave side and vice versa using the SPI protocol.

# Introduction

Microcontrollers use various communication protocols to interact with sensors and modules. Serial communication is one of the most widely used techniques, and it involves transmitting data one bit at a time over a communication channel or bus. There are several serial communication protocols, including UART, CAN, USB, I2C, and SPI communication.

The Serial Peripheral Interface (SPI) is a synchronous serial communication protocol. SPI was first developed by Motorola in 1970 and is now widely used in embedded systems. SPI is a full-duplex connection, which means that data can be sent and received simultaneously. This feature enables a master to send data to a slave, and the slave can send data back to the master. The clock is essential in SPI communication since it synchronizes the data transfer.

# Theory and Methodology

A SPI has a master/Slave communication by using four lines. A SPI can have only one master and can have multiple slaves. A master is usually a microcontroller and the slaves can be a microcontroller, sensors, ADC, DAC, LCD etc.

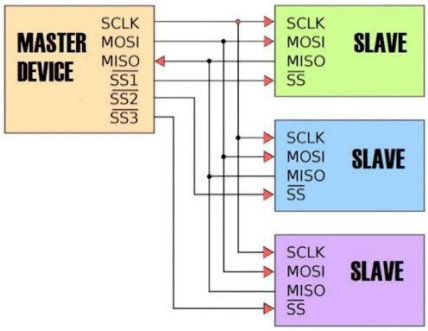



Fig. 1 block diagram representation of SPI Master with several slaves.

SPI has following four lines MISO, MOSI, SS, and CLK

- **MISO (Master in Slave Out)** - The Slave line for sending data to the master.
- **MOSI (Master Out Slave In)** - The Master line for sending data to the peripherals.
- **SCK (Serial Clock)** - The clock pulses which synchronize data transmission generated by the master.

**SS (Slave Select)** –Master can use this pin to enable and disable specific devices.

**To start communication between master and slave** we need to set the required device's Slave Select (SS) pin to LOW so that it can communicate with the Master. When it is high, it ignores the Master. This allows you to have multiple SPI devices sharing the same MISO, MOSI, and CLK lines of master. As you can see in Fig. 1, there are 3 slaves in which the SCLK, MISO, MOSI are common connected to the Master and the SS of each slave is connected separately to individual SS pins (SS1, SS2, SS3) of the Master. By setting the required SS pin to LOW, a Master can communicate with that slave.

**SPI Pins in Arduino UNO**

Fig. 2 shows the SPI pins present Arduino UNO (in the red box).

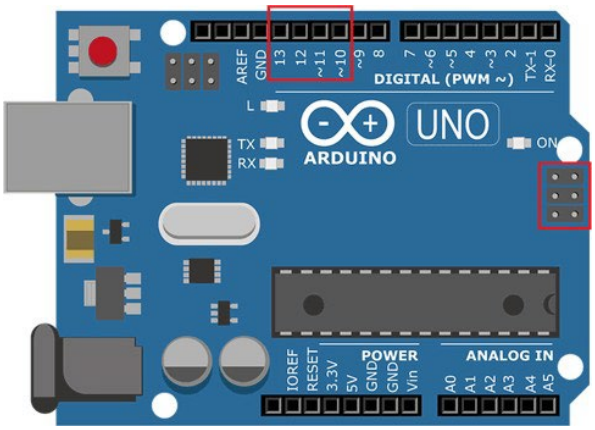



Fig. 2 Arduino board's pins for SPI communication Table 1: Pin numbers for SPI lines

| SPI Line | Pin in Arduino |
|---|---|
| MOSI | 11 or ICSP-4 |
| MISO | 12 or ICSP-1 |
| SCK | 13 or ICSP-3 |
| SS | 10 |

**Using SPI Protocol in Arduino**

Before the start of the programming for **SPI communication between two Arduinos**, we need to learn about the

**Arduino SPI libraries** used in Arduino IDE.

The header file **<SPI.h>** is included in the main program for using the following functions for the SPI communication.

