AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Faculty of Engineering

Lab Report

Experiment # 6

Experiment Title: Communication between two Arduino Boards using SPI.

Date of Perform:	Date Month 2023	**Date of Submission:**	24 July 2023
Course Title:	MICROPROCESSOR AND EMBEDDED SYSTEM LAB		
Course Code:	COE3104	**Section:**	N/A
Semester:	Summer 2022-23	**Degree Program:**	BSc in CSE/EEE
Course Teacher:	**Prof. Dr. Engr. Muhibul Haque Bhuyan**		

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 #

Sl No	Name	ID	PROGRAM	SIGNATURE

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

Table of Contents

Experiment Title: Communication between two Arduino Boards using SPI.

Objectives:

The objectives of this experiment are to-

- Familiarize with the SPI protocol used in Arduino.
- Implement assembly language programming code for SPI communication with Arduinos.
- Implement SPI protocol for communication between two Arduinos.
- Implement a circuit for controlling the master side LED by the push button at the slave side and vice versa using the SPI serial communication protocol.
- Familiarize with the working principles of the SPI used in Arduino.

Equipment List:

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

Circuit Diagram:

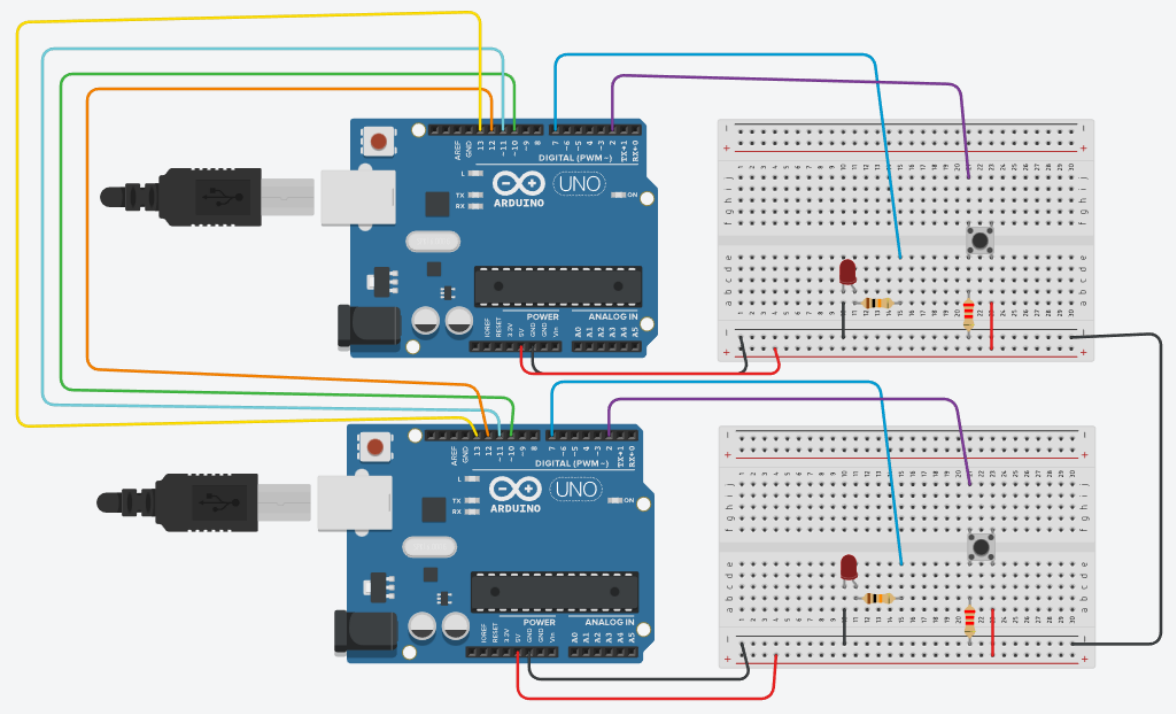

Fig. 1 Experimental setup of a LED light control using a switch from Master and slave

Code/Program:
The following is the code for the implementation of a LED light control using a switch from Master and slave with the necessary code explanation:

PART 1: Master 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:

```
//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 toMaster 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);
 }
}
```

Hardware Output Results:

Here is the hardware implementation of a LED light control using a switch from Master and slave and the necessary explanation of the implementation:

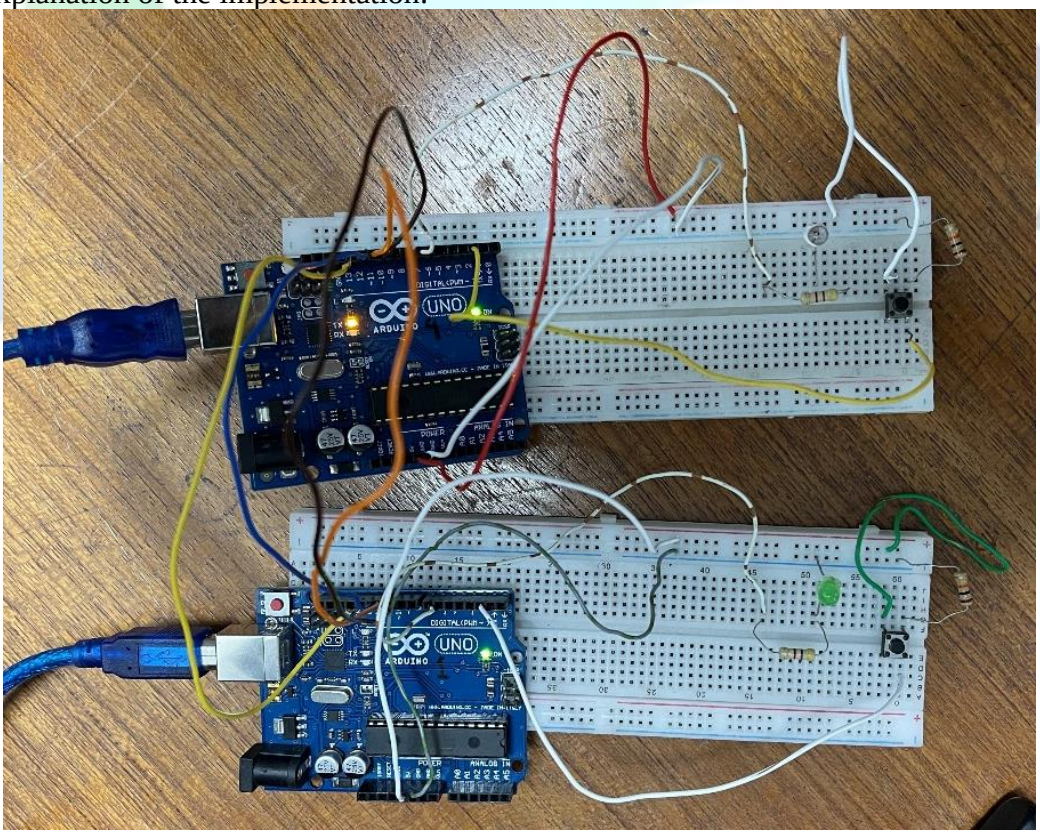

Fig. 2 Hardware implementation of a LED light control using a switch from Master and slave

Explanation:

In this experiment, two Arduino Uno Microcontrollers were taken. One of the microcontrollers were used as the "Controller" (*known as 'Master'*) and the other was used as "Peripheral" (*known as 'Slave'*). To transfer data from one microcontroller to another, the microcontrollers were connected using SPI communication medium. Pin 10 – 13 of both microcontrollers were connected accordingly for proper SPI connectivity. To provide data in the microcontrollers, a button was connected at pin 2 which was grounded using a 10k resistor. It was powered using the 5V power of the microcontroller. An LED was also set at both microcontrollers at pin 7 using a 2.2k resistor at the ground. Both of the breadboard that were used to connect the LED and button were commonly grounded using a wire.

Experimental Output Results:

Here are the results when pressing data button of controller and the necessary explanation of the results:

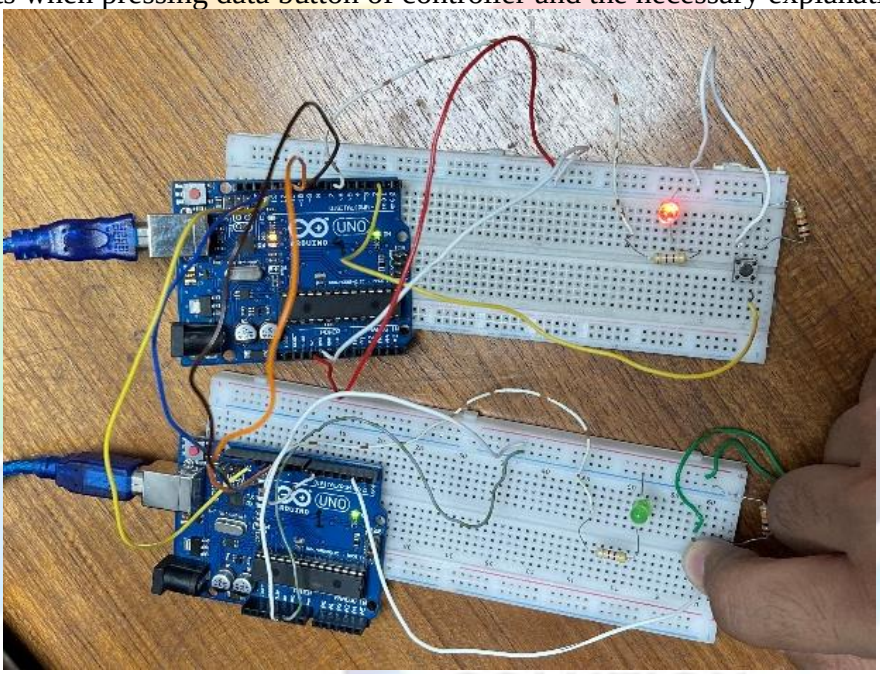

Fig.3 Peripheral LED was ON in the SPI communication circuit when pressing the button of Controller

