AMERICAN INTERNATIONAL UNIVERSITY BANGLADESH

Faculty of Engineering

Lab Report

Experiment # 06

Experiment Title:

Course Title:	MICROPROCESSOR AND EMBEDDED SYSTEMS LAB		
Course Code:	COE3104	**Section:**	A
Semester:	Spring 2022-23	**Degree Program:**	BSc in CSE/BSc in EEE
Course Teacher:	**Md. Ali Noor**		

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 othersource 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 # 06

Sl No	Name	ID	Program
1	Maimona Rahman Farjana	20-42954-1	CSE
2	Md. Sumon	20-42556-1	CSE
3	Md. Sajidul Haque Shohan	20-42022-1	CSE
4	Md. Rashedul Islam	20-43301-1	CSE
5	Syeda Aynul Karim	19-41829-3	CSE
6	Medha Chowdhury	20-41930-1	CSE
7	Syed Aftab Uddin	19-41522-3	EEE

Faculty use only

FACULTY COMMENTS	**Marks Obtained**	
	Total Marks	

Table of Contents

Experiment Title: Communication between two Arduino Boards using SPI.

Introduction: 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. SPI interface was found 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.

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.

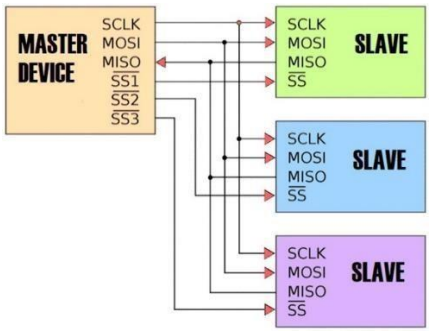


Fig. 1 block diagram representation of SPI Master with several slaves.

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).

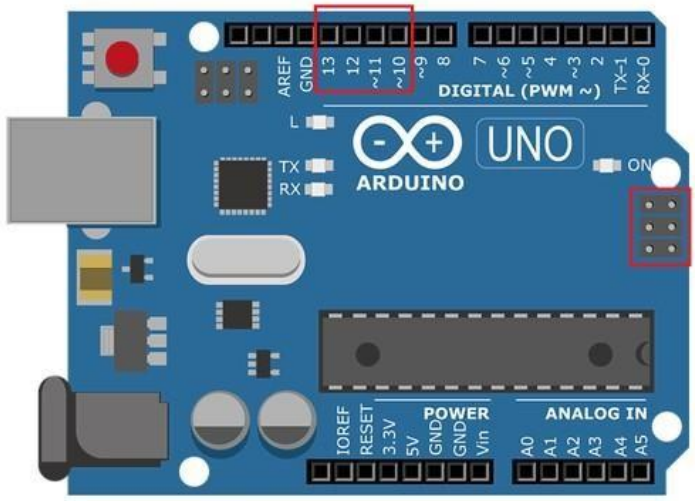


Fig. 2 Arduino board's pins for SPI communication

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.