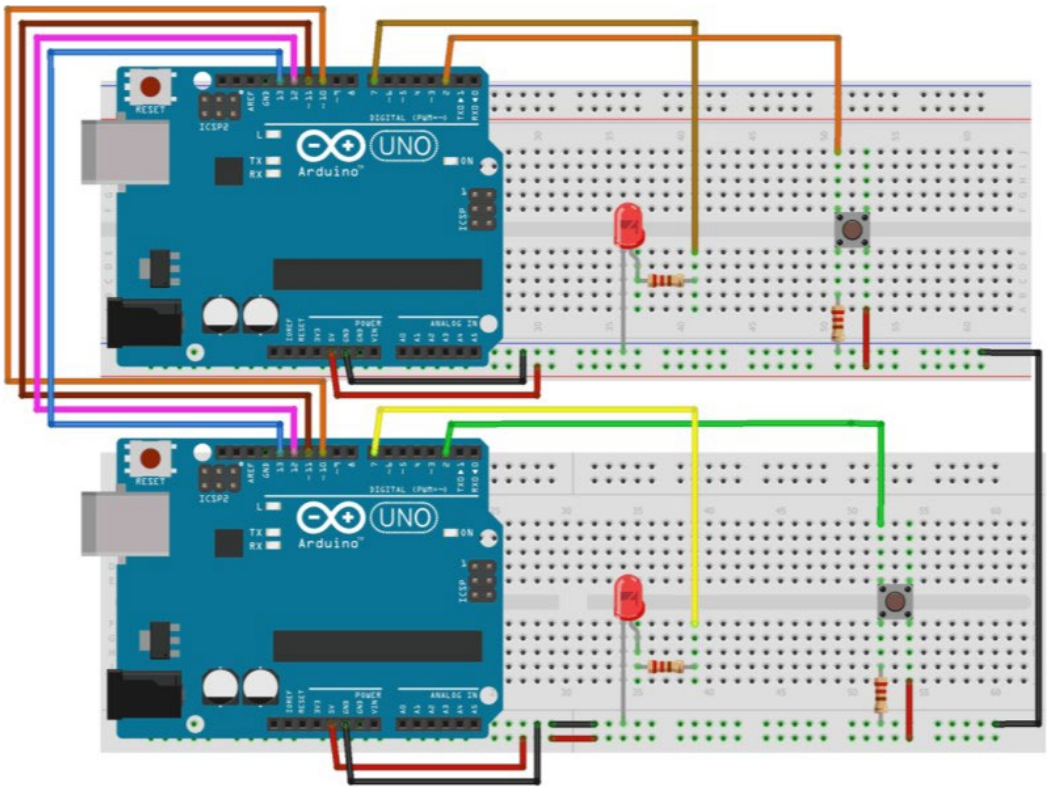
1. **SPI.begin():** To Initialize the SPI bus by setting SCK, MOSI, and SS to outputs, pulling SCK and MOSI low, and SS high.
2. **SPI.setClockDivider(divider):** To Set the SPI clock divider relative to the system clock. The available dividers are 2, 4, 8, 16, 32, 64, or 128.
   Dividers are SPI_CLOCK_DIVn, where n = 2, 4, 8, 16, 32, 64, or 128.
3. **SPI.attachInterrupt(handler):** This function is called when a slave device receives data from the Master.
4. **SPI.transfer(val):** This function is used to simultaneous send and receive the data between Master and slave.

So, now let us start with practical demonstration of SPI protocol in Arduino. In this experiment, we will use two Arduinos- one as the Master and other as a slave. Both Arduinos are attached with a LED and a push button switch separately. Master LED can be controlled by using slave Arduino's push button and vice-versa through SPI communication protocols.

## Experimental Procedure

1. Two Arduino boards were connected to the computer via USB cables.
2. The SPI Master and Slave codes were uploaded to their respective boards using the Arduino IDE.
3. A push button and an LED were connected to each board's digital pins 8 and 9, respectively.
4. The SPI pins between the two Arduinos were connected as follows:
   - The Master's MOSI pin was connected to the Slave's SDI pin.
   - The Master's MISO pin was connected to the Slave's SDO pin.
   - The Master's SCK pin was connected to the Slave's SCK pin.
   - The Master's SS pin was connected to the Slave's CS pin.
5. The Serial Monitor was opened on the Master board to view the communication between the boards.
6. "1" was typed into the Serial Monitor to turn on the Slave's LED with the Master's push button. "0" was typed to turn it off.
7. "2" was typed into the Serial Monitor to turn on the Master's LED with the Slave's push button. "0" was typed to turn it off.
8. The LEDs and push buttons were observed as commands were sent between the boards.

