American International University- Bangladesh (AIUB)
Faculty of Engineering (EEE)

Course Name :	MICROPROCESSOR	**Course Code :**	EEE 4103
Semester :	SUMMER 24-25	**Sec :**	A
Lab Instructor :	Md Sajid Hossain		

Experiment No :	07.2
Experiment Name :	*Communication between two Arduino Boards using SPI*

Submitted by (NAME):	*RUDRA DHAR*	**Student ID:**	*23-51381-1*

Group Members	**ID**	**Name**
1.	23-50176-1	MD.Minhazul Mostofa Pranto
2.	23-51381-1	Rudra Dhar
3.	22-47271-1	Mohamodul Hasan Taj
4.	23-50280-1	Arnab Hasan Kabbo
5.	21-44814-1	Md.Enam Rahman Eraz
6.		
7.		

Performance Date :	**27/08/2025**	**Due Date :**	**12/09/2025**

Marking Rubrics (to be filled by Lab Instructor)

Category	Proficient [6]	Good [4]	Acceptable [2]	Unacceptable [1]	Secured Marks
Theoretical Background, Methods & procedures sections	All information, measures and variables are provided and explained.	All Information provided that is sufficient, but more explanation is needed.	Most information correct, but some information may be missing or inaccurate.	Much information missing and/or inaccurate.	
Results	All of the criteria are met; results are described clearly and accurately;	Most criteria are met, but there may be some lack of clarity and/or incorrect information.	Experimental results don't match exactly with the theoretical values and/or analysis is unclear.	Experimental results are missing or incorrect;	
Discussion	Demonstrates thorough and sophisticated understanding. Conclusions drawn are appropriate for analyses;	Hypotheses are clearly stated, but some concluding statements not supported by data or data not well integrated.	Some hypotheses missing or misstated; conclusions not supported by data.	Conclusions don't match hypotheses, not supported by data; no integration of data from different sources.	
General formatting	Title page, placement of figures and figure captions, and other formatting issues all correct.	Minor errors in formatting.	Major errors and/or missing information.	Not proper style in text.	
Writing & organization	Writing is strong and easy to understand; ideas are fully elaborated and connected; effective transitions between sentences; no typographic, spelling, or grammatical errors.	Writing is clear and easy to understand; ideas are connected; effective transitions between sentences; minor typographic, spelling, or grammatical errors.	Most of the required criteria are met, but some lack of clarity, typographic, spelling, or grammatical errors are present.	Very unclear, many errors.	
Comments:				Total Marks (Out of):	

Title:

Communication between two Arduino Boards using SPI

Abstract:

Serial Peripheral Interface (SPI) is a high-speed synchronous communication protocol commonly used for short-distance communication between microcontrollers and peripheral devices. In this project, two Arduino boards are configured to communicate with each other using SPI: one as the Master and the other as the Slave.

The Master Arduino initiates and controls the data transfer by generating the clock signal (SCK), while the Slave Arduino responds to the Master's commands. Data is exchanged using the MOSI (Master Out Slave In) and MISO (Master In Slave Out) lines, with an additional SS (Slave Select) pin to activate the slave device.

This setup allows for bidirectional data transfer between the two Arduino boards, enabling applications such as sensor data exchange, distributed control, and device coordination. The project demonstrates the wiring, configuration, and coding required for SPI communication and highlights the advantages of SPI, such as speed and efficiency, compared to other serial protocols like UART or I²C.

Introduction:

In modern embedded systems, efficient communication between microcontrollers is essential for building complex and reliable applications. Various communication protocols such as UART, I²C, and SPI are widely used to exchange data between devices. Among these, the Serial Peripheral Interface (SPI) is one of the most popular due to its simplicity, high speed, and full-duplex capability.

SPI uses a master–slave architecture, where the master device controls the clock signal and initiates communication, while the slave device responds accordingly. Data is transferred through four main connections:

- *MOSI (Master Out Slave In)*
- *MISO (Master In Slave Out)*
- *SCK (Serial Clock)*
- *SS (Slave Select)*

In this project, two Arduino boards are used to demonstrate SPI communication. One Arduino is configured as the Master, responsible for controlling the communication, while the other acts as the Slave, receiving and sending data back. This project helps to understand how two microcontrollers can exchange information quickly and reliably using SPI.