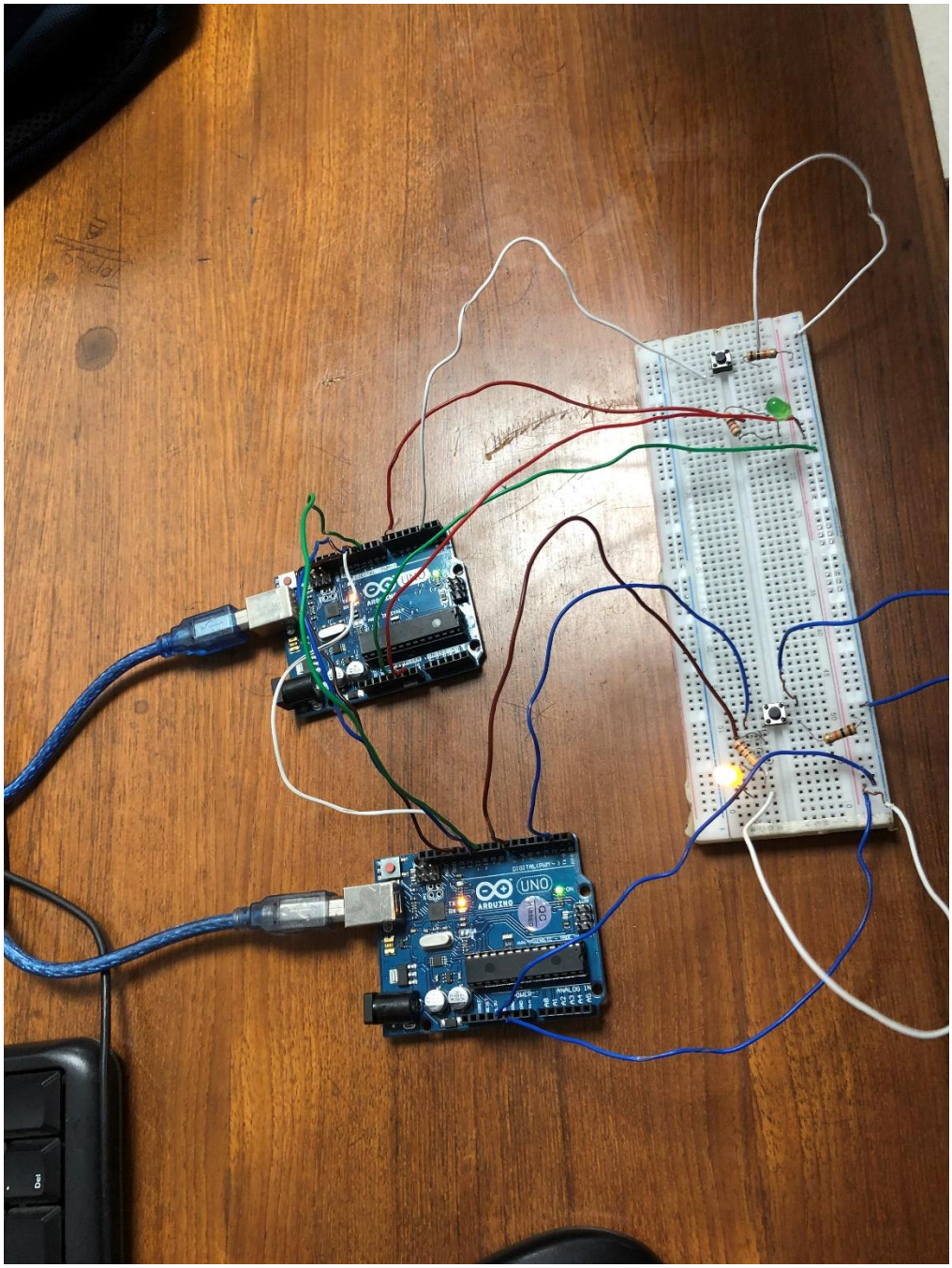
3. **SPI.attachInterrupt(handler):** This function is called when a slave device receives data from the Master.
4. **SPI.transfer(val):** This function is used to simultaneous send and receive the data between Master and slave.