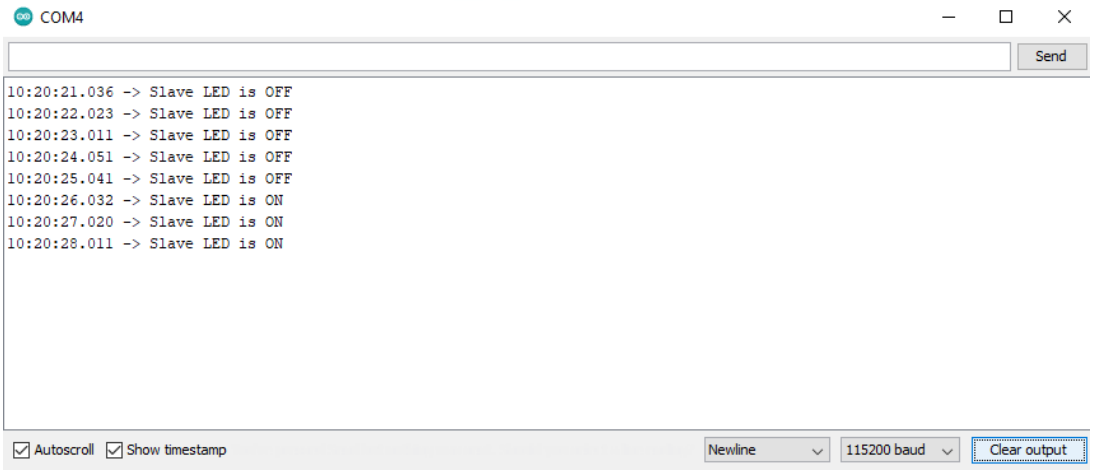

Fig.4 Serial Monitor of Peripheral when pressing Controller Data Button

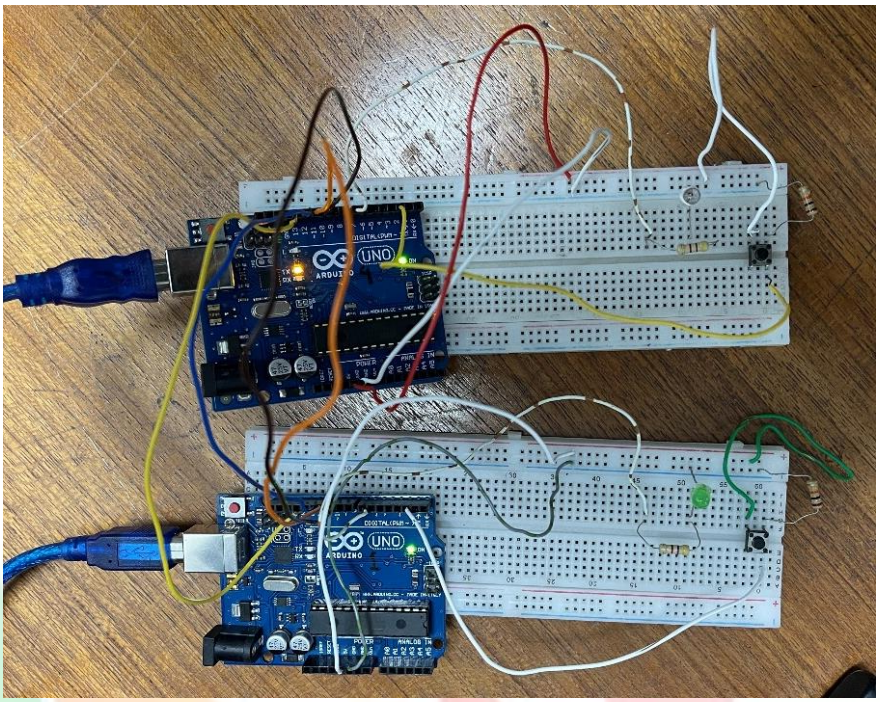

Fig.5 Peripheral LED was OFF in the SPI communication circuit when not pressing the button of Controller

