

**AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH (AIUB)**

**FACULTY OF SCIENCE & TECHNOLOGY**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**MICROPROCESSOR AND EMBEDDED SYSTEMS**

**Spring 2024-2025**

**Section:** Q**, Group: 08**

**LAB REPORT ON**

Communication between two Arduino Boards using SPI

**Supervised By**

Protik Parvez Sheikh

**Submitted By**

MD. TANJIL TASHRIK ZIM

**Group Member**

|  |  |  |
| --- | --- | --- |
| **No.** | **Name** | **ID** |
| 1. | **MD. TANJIL TASHRIK ZIM** | **22-48021-2** |
| 2. | MD. AL-IMRAN SAYEM | 22-48023-2 |
| 3. | MD. ABRAR RAFID SHARIAR | 22-48055-2 |
| 4. | HUMAIRA ZANNAT | 22-47789-2 |
| 5. | SANJUKTA DAS | 22-47801-2 |

**Date of Submission:** May13, 2025

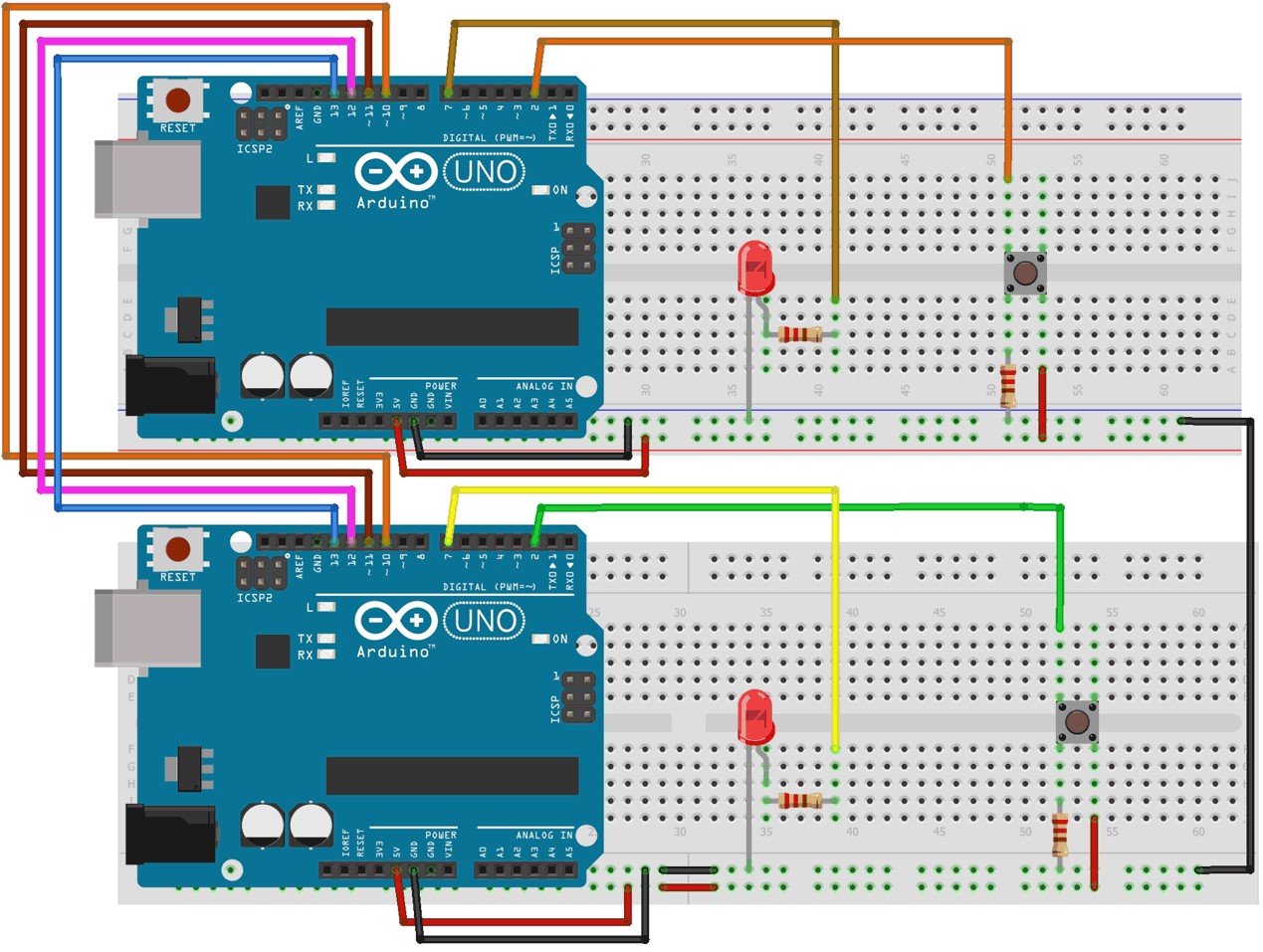
**Title:**  Communication between two Arduino Boards using SPI.