The implementation provides a foundation for real-world applications such as sensor data sharing, distributed control systems, robotics, and IoT devices, where multiple controllers need to work together.

Theory and Methodology :

The Serial Peripheral Interface (SPI) is a synchronous serial communication protocol that allows high-speed data transfer between microcontrollers and peripheral devices. It follows a master–slave architecture:

- *The Master generates the clock (SCK) and initiates communication.*
- *The Slave responds according to the Master's instructions.*

SPI uses four primary connections:

1. *MOSI (Master Out Slave In): Sends data from Master to Slave.*
2. *MISO (Master In Slave Out): Sends data from Slave to Master.*
3. *SCK (Serial Clock): Generated by Master to synchronize data transfer.*
4. *SS (Slave Select): Activates the Slave device for communication.*

Unlike UART and I²C, SPI supports full-duplex communication, meaning data can be transmitted and received simultaneously. It is faster than I²C because it does not use addressing and supports multiple slaves through individual SS pins.

1. *Hardware Setup:*
    - *Two Arduino boards are connected via SPI pins:*
        - *Master → MOSI → Slave*
        - *Slave → MISO → Master*
        - *Master → SCK → Slave*
        - *Master → SS → Slave*
        - *Common GND connection between boards.*
2. *Software Implementation:*
    - *Include the <SPI.h> library.*
    - *Configure one Arduino as Master and the other as Slave.*
    - *Master sends a data byte to the Slave.*
    - *Slave receives the data and optionally sends a response back via MISO.*
3. *Testing Procedure:*
    - *Upload the Master code to one Arduino and the Slave code to the other.*
    - *Open the Serial Monitor to observe communication.*
    - *Verify that data sent by the Master is received by the Slave and vice versa.*
4. *Applications:*
    - *Multi-controller systems (robotics, IoT).*
    - *Fast data transfer between sensors and controllers.*
    - *Distributed embedded systems where tasks are shared across devices.*

Pre-Lab Homework:

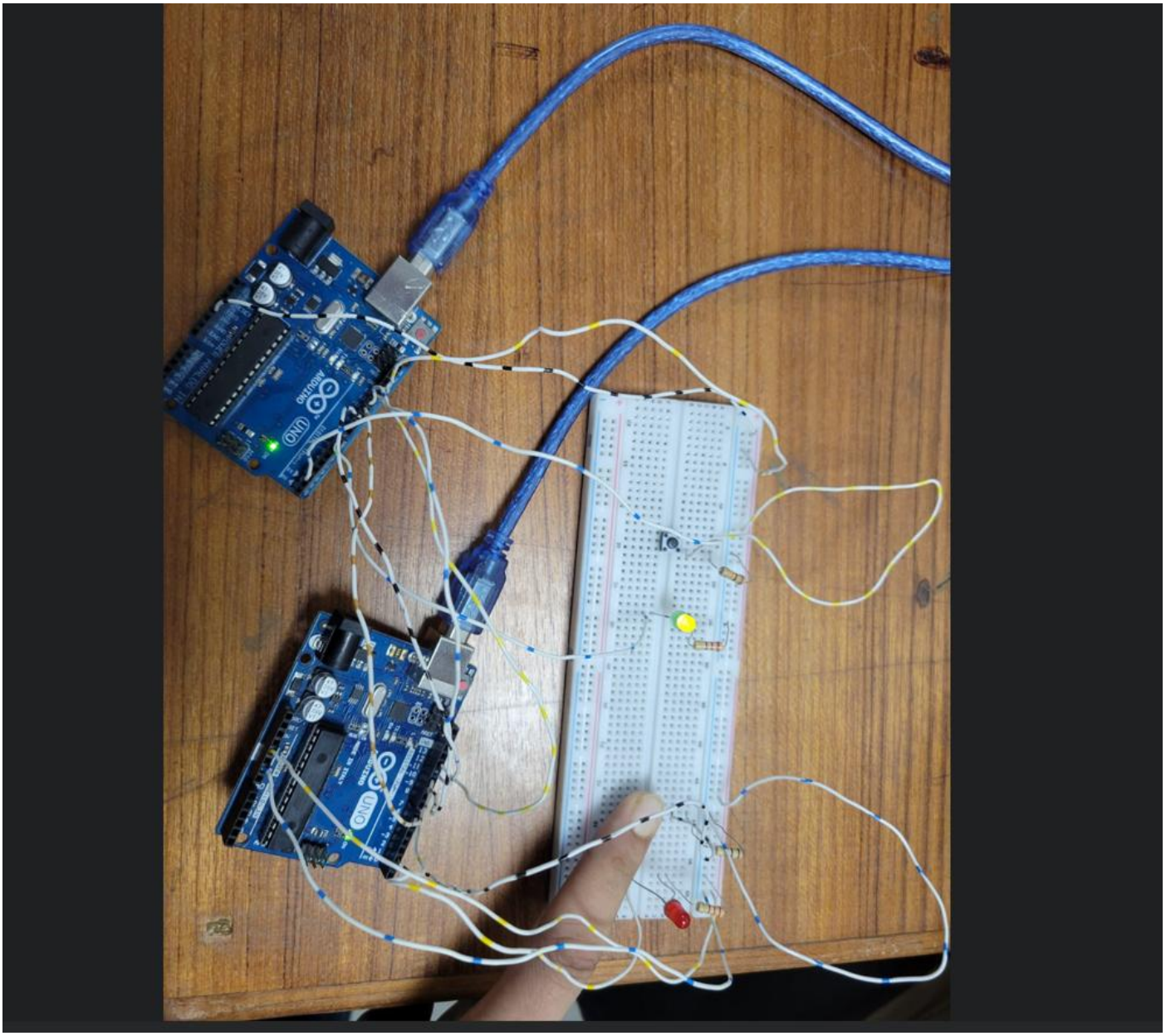

Fig: Blink the yellow LED

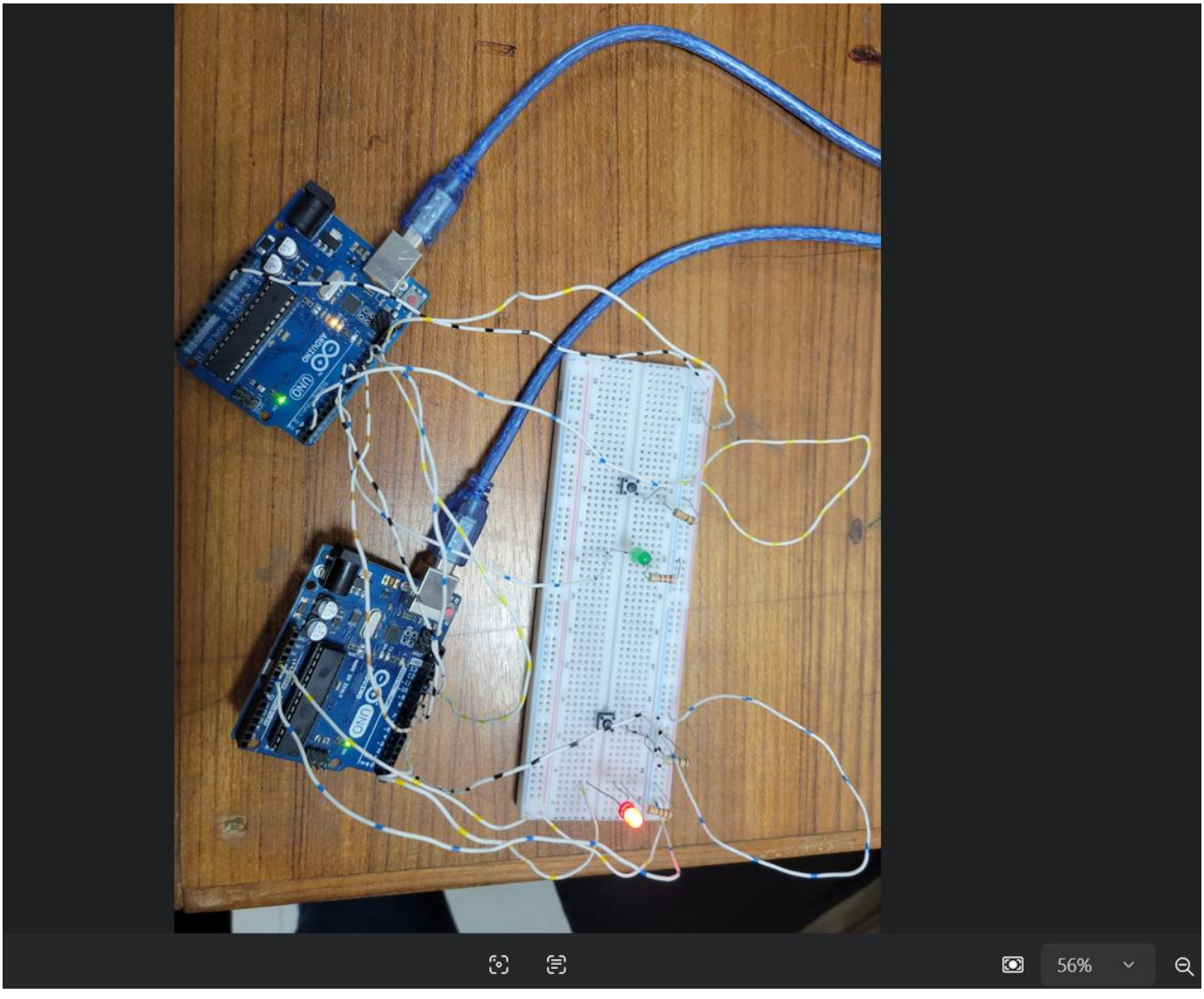

Fig: Blink the red LED

Apparatus:

1.Arduino UNO(2)
2. LED(2)
3.Push Button(2)
4.Resistors 10k ,2.2k(2+2)
5.Breadboard
6.Connecting Wires

Precautions:

1. *Common Ground*
    - *Always connect the GND of Master and Slave boards together. Without a common ground, data will be corrupted.*
2. *Voltage Level Matching*
    - *Make sure both Arduinos use the same logic level (5V ↔ 5V or 3.3V ↔ 3.3V).*
    - *If one is 5V and the other is 3.3V, use a level shifter.*
3. *Correct Wiring*
    - *Double-check pin connections:*
        - *MOSI → MOSI*
        - *MISO → MISO*
        - *SCK → SCK*
        - *SS (CS) → SS*
    - *Wrong wiring can damage pins or fail communication.*
4. *Master–Slave Role*
    - *Only one Arduino should act as Master, others as Slaves.*
    - *Do not set multiple Masters unless you know how to handle bus arbitration.*
5. *Use Pull-up/Pull-down if Needed*
    - *Sometimes SS (Slave Select) pin may float; ensure it is properly set HIGH when idle.*
6. *Clock Speed (SCK)*
    - *Use a suitable SPI clock speed (e.g., SPI_CLOCK_DIV8).*
    - *Too high clock speed can cause data corruption.*
7. *Stable Power Supply*
    - *Both Arduinos must have a stable power source; unstable power leads to communication errors.*
8. *Data Synchronization*
    - *Make sure to wait until transmission is complete before sending new data.*
9. *Noise & Cable Length*
    - *Keep SPI wires short; long wires can introduce noise and errors.*
10. *Debugging*
    - *Use Serial.print() to check sent/received data while testing.*

Experimental Procedure:

1. *Setup the Hardware*
    - *Take two Arduino boards (Arduino Uno recommended).*
    - *Decide one as Master and the other as Slave.*
    - *Connect the pins as follows:*
        - *Master MOSI (Pin 11) → Slave MOSI (Pin 11)*
        - *Master MISO (Pin 12) → Slave MISO (Pin 12)*
        - *Master SCK (Pin 13) → Slave SCK (Pin 13)*
        - *Master SS (Pin 10) → Slave SS (Pin 10)*
        - *Connect GND ↔ GND between both Arduinos.*
2. *Power the Boards*
    - *Connect both Arduinos to the computer or an external 5V supply.*
    - *Ensure stable voltage.*
3. *Write the Master Code*
    - *Initialize SPI using SPI.begin().*
    - *Define the SS (Slave Select) pin as OUTPUT.*
    - *Send data to the Slave using SPI.transfer().*
4. *Write the Slave Code*

- *Enable SPI by setting MISO as OUTPUT.*
- *Enable SPI interrupt (if needed).*
- *Receive the data sent by the Master and process it (e.g., turn on LED, print in Serial Monitor).*

5. *Upload Codes*
   - *Upload Master code to Master Arduino.*
   - *Upload Slave code to Slave Arduino.*
6. *Run the Experiment*
   - *Start Serial Monitor to observe communication.*
   - *Press a button or send commands from the Master.*
   - *Observe response on the Slave (like LED ON/OFF or message received).*
7. *Test Different Data*
   - *Try sending different numbers, characters, or sensor values.*
   - *Verify if Slave correctly receives and processes them.*
8. *Record Observations*
   - *Note the data sent from the Master and the exact data received at the Slave.*

Simulation :