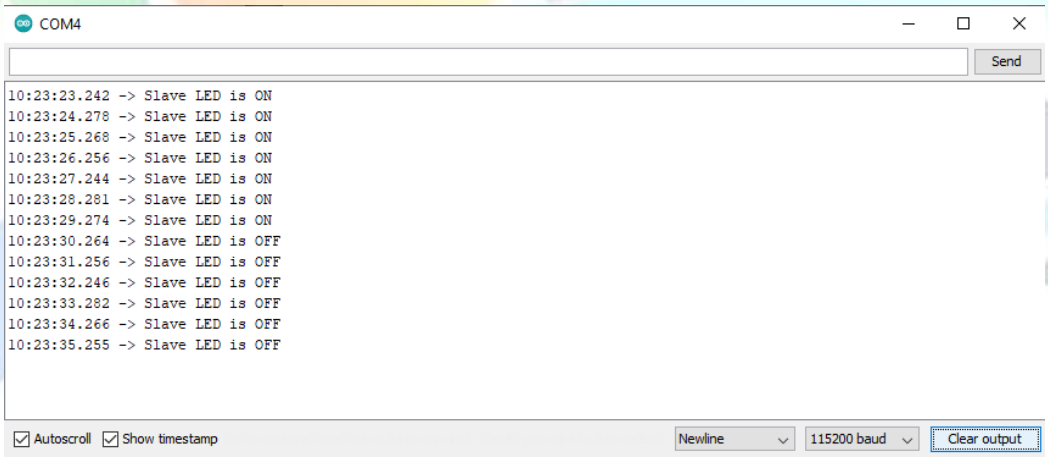

Fig.6 Serial Monitor of Peripheral when not pressing Controller Data Button

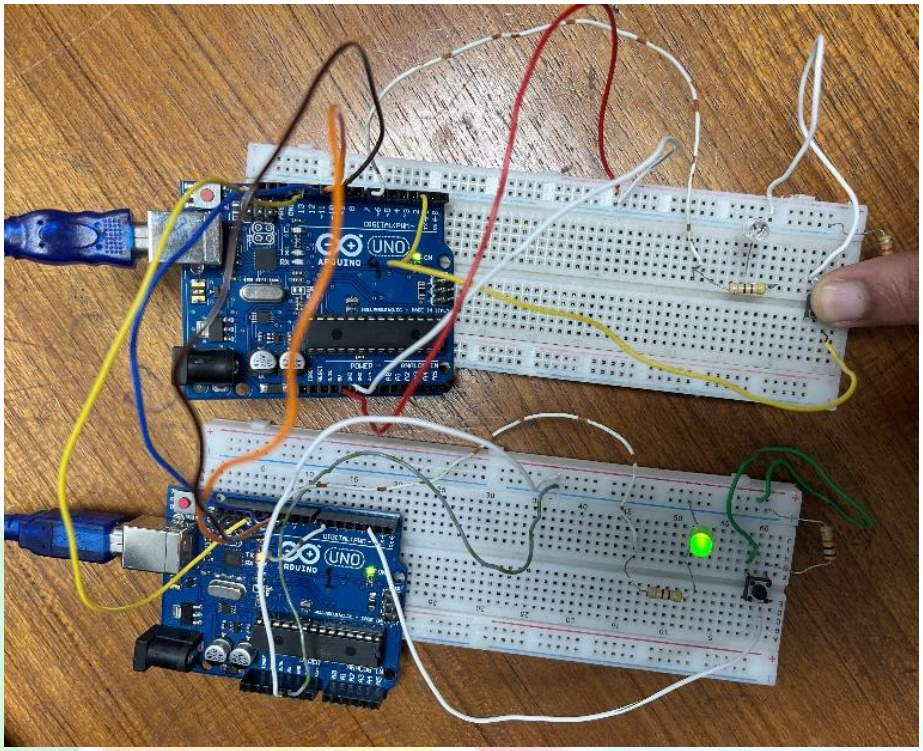

Fig.7 The Controller LED is ON in the SPI communication circuit when pressing the button of Peripheral