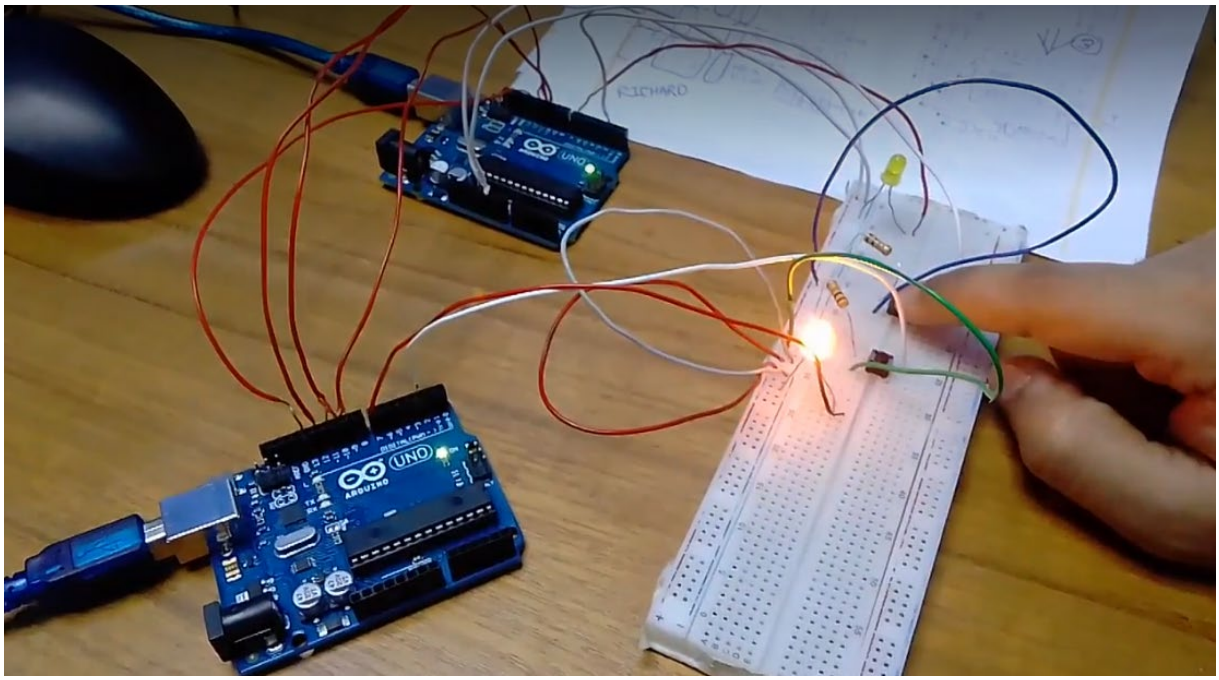
## Circuit Diagram

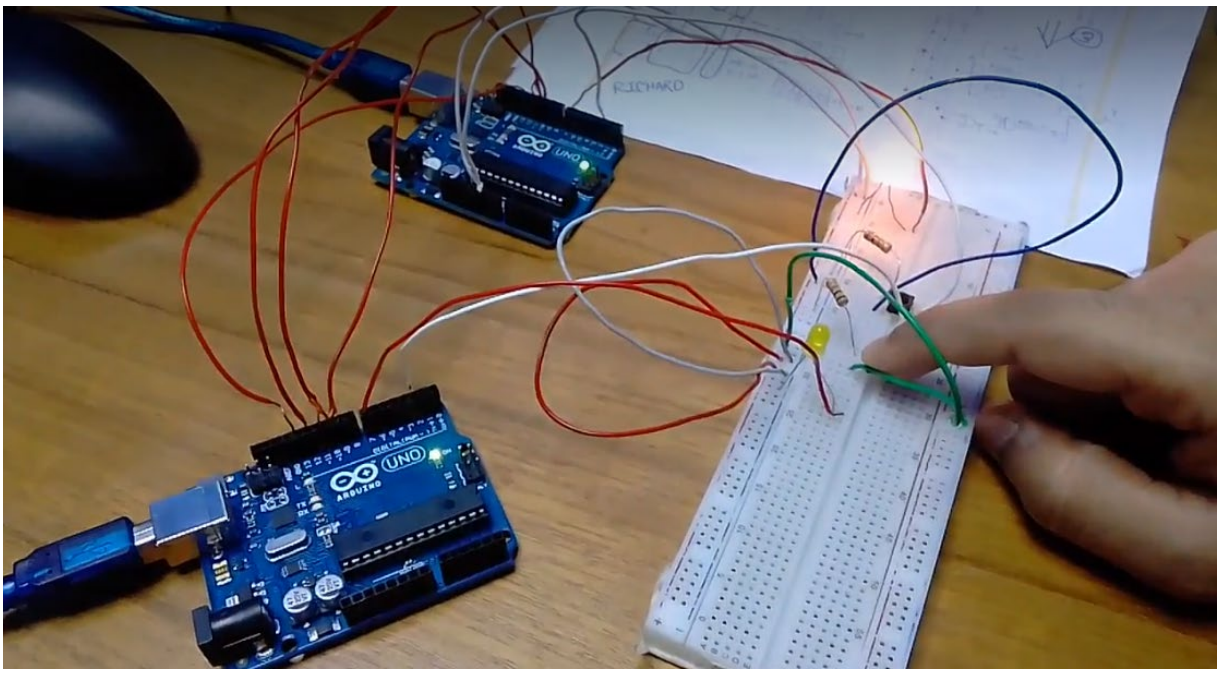


## Apparatus

- Arduino UNO (2)
- LED (2)
- Push Button (2)
- Resistors 10 k, 2.2 k (2 + 2)
- Breadboard
- Connecting Wires