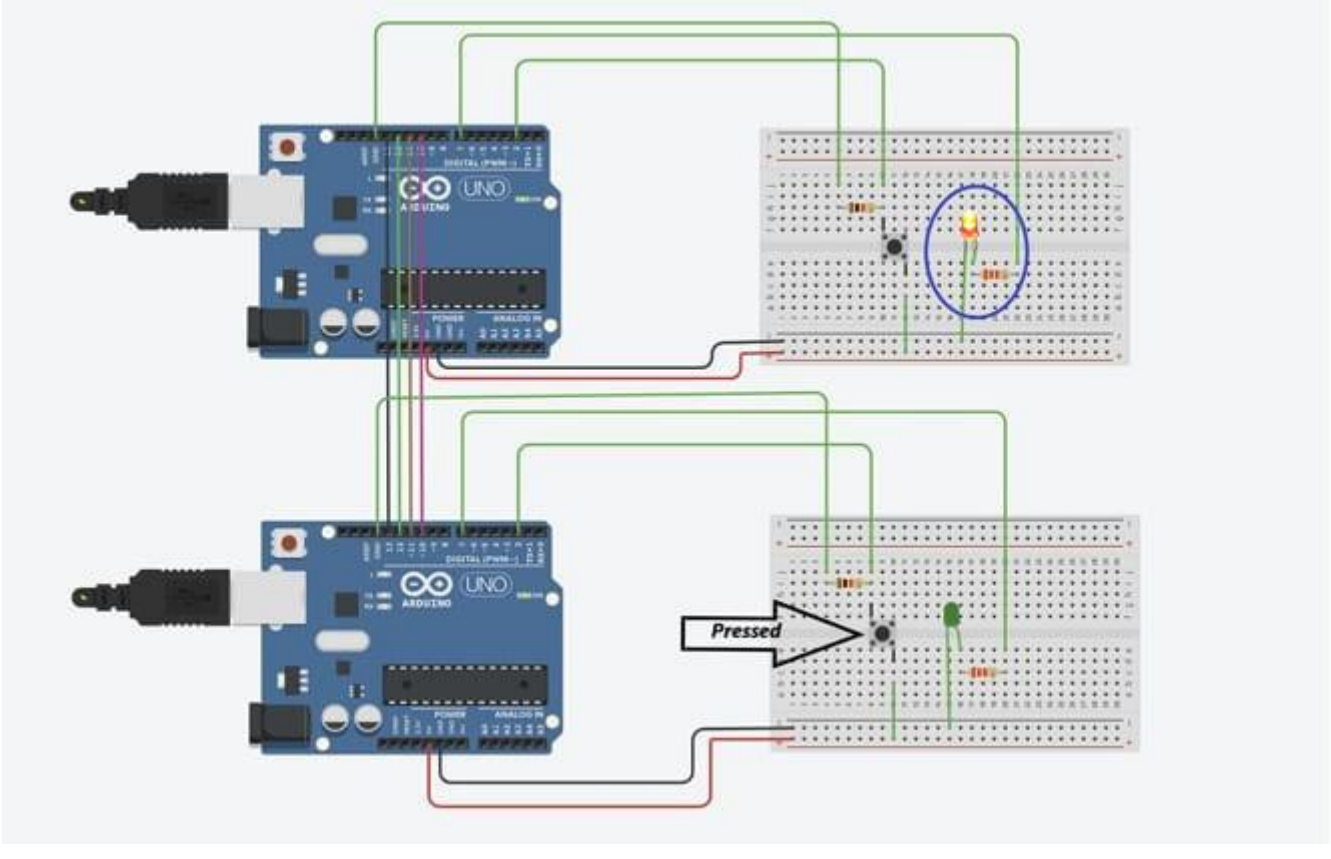

Fig :Blink the RED LED

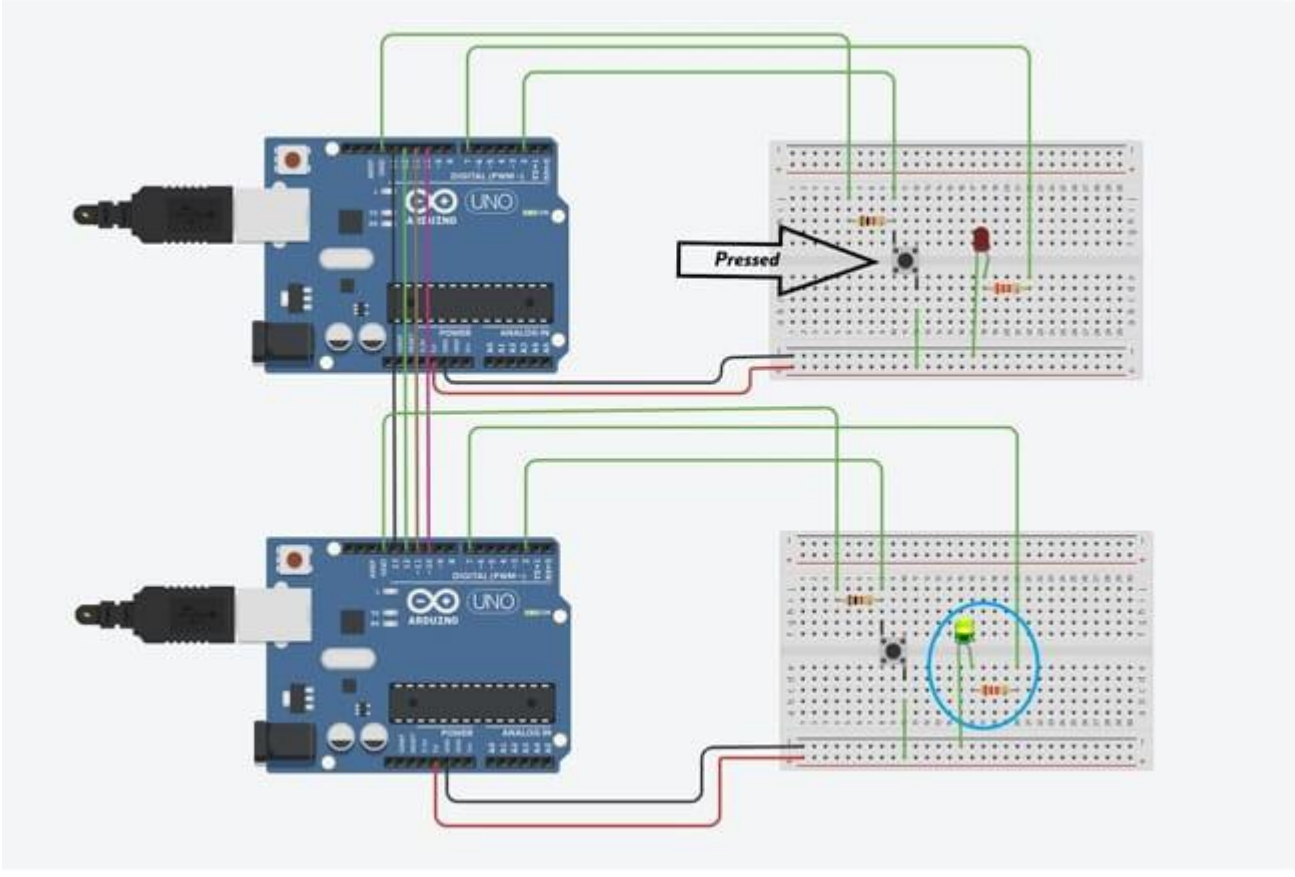

Fig: Blink the GREEN LED

Results/Finding(s):

Result was so satisfying.All the elements worked perfectly and also the code was perfectly runed.

Answer to Report Question:

1. Master/Controller Arduino Code:

```
#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){
```

```
send, 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 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);

}
```

2. Slave/Peripheral Arduino Code:

```
#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
```

```
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:

In this experiment, communication between two Arduino boards was successfully established using the Serial Peripheral Interface (SPI) protocol. SPI is a synchronous communication method that uses a master–slave architecture. The Master controls the clock (SCK) and initiates data transfer, while the Slave responds accordingly.

The experiment demonstrated how data can be reliably transferred between the two boards using four main lines: MOSI, MISO, SCK, and SS. A common ground was essential to ensure proper reference voltage. The Master Arduino was able to send data using SPI.transfer(), and the Slave Arduino received the data by configuring its SPI registers.

It was observed that the speed and reliability of SPI communication depend on correct wiring, clock speed configuration, and synchronization between the Master and Slave. Unlike asynchronous protocols such as UART, SPI ensures faster and more efficient communication but requires more physical connections.

During the test, data transmission worked correctly when the clock divider was set to a suitable value (e.g., SPI_CLOCK_DIV8). Too high clock speeds introduced errors, while lower speeds ensured stable communication. The experiment also showed that SPI can be extended to multiple slave devices, but proper selection using Slave Select (SS) pins is necessary.