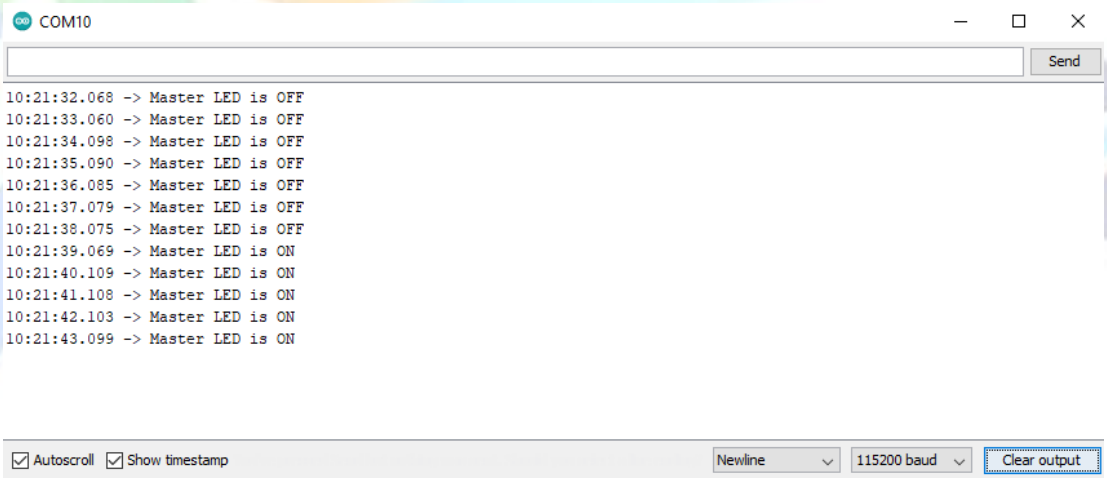

Fig.8 Serial Monitor of Peripheral when pressing Peripheral Data Button

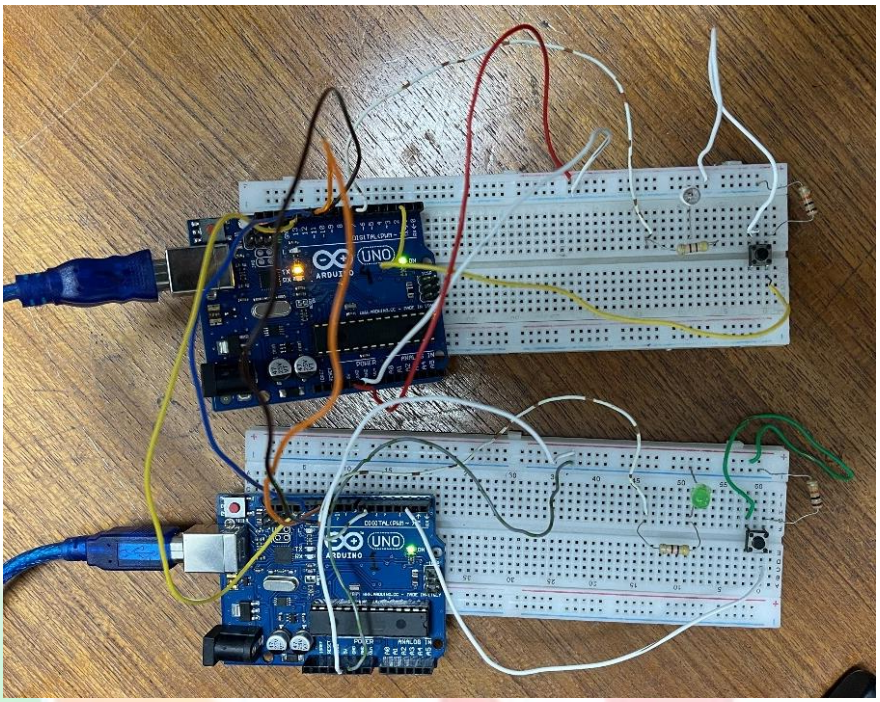

Fig.9 Controller LED was OFF in the SPI communication circuit when not pressing the button of Peripheral

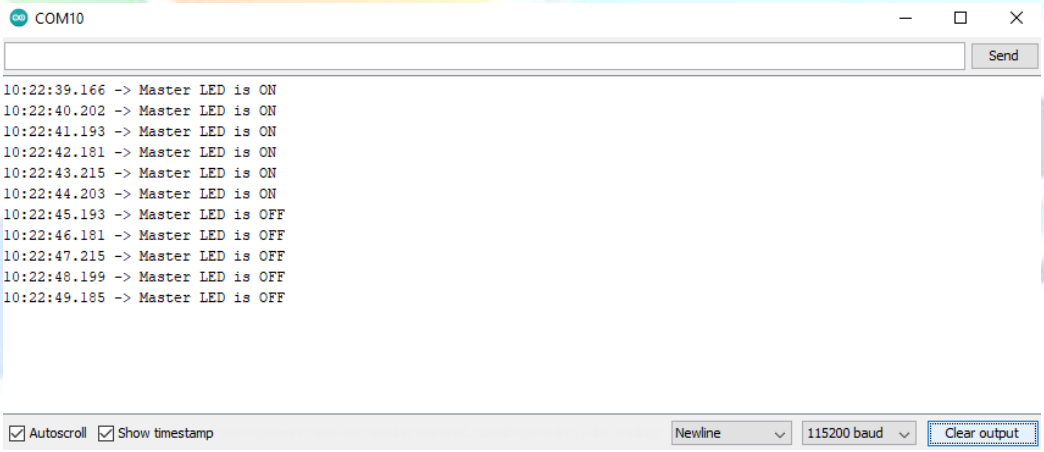

Fig.10 Serial Monitor of Peripheral when not pressing Peripheral Data Button

Explanation:

In this experiment, there were two separate experimental outcomes. At first was the result that was obtained when pressing the data transfer button at the Controller. When pressing the button at the controller, after few seconds the light on the peripheral lit up. This indicates that the data of lighting the LED of peripheral passed from controller to peripheral using the SPI communication interface. In the serial monitor of Controller, it is also found the monitor showed 'Slave LED is ON'. When the button on the Controller was not pressed any longer, the LED connected with the Peripheral side turned OFF. As a result, this meant that the data to turn off the LED was passed through the SPI communication interface without any problem and the serial monitor shower 'Slave LED is OFF'. Afterward the result that was obtained when pressing the data transfer button at the Peripheral. When pressing the button at the controller, after few seconds the light on the Controller lit up. This indicates that the data of turning the LED ON at the Controller passed from Peripheral to Controller using the SPI communication interface.