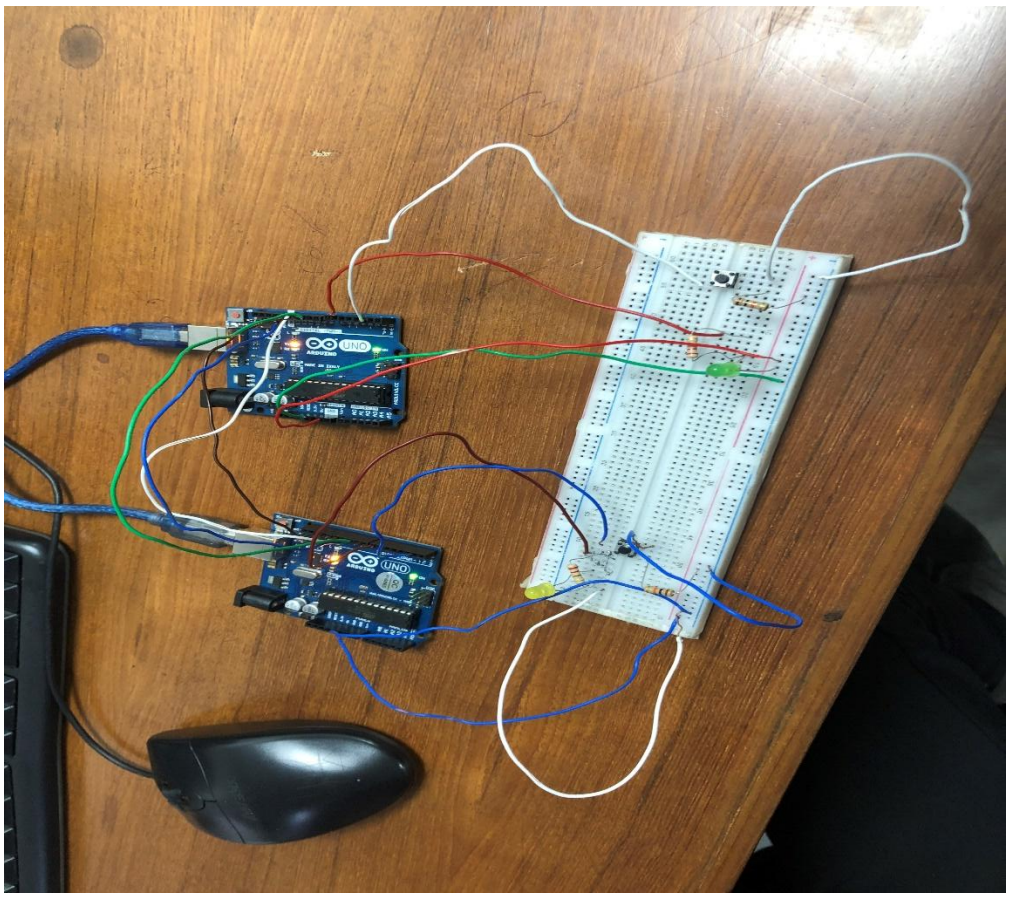
So, now let us start with practical demonstration of SPI protocol in Arduino. In this experiment, we will use two Arduinos- one as the Master and other as a slave. Both Arduinos are attached with a LED and a push button switch separately. Master LED can be controlled by using slave Arduino's push button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

