American International University- Bangladesh
Department of Electrical and Electronic Engineering
EEE 4103: Microprocessor and Embedded Systems Laboratory

Title: Communication between two Arduino Boards using SPI.

Introduction:

The objectives of this experiment are to-
1. Study the SPI protocol used in Arduino.
2. Write assembly language programming code for SPI communication with Arduinos.
3. Use SPI protocol for communication between two Arduinos.
4. Build a circuit to control the master side LED by the push button at the slave side and vice versa using the SPI Serial communication protocol.
5. Know the working principles of the SPI used in Arduino.

Theory and Methodology:

A Microcontroller uses many different protocols to communicate with various sensors and modules. There are many different types of communication protocols for wireless and wired communication, and the most commonly used communication technique is Serial Communication. Serial communication is the process of sending data one bit at a time, sequentially, over a communication channel or bus. There are many types of serial communication, like UART, CAN, USB, I2C, and SPI communication.

SPI (Serial Peripheral Interface) is a serial communication protocol. The SPI interface was founded by Motorola in 1970. SPI has a full-duplex connection, which means that the data is sent and received simultaneously. That is a master can send data to a slave and a slave can send data to the master simultaneously. SPI is synchronous serial communication means the clock is required for communication purposes.

Important Note: A new resolution is underway to improve the terminologies used in SPI communication by removing words like "Master" and "Slave" while discussing SPI. According to this new resolution, people are encouraged to use the word "Controller" in place of "Master" and "Peripheral" in place of "Slave". It is expected that the terms MOSI/MISO and SS will be changed to SDI (Serial Data In)/SDO (Serial Data Out) and CS (Chip Select) respectively. For the sake of avoiding confusion, we have still used the old terminologies in the article, but we encourage our readers to practice the new terms.

An SPI has a master/Slave communication by using four lines. An 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.

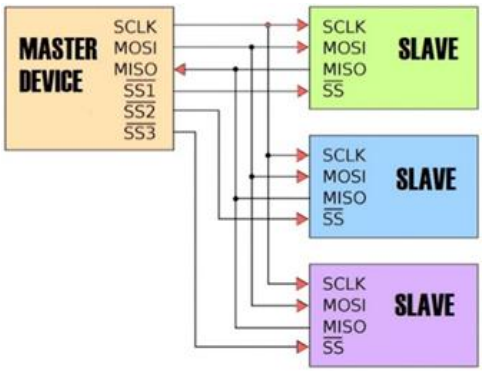

Fig. 1 block diagram representation of SPI Master with several slaves

SPI has the 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)** –The 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 the master. As you can see in Fig. 1, there are 3 slaves in which the SCLK, MISO, and MOSI are commonly connected to the Master and the SS of each slave is connected separately to individual SS pins (SS1, SS2, and 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 in Arduino UNO (in the red box).

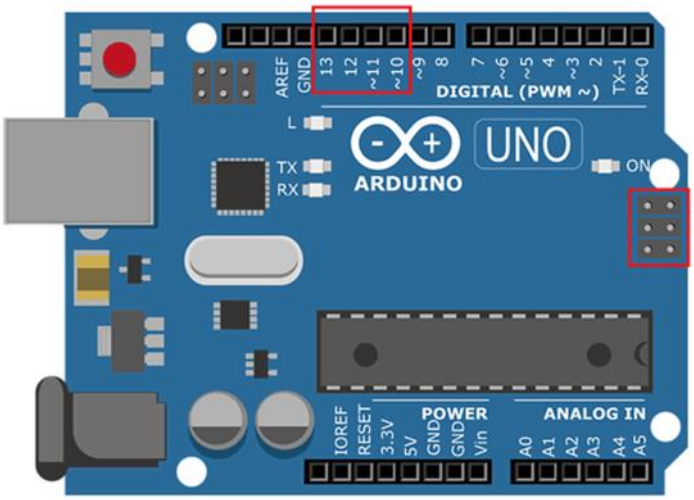

Fig. 2 Arduino board's pins for SPI communication

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 simultaneously send and receive the data between Master and slave.

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

Apparatus:

1. Arduino IDE (2.0.1 or any recent version)
2. Arduino UNO (2)
3. LED (2)
4. Push Button (2)
5. Resistors 10 k, 2.2 k (2 + 2)
6. Breadboard
7. Connecting Wires

Arduino SPI Communication Experimental Circuit Diagram:

The below circuit diagram shows how to use SPI on Arduino UNO, but you can follow the same procedure for the Arduino Mega or Arduino Nano SPI communications. Almost everything will remain the same except for the pin numbers. You must check the pinouts of Arduino Nano or Mega.