In the serial monitor of Peripheral, it is also found the monitor showed ‘Master LED is ON’. When the button on the Peripheral was not pressed any longer, the LED connected with the Controller side turned OFF. As a result, this meant that the data to turn off the LED was passed through the SPI communication interface without any problem and the serial monitor shower ‘Master LED is OFF’.

Simulation Output Results:

Here is the simulation implementation of a LED light control using a switch from Master and slave and the necessary explanation of the implementation:

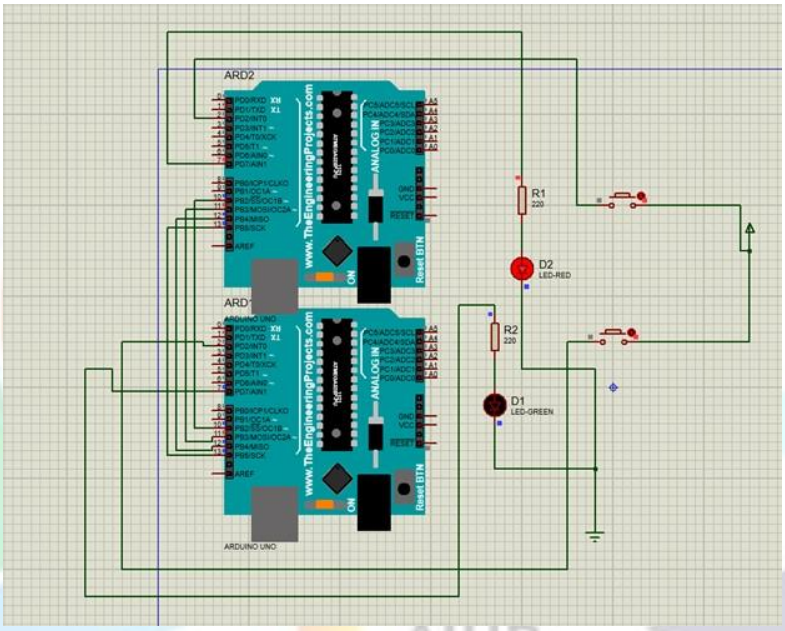

Fig.11 Peripheral LED was ON in the SPI communication circuit simulation when pressing the button of Controller

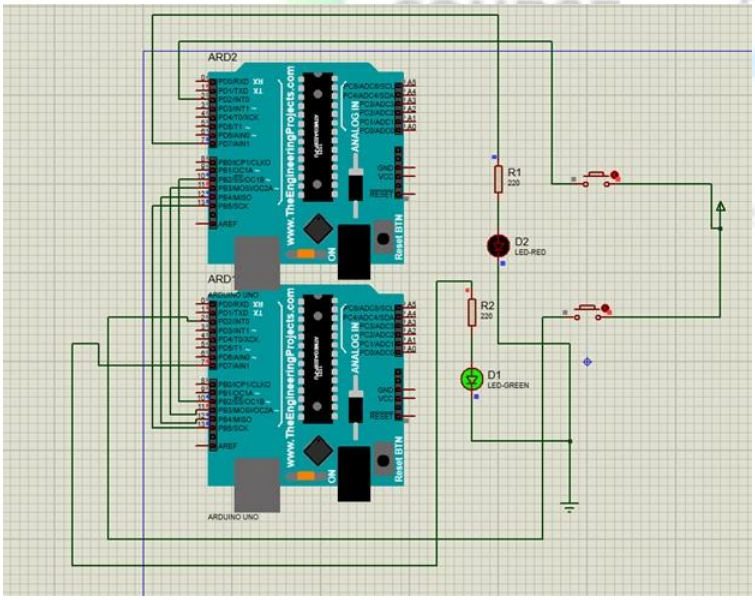

Fig.12 Controller LED was ON in the SPI communication circuit simulation when pressing the button of Peripheral

Explanation:
In this simulation, two Arduino Uno Boards, Resistors, GREEN and RED LEDs and buttons were configured according to the instructions provided in the commented code that was used to perform the task. Buttons to pass information from one microcontroller to another were also added in the simulation and were connected accordingly based on the code comments. Both of the boards were connected using the SPI communication interface using pin 10 – 13. The program was made in the Arduino IDE and was verified. As a result, a HEX file was generated. The following HEX file was implemented in the Proteus simulation for operation related instructions of the simulation. Afterwards, the simulation was performed and the results that were obtained were noted and compared with the hardware results.