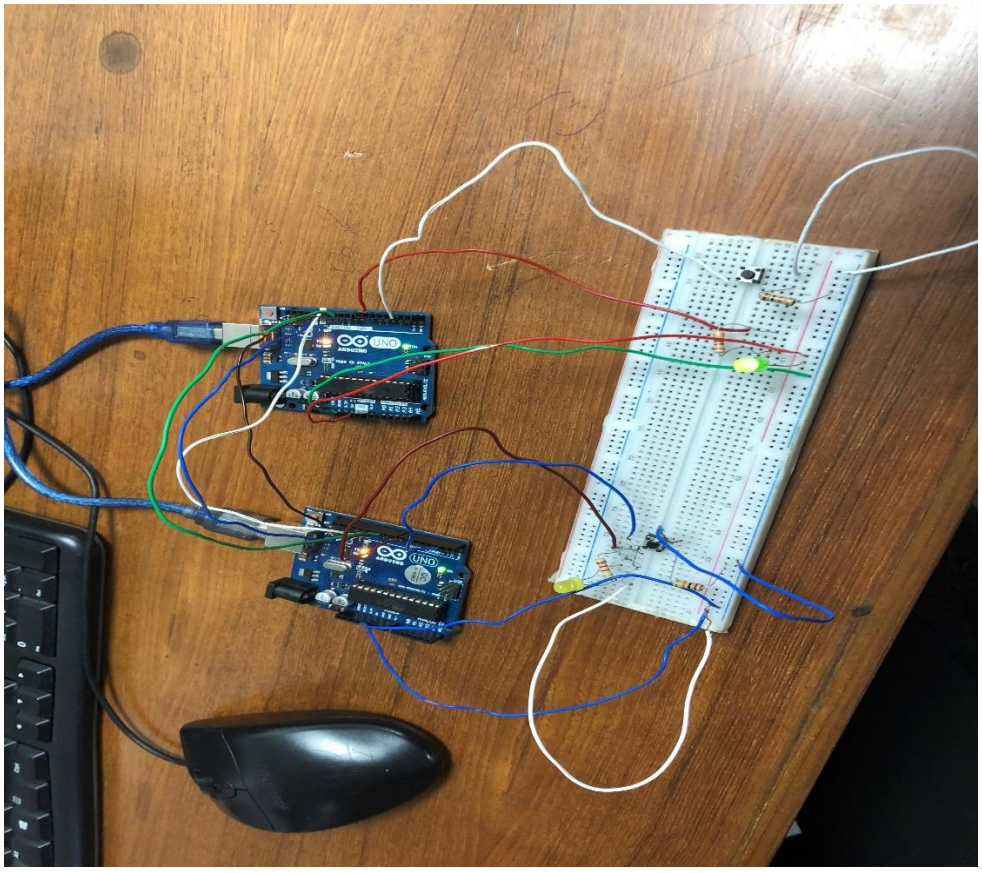
Equipment List:

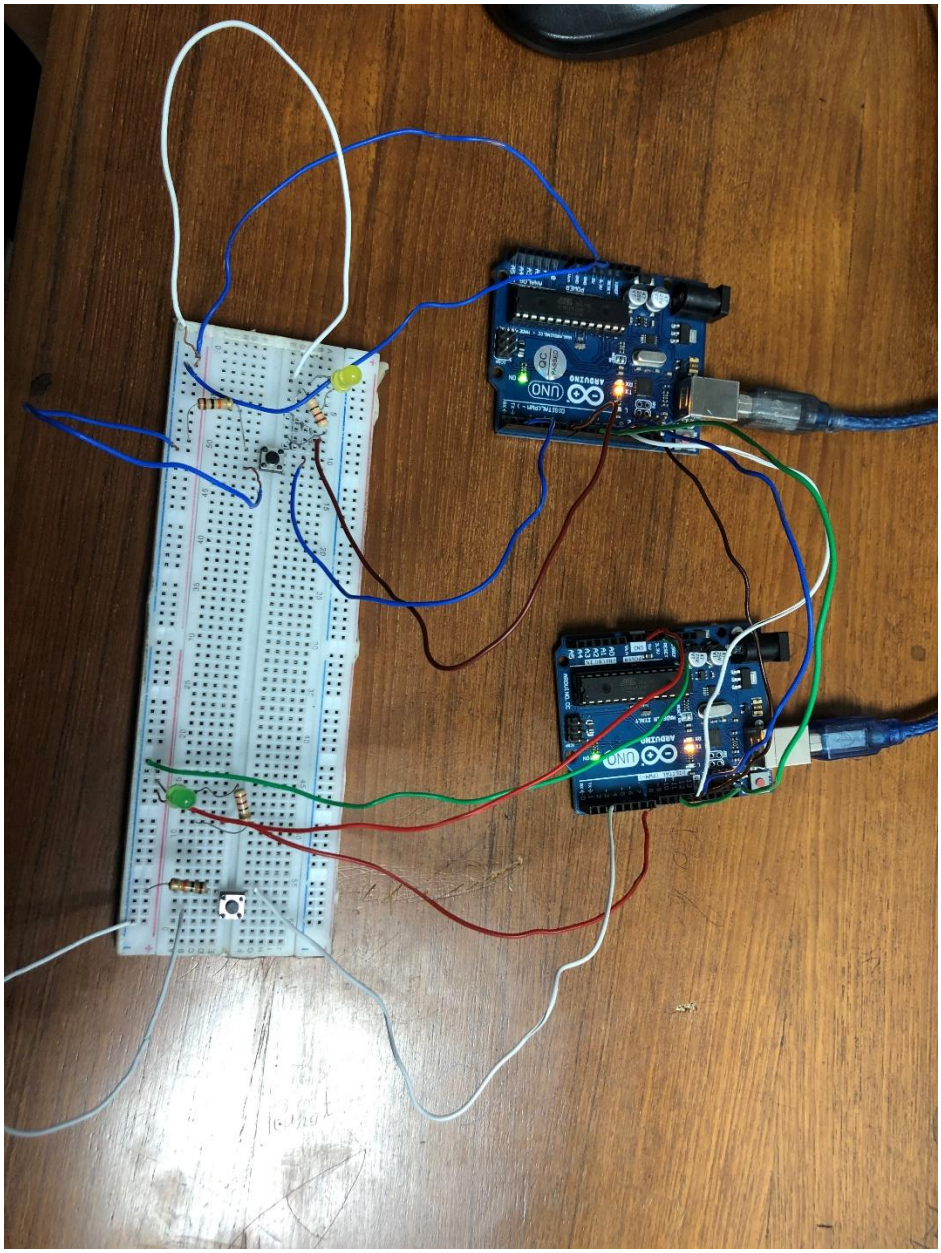
- Arduino UNO (2)
- LED (2)
- Push Button (2)
- Resistors 10k,2.2k (2+2)
- Breadboard
- Connecting Wires
-