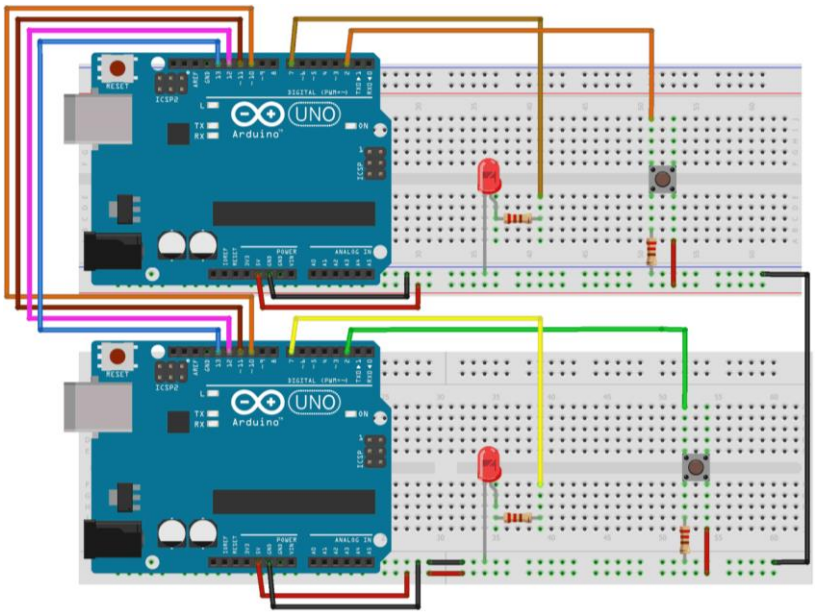

Fig. 3 Two Arduino board's pin connections for SPI communications (schematic diagram)

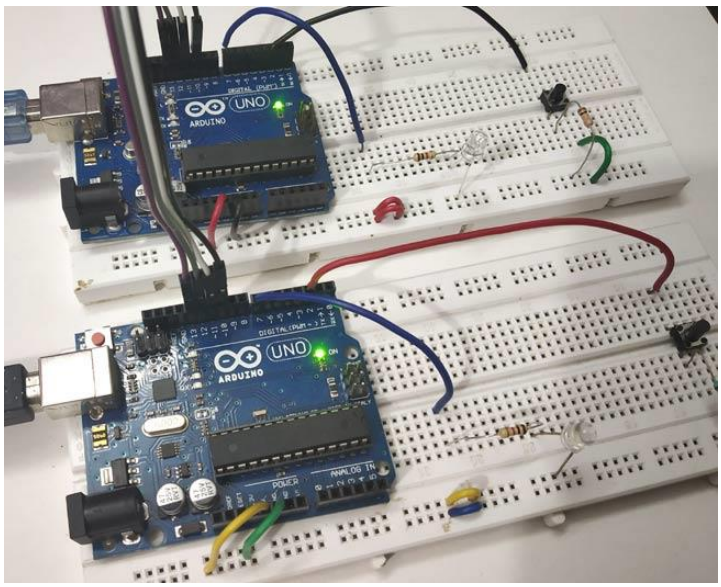

Fig. 4 Two Arduino board's pin connections for SPI communications (breadboard diagram)

How to Program Arduino for SPI Communication:

This experiment has two programs- one for Master Arduino and the other for the slave Arduino.

Arduino SPI Master Programming Explanation:

1. First of all we need to include the SPI library for using SPI communication functions.

```
#include<SPI.h>
```

2. In void setup()

We Start Serial Communication at a Baud Rate of 1,15,200.

```
Serial.begin(115200);
```

Attach LED to pin 7 and Push button to pin 2 and set those pins as OUTPUT and INPUT respectively.

```
pinMode(ipbutton,INPUT);
pinMode(LED,OUTPUT);
```

Next, we begin the SPI communication.

```
SPI.begin();
```

Next, we set the Clockdivider for SPI communication. Here, we have set divider 8.

```
SPI.setClockDivider(SPI_CLOCK_DIV8);
```

Then we set the SS pin HIGH since we did not start any transfer from the Master to the slave Arduino.

```
digitalWrite(SS,HIGH);
```