Answers to the Questions in the Lab Manual:
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.
Solution: The following university ID was used:

2	1	-	4	4	6	4	9	-	1
X	Y	-	P	Q	A	B	C	-	Z

Port of Output pin would be accordingly:
LED 9 for master [Previously it was used as 7]
LEDpin 9 for slave

Port of Input pin would be accordingly:
ipbutton 4 for master [Previously it was used as 2]
buttonpin 4 for slave

The following is the code for a LED light control using a switch from Master and slave with the necessary code explanation:

The Modified Master Code:

```
#include<SPI.h>
#define LED 9
#define ipbutton 4

int buttonvalue;
int x;

void setup (void){
 Serial.begin(115200);
 pinMode(ipbutton,INPUT);
 pinMode(LED,OUTPUT);
 SPI.begin();
 SPI.setClockDivider(SPI_CLOCK_DIV8);
 digitalWrite(SS,HIGH);
}
```

```
void loop(void){
 byte Mastersend, Mastereceive;
 buttonvalue = digitalRead(ipbutton);
 if(buttonvalue == HIGH)
 {
  x = 1;
 }
 else
 {
  x = 0;
 }
 digitalWrite(SS, LOW);
 Mastersend = x;
 Mastereceive = SPI.transfer(Mastersend);
 if(Mastereceive == 1)
 {
  digitalWrite(LED,HIGH);
  Serial.println("Master LED is ON");
 }
 else
 {
  digitalWrite(LED,LOW);
  Serial.println("Master LED is OFF");
 }
 delay(1000);
}
```

The Modified Slave Code:

```
#include<SPI.h>
#define LEDpin 9
#define buttonpin 4

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);
 received = false;
 SPI.attachInterrupt();
```

```
}

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

void loop() {
if(received)
 {
  if (Slavereceived == 1)
  {
    digitalWrite(LEDpin, HIGH);
    Serial.println("Slave LED is ON");
  }
  else
  {
    digitalWrite(LEDpin,LOW);
    Serial.println("Slave LED is OFF");
  }
  buttonvalue = digitalRead(buttonpin);

  if (buttonvalue == HIGH)
  {
    x = 1;
  }
  else
  {
    x=0;
  }
  Slavesend = x;
  SPDR = Slavesend;
  delay(1000);
 }
}
```

Here is the simulation implementation of a LED light control using a switch from Master and slave and the necessary explanation of the implementation:

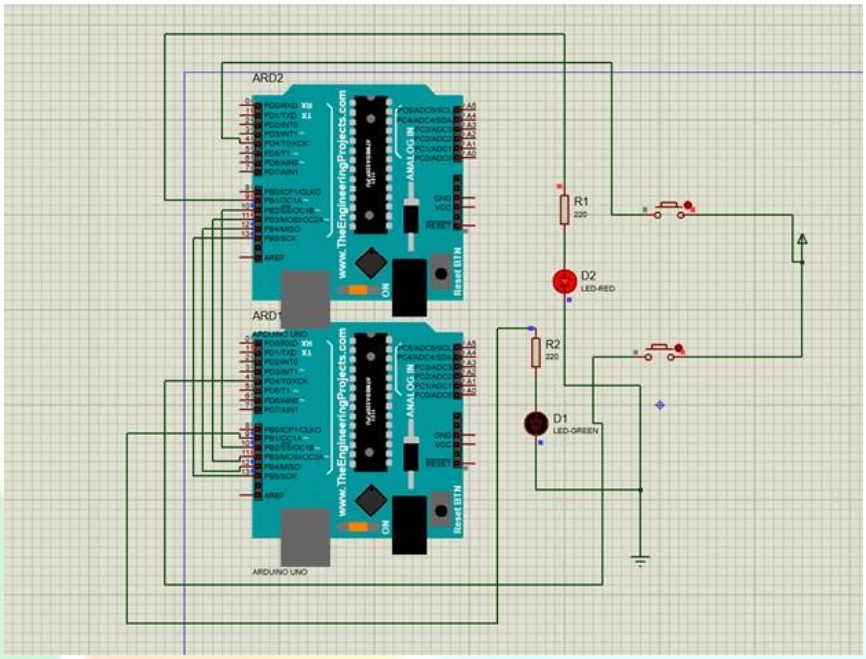

Fig.13 Peripheral LED was ON in the SPI communication circuit simulation when pressing the button of Controller on the modified code