Hardware Set-up:








UNO
ARDUINO

Simulation & results:

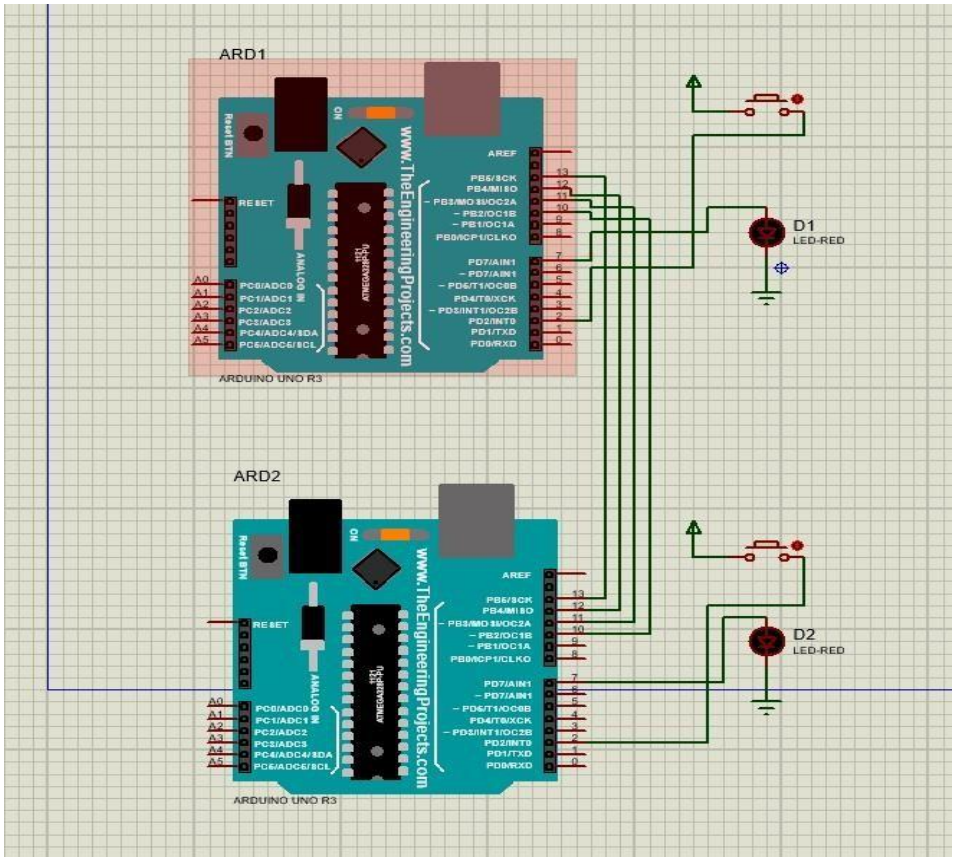


Fig: Both Switches are not pressed

In this picture, no push button is pressed. Therefore, no light lit up

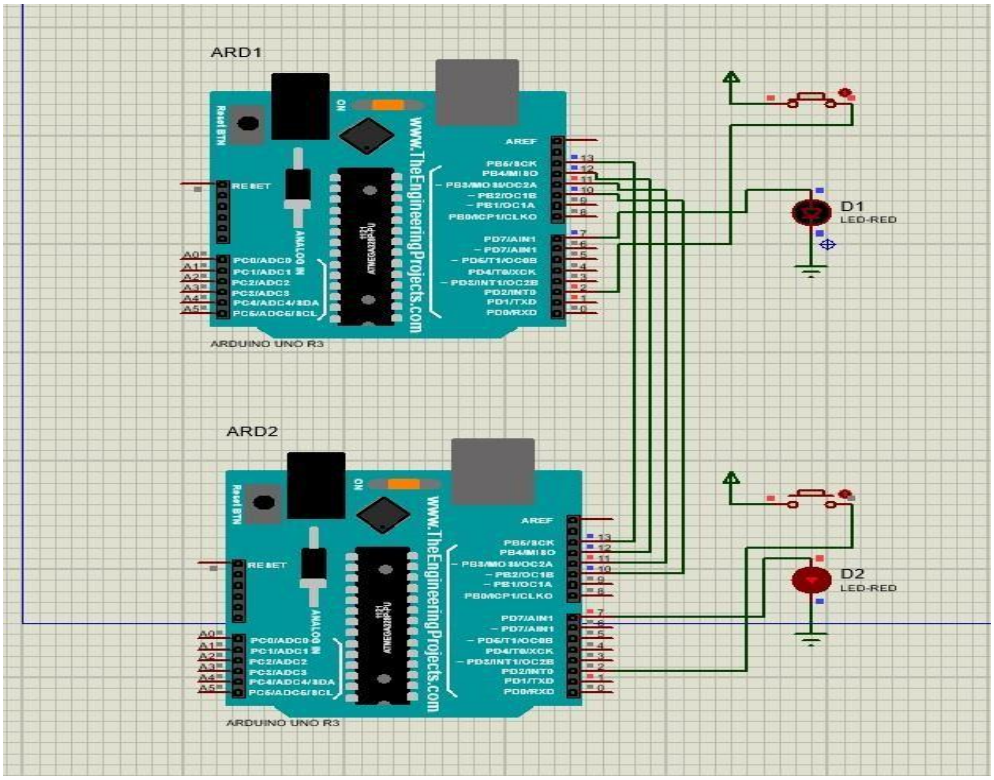


Fig: Push Button in Master Arduino is pressed and led of slave Arduino.