Overall, the experiment provided practical knowledge of how SPI works in real embedded systems. It emphasized the importance of correct pin configuration, grounding, and synchronization. This type of communication can be used in real-world applications like sensor data acquisition, memory card interfacing, and communication between multiple microcontrollers.

Conclusions:

From this experiment, it can be concluded that:

1. *The SPI protocol is an efficient and reliable method of communication between two Arduino boards.*
2. *A Master–Slave relationship is necessary, where the Master generates the clock and initiates communication.*
3. *Proper wiring of MOSI, MISO, SCK, SS, and GND is essential for correct data transfer.*
4. *The communication speed depends on the SPI clock setting; stable operation was achieved at moderate clock speeds.*
5. *The experiment demonstrated the ability of the Master to send data and the Slave to correctly receive and respond to it.*
6. *SPI offers advantages such as high speed and synchronous transfer, but it requires more physical connections compared to UART.*
7. *This technique can be applied in real-world applications, such as sensor interfacing, memory devices, and multi-microcontroller systems.*

Overall, the experiment successfully showed the working principle of SPI and enhanced the understanding of synchronous serial communication between microcontrollers.

Reference(s):

 Arduino Official Documentation – SPI Library
https://www.arduino.cc/en/reference/SPI

 Banzi, M. & Shiloh, M. (2014). Getting Started with Arduino. Maker Media, 3rd Edition.

 Simon Monk (2016). Programming Arduino: Getting Started with Sketches. McGraw-Hill Education.

 Online Tutorial – SPI Communication Between Two Arduinos
https://circuitdigest.com/microcontroller-projects/arduino-spi-communication-tutorial

 Atmel (Microchip) Datasheet – ATmega328P Microcontroller (used in Arduino Uno).