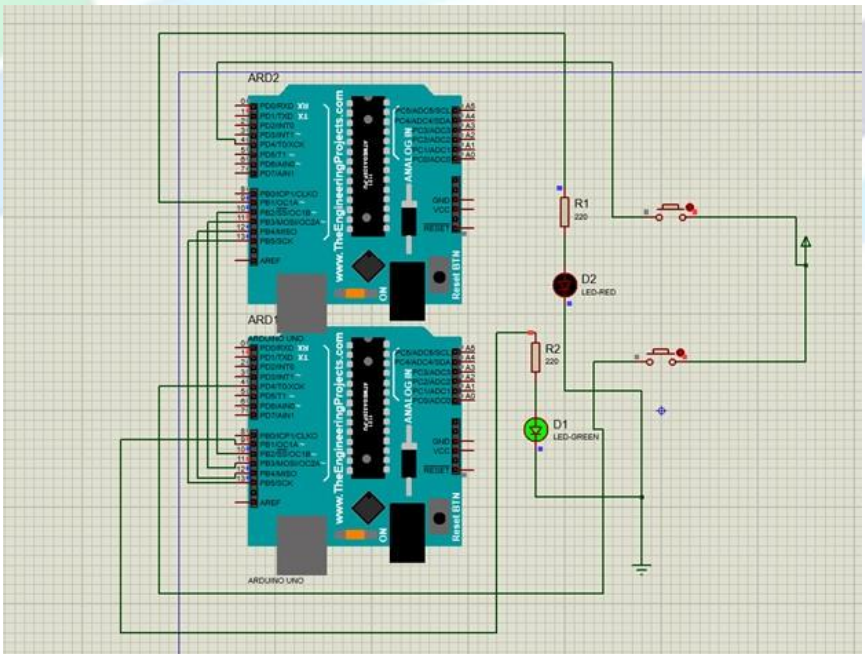

Fig.12 Controller LED was ON in the SPI communication circuit simulation when pressing the button of Peripheral on the modified code

Discussion:

The purpose of this experiment was to familiarize with the SPI communication interface of the microcontroller. At first, various kinds of communication interface were familiarized while learning about the microcontrollers. All the communication interface that was available on the Arduino microcontroller were understood from the datasheets that were collected. Serial Peripheral Interface is one of the communication interfaces that were available in the microcontroller. This communication interface was a full-duplex communication medium. Using this communication interface, various kinds of data can be transferred from one end to another without any problems. This communication interface required 4 connections. These were Slave Select (SS), Serial Clock (SCK), Master In Slave Out (MISO) and Master Out Slave In (MOSI). On any Arduino microcontroller, these can be found at Pin 10 – 13. All of this information were obtained and studied before proceeding to the practical implementation through a small experiment. In order to perform the small experiment, two microcontrollers were collected and they were connected through the SPI communication interface. During this experiment, one of the microcontrollers were marked as the Master and the other was marked as Slave. According to modern terminology, these are denoted as Controller and Peripheral accordingly for better understanding and convenience. A button was set on the Controller and a LED light was set on the Peripheral. Necessary code was developed and compiled to transfer the switch press information at the Controller. When the switch at the Controller was pressed, the SPI connection setup and the code that was verified sent data from the Controller to Peripheral. This caused the LED to turn on at the Peripheral microcontroller. On the serial monitor at the peripheral, it was also showing necessary data that the peripheral LED was turned on. This means that data successfully transferred from Controller to Peripheral. This part of the experiment helped us understood the working methodology behind MOSI – Master Out Slave In. Afterwards, a button was set on the Peripheral as well and a LED light was set on the Controller. Necessary code was developed and compiled to transfer the switch press information at the Peripheral. When the switch at the Peripheral was pressed, the SPI connection setup and the code that was verified sent data from the Peripheral to Controller. This caused the LED to turn on at the Controller microcontroller. On the serial monitor at the peripheral, it was also showing necessary data that the controller/master LED was turned on. This means that data successfully transferred from Peripheral to Controller. This part of the experiment helped us understood the working methodology behind MISO – Master In Slave Out. During this whole experiment, the Slave Select pin was set to LOW so that Slave(s)/Peripheral(s) could communicate with the Controller/Master. Serial Clock (SCK) generated clock pulses which synchronized data transmission generated by the master/controller with all the peripheral(s)/slave(s). All of the outcomes were observed and noted for future reference. All of these observations were then re-evaluated through simulations that were performed using computer software on a virtual environment. From the observation it can be said that after both hardware and software implementation showed the expected outcomes and the experimental objectives was achieved.

References:

1)Tutorialspoint, [Cited: July 22, 2023] Available: www.tutorialspoint.com/arduino/arduiono_serial_peripherals_interface.html

2) Arduino CC Website, [Online] [Cited: July 22, 2023] Available: https://docs.arduino.cc/hardware/uno-rev3

3) Arduino CC Website, [Online] [Cited: July 22, 2023] Available: https://docs.arduino.cc/learn/communication/spi

4) AIUB Microprocessor and Embedded Systems Lab Manual 6