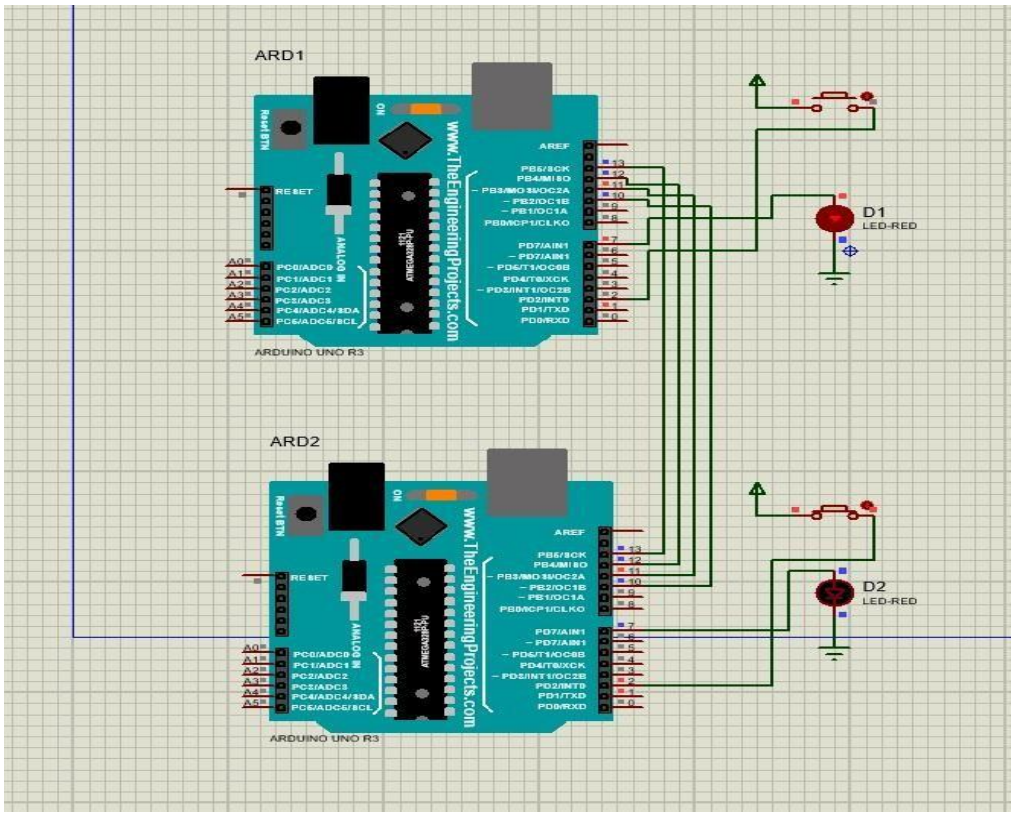


Fig: Push Button in Slave Arduino is pressed and led of Master Arduino lit up

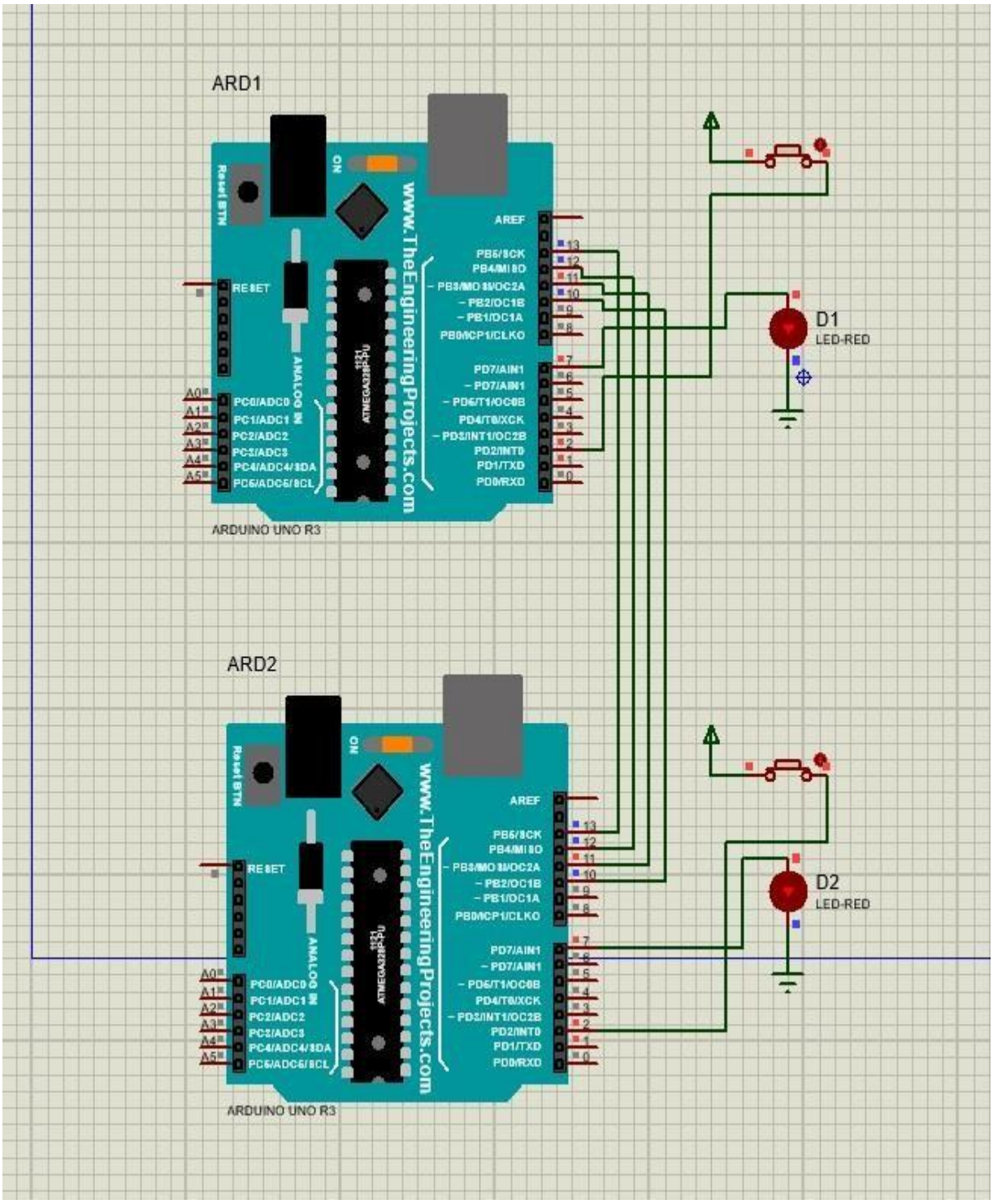


Fig: Push Button in Both Master & Slave Arduino is pressed and led of Both Master & Slave Arduino lit up

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 value received from slaveu
  {
    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);
}
```

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);
    }
}
```

Discussion:

The experiment we performed, "Communication between two Arduino Boards using SPI", aimed to demonstrate the Serial Peripheral Interface (SPI) protocol for communication between two microcontrollers, in this case, two Arduino boards. The SPI protocol is a synchronous serial communication protocol that allows for communication between devices over short distances at high speeds. During the experiment, we connected two Arduino boards together using SPI, with one board acting as the master and the other as the slave. The master board sent data to the slave board, which responded by sending data back to the master. One of the key aspects of this experiment is the use of the SPI protocol, which is widely used in embedded systems, particularly for communication between microcontrollers and peripherals such as sensors, displays, and memory devices. By understanding how SPI works and how to implement it in code, we gained valuable experience that can be applied in future projects. In addition to learning about SPI, the experiment also provided an opportunity to practice coding and debugging skills. We had to write code for both the master and slave boards, ensuring that the data transfer was working correctly and troubleshooting any issues that arose during the experiment. Overall, the "Communication between two Arduino Boards using SPI" experiment was a valuable learning experience that allowed us to gain hands-on experience with the SPI protocol and practice important coding and debugging skills. By successfully completing the experiment, we have developed skills that will be useful in future projects and in our future career as engineer.

References:

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