## Hardware Output Results



# Code/Program

### Master Arduino Code:

```
// SPI MASTER (ARDUINO)
// SPI COMMUNICATION BETWEEN TWO ARDUINO
#include <SPI.h> // Library for SPI
#define LED 7
#define ipbutton 2

int buttonvalue;
int x;

void setup() {
  Serial.begin(115200); // Starts Serial Communication at Baud Rate 115200

  pinMode(ipbutton, INPUT); // Sets pin 2 as input
  pinMode(LED, OUTPUT); // Sets pin 7 as Output

  SPI.begin(); // Begins the SPI communication
  SPI.setClockDivider(SPI_CLOCK_DIV8); // Sets clock for SPI communication at
8 (16/8=2Mhz)
  digitalWrite(SS, HIGH); // Setting SlaveSelect as HIGH (So master doesn't
connect with slave)
}

void loop() {
  byte Mastersend, Mastereceive;

  buttonvalue = digitalRead(ipbutton); // Reads the status of the pin 2

  if (buttonvalue == HIGH) { // Logic for Setting x value (To be sent to
slave) depending upon input from pin 2
    x = 1;
  } else {
    x = 0;
  }

  digitalWrite(SS, LOW); // Starts communication with Slave connected to
master

  Mastersend = x;
  Mastereceive = SPI.transfer(Mastersend); // Send the Mastersend value to
slave also receives value from slave

  if (Mastereceive == 1) { // Logic for setting the LED output depending upon
value received from slave
    digitalWrite(LED, HIGH); // Sets pin 7 HIGH
    Serial.println("Master LED ON");
  } else {
    digitalWrite(LED, LOW); // Sets pin 7 LOW
    Serial.println("Master LED OFF");
  }

  delay(1000);
}
```

**Slave Arduino Code:**

```
// SPI SLAVE (ARDUINO)
// SPI COMMUNICATION BETWEEN TWO ARDUINO
#include <SPI.h>
#define LEDpin 7
#define buttonpin 2

volatile boolean received;
volatile byte Slavereceived, Slavesend;

int buttonvalue;
int x;

void setup() {
  Serial.begin(115200);
  pinMode(buttonpin, INPUT);
  pinMode(LEDpin, OUTPUT);
  pinMode(MISO, OUTPUT);
  SPCR |= _BV(SPE);  // Turn on SPI in Slave Mode
  received = false;
  SPI.attachInterrupt();  // Interrupt ON is set for SPI communication
}

ISR (SPI_STC_vect) {  // Interrupt routine function
  Slavereceived = SPDR;  // Value received from master is stored in variable
slavereceived
  received = true;  // Sets received as True
}

void loop() {
  if (received) {  // Logic to set LED on or off depending upon the value
received from master
    if (Slavereceived == 1) {
      digitalWrite(LEDpin, HIGH);  // Sets pin 7 as HIGH (LED on)
      Serial.println("Slave LED ON");
    } else {
      digitalWrite(LEDpin, LOW);  // Sets pin 7 as LOW (LED off)
      Serial.println("Slave LED OFF");
    }

    buttonvalue = digitalRead(buttonpin);  // Reads the status of pin 2

    if (buttonvalue == HIGH) {  // Logic to set the value of x to send to
master
      x = 1;
    } else {
      x = 0;
    }

    Slavesend = x;
    SPDR = Slavesend;  // Sends the x value to master via SPDR
    delay(1000);
  }
}
```

## Discussion & Conclusion

In this experiment, we successfully established SPI communication between two Arduino boards using a Master-Slave configuration. The codes were uploaded to their respective boards, and the pins were connected accordingly. The push buttons and LEDs were used as input and output devices to send and receive data between the boards. We observed that the Master board sent a value to the Slave board, which then triggered the Slave board to turn on or off its LED. Similarly, the Slave board sent a value to the Master board, which turned on or off the Master board's LED. We also noted that the data transfer was reliable, and the communication was bidirectional. Overall, this experiment demonstrated the ease and effectiveness of SPI communication in Arduino boards and its practical applications in various electronic projects.

## References


- AIUB Lab Manual
- https://www.arduino.cc