3. In void loop():

We read the status of the push button pin connected to pin2 (Master Arduino) for sending those values to the slave.

```
buttonvalue = digitalRead(ipbutton);
```

Set Logic for setting x value (to be sent to the slave) depending upon the input from pin 2

```
  if(buttonvalue == HIGH)
  {
    x = 1;
  }
  else
  {
    x = 0;
  }
```

Before sending the value, we need to send LOW to the slave and select a value to begin the transfer to the slave from the Master.

```
digitalWrite(SS, LOW);
```

Then we send the push button value stored in *Mastersend* variable to the slave Arduino and also receive value from the slave that will be stored in *Mastereceive* variable.

```
Mastereceive=SPI.transfer(Mastersend);
```

After that, depending upon the *Mastereceive* value, we will turn the Master Arduino LED ON or OFF.

```
  if(Mastereceive == 1)
  {
    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");
  }
```

Note: We use *serial.println()* to view the results in a new line in Serial Motor of Arduino IDE.

Arduino SPI Slave Programming Explanation:

1. First of all we need to include the SPI library for using SPI communication functions.

```
#include<SPI.h>
```

2. In void setup()

We Start Serial Communication at Baud Rate of 1,15,200.

```
Serial.begin(115200);
```

Attach LED to pin 7 and Push button to pin2 and set those pins OUTPUT and INPUT respectively.

```
pinMode(ipbutton,INPUT);
pinMode(LED,OUTPUT);
```

We set MISO as OUTPUT (to send data to Master IN). So, data is sent via MISO of Slave Arduino.

```
pinMode(MISO,OUTPUT);
```

Now, turn on or enable the SPI in Slave Mode by using the SPI Control Register (bit 6 of SPCR).

```
SPCR |= _BV(SPE);
```

Then turn ON interrupt for SPI communication. If data is received from the Master, the Interrupt Service Routine (ISR) is called and the received value is taken from SPDR (SPI Data Register)

```
SPI.attachInterrupt();
```

The value from the master is taken from SPDR and stored in *Slavereceived* variable. This takes place by following the Interrupt Routine function.

```
ISR (SPI_STC_vect)
{
  Slavereceived = SPDR;
  received = true;
}
```

3. Next in void loop(), we set the Slave Arduino LED to turn ON or OFF depending upon the Slavereceived value.

```
  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");
    }
```

Next, we read the status of the Slave Arduino Push button and store the value in *Slavesend* to send the value to Master Arduino by giving value to SPDR register.

```
buttonvalue = digitalRead(buttonpin);
if (buttonvalue == HIGH)
     {
       x=1;
     }
else
     {
      x=0;
     }
Slavesend = x;
SPDR = Slavesend;
```

To check how SPI works on Arduino, let us test the program in hardware.
When the push button on the Master side is pressed, the LED on the slave side turns ON and when the push button on the Slave side is pressed, the LED on the Master side turns ON.

Experimental Procedure:

The main task of this experiment is to implement an LED control system using a push switch from the Master or Slave device to vice versa. Connect the circuits as per the diagram of Figs. 3. Then plug the Arduino microcontroller board into the PC.

Using Arduino IDE to write code:

1. Open the Arduino Uno IDE 2.0.1 (version may have been updated automatically on your computer) and a blank sketch will open. The window as in Fig. 2 will come up on your PC:

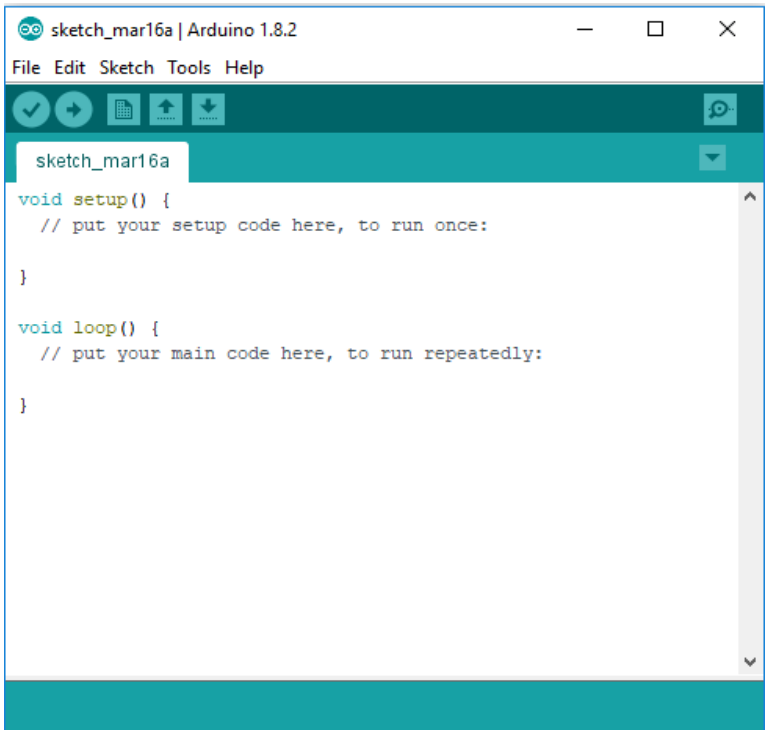

Fig. 5 IDE Environment (text editor)

2. Now, write the program by copying the codes given in the boxes in sequences on the blank sketch of Fig. 5 for the LED control.
    a. Save the files using the codes given above.
    b. Place the above two files in the same folder.
    c. Compile and upload to the hardware.
    d. Press the buttons and observe the control of LED outputs.

Code of an LED light control using a switch from the Master and Slave:

PART 1: Master Code:

Master/Controller Arduino Code:

```
//SPI MASTER (ARDUINO)
//SPI COMMUNICATION BETWEEN TWO ARDUINO CIRCUIT DIGEST

#include<SPI.h>            //Library for SPI

#define LED 7
#define ipbutton 2

int buttonvalue;

int x;

void setup (void){

  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 = 2 MHz)
  digitalWrite(SS,HIGH); //Setting SS to HIGH do disconnect master
from slave
}

void loop(void){

  byte Mastersend, Mastereceive;
  buttonvalue = digitalRead(ipbutton);   //Reads the status of the
pin 2

  if(buttonvalue == HIGH) //Setting x for the slave based on input
at pin 2
  {
    x = 1;
  }
  else
  {
    x = 0;
  }

  digitalWrite(SS, LOW); //Starts communication with Slave from the
Master
  Mastersend = x;
  Mastereceive = SPI.transfer(Mastersend); //Sends the Mastersend
value to
                      //the slave and also receives value from the
                      slave
```

```
  if(Mastereceive == 1) //To set the LED based on the value received
from slave
  {
    digitalWrite(LED,HIGH); //Sets pin 7 HIGH
    Serial.println("Master LED is ON");
  }

  else
  {
   digitalWrite(LED,LOW);  //Sets pin 7 LOW
   Serial.println("Master LED is OFF");
  }
  delay(1000);
}
```

PART 2: Slave Code:

Slave/Peripheral 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);    // Setting pin 2 as INPUT
  pinMode(LEDpin,OUTPUT);      // Setting pin 7 as OUTPUT
  pinMode(MISO,OUTPUT);      //Sets MISO as OUTPUT to send data to
Master In
  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 stored in
Slavereceived
  received = true;      //Sets received as True
}

void loop() {
   if(received) //To set LED ON/OFF based on the value received from
   Master
    {
```

```
      if (Slavereceived == 1)
      {
        digitalWrite(LEDpin, HIGH);   //Sets pin 7 as HIGH to turn
on LED

        Serial.println("Slave LED is ON");

      }

     else
      {
        digitalWrite(LEDpin,LOW);    //Sets pin 7 as LOW to turn off
LED
        Serial.println("Slave LED is OFF");
      }

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

      if (buttonvalue == HIGH)  //To set the value of x to send to
Master
      {
        x = 1;
      }

     else
      {
        x=0;
      }

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

Questions for report writing:

1) Include all codes and scripts in the lab report following the lab report writing template.
2) Show the output/results in the form of images. Give their captions and descriptions.
3) Configure the port numbers for outputs and inputs according to your ID. Consider the last two digits from your ID (if your ID is XY-PQABC-Z then consider input port as B and output port as C of your ID). Include all the programs and results within your lab report.
4) Include the **Proteus simulation** of the LED blink program and LED control system using push buttons. Explain the simulation methodology. You may learn the simulation from the following video link: *https://www.youtube.com/watch?v=yHB5it0s2oU*

Reference(s):

[1] https://www.arduino.cc/.
[2] ATMega328 manual
[3] https://www.avrfreaks.net/forum/tut-c-newbies-guide-avr-timers
[4] http://maxembedded.com/2011/06/avr-timers-timer0/