**Objective:** The objective of the "Communication between two Arduino Boards using SPI" experiment is to demonstrate the implementation of Serial Peripheral Interface (SPI) for inter-board communication in embedded systems. By connecting two Arduino boards via SPI, the experiment aims to establish a reliable and efficient data exchange mechanism. This involves configuring one Arduino as the master device and the other as the slave, enabling synchronous serial communication between them. Through this experiment, participants gain practical experience in utilizing SPI protocols for real-time data transfer, enabling applications such as sensor networks, control systems, and collaborative multi-device projects in the field of electronics and robotics.

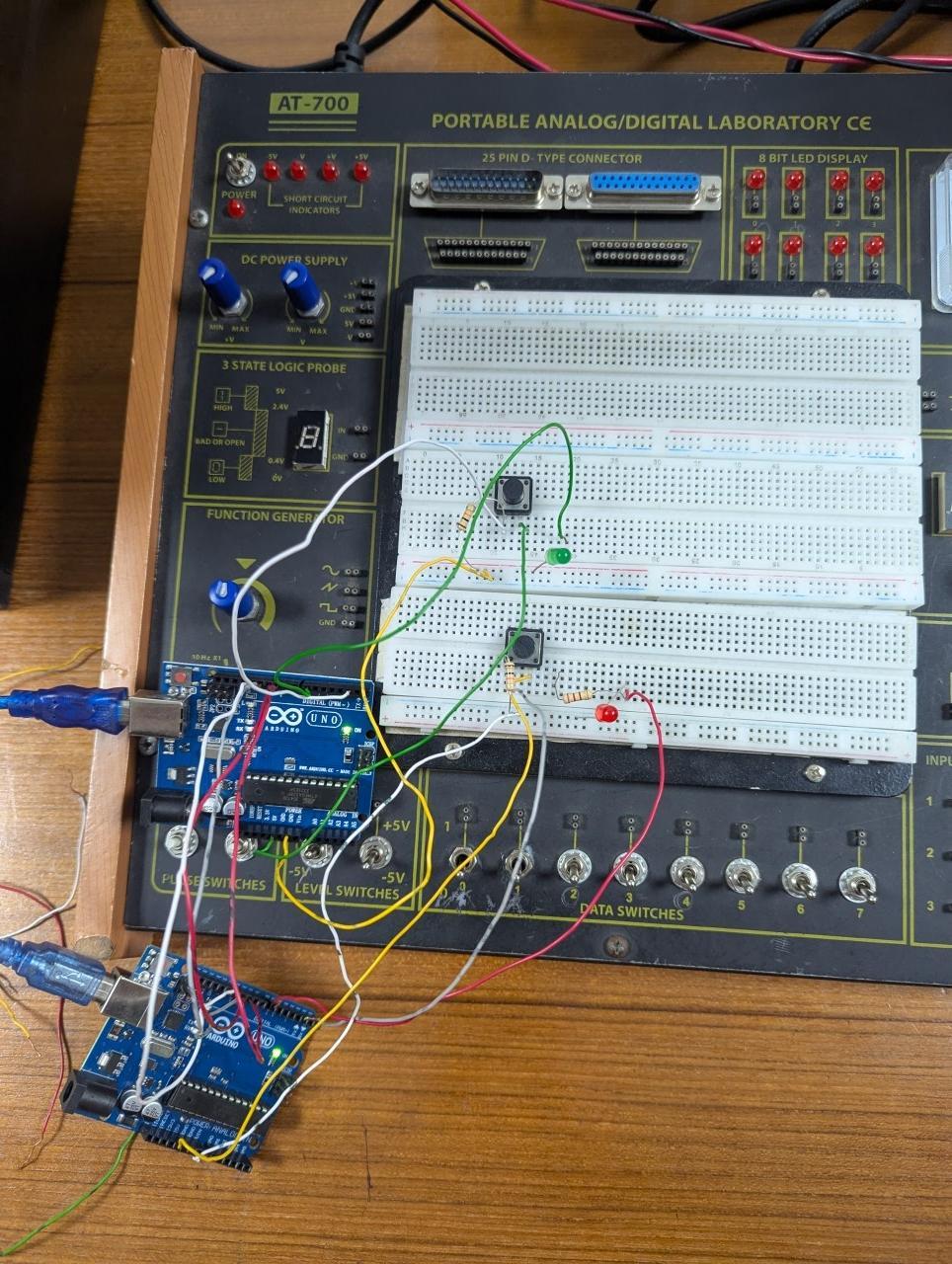
**Apparatus:**

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

**Experimental Setup:**



**Fig-1:** Two Arduino board’s pin connections for SPI communications (schematic diagram)

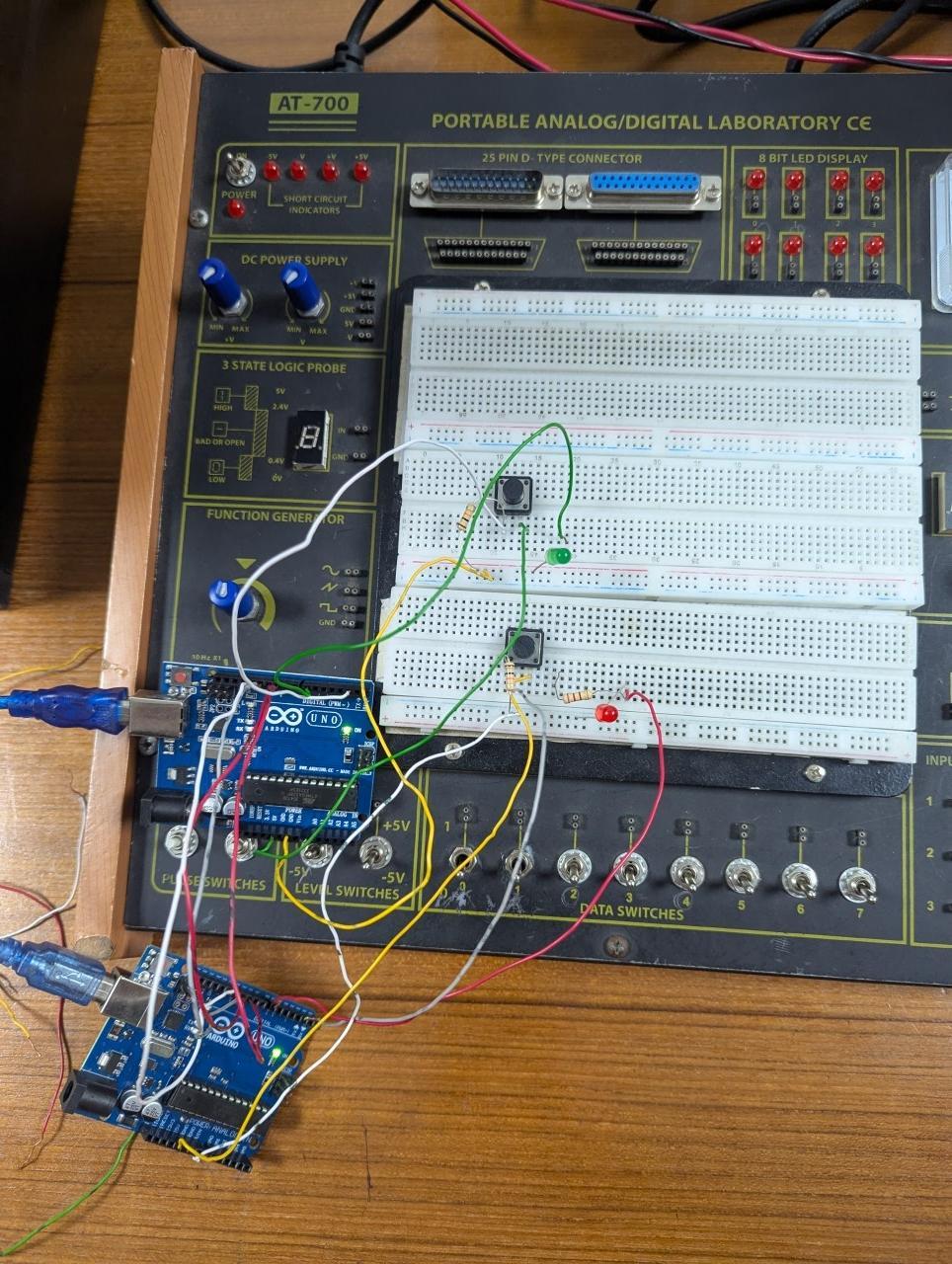
** Fig-2:** Experimentalsetup

**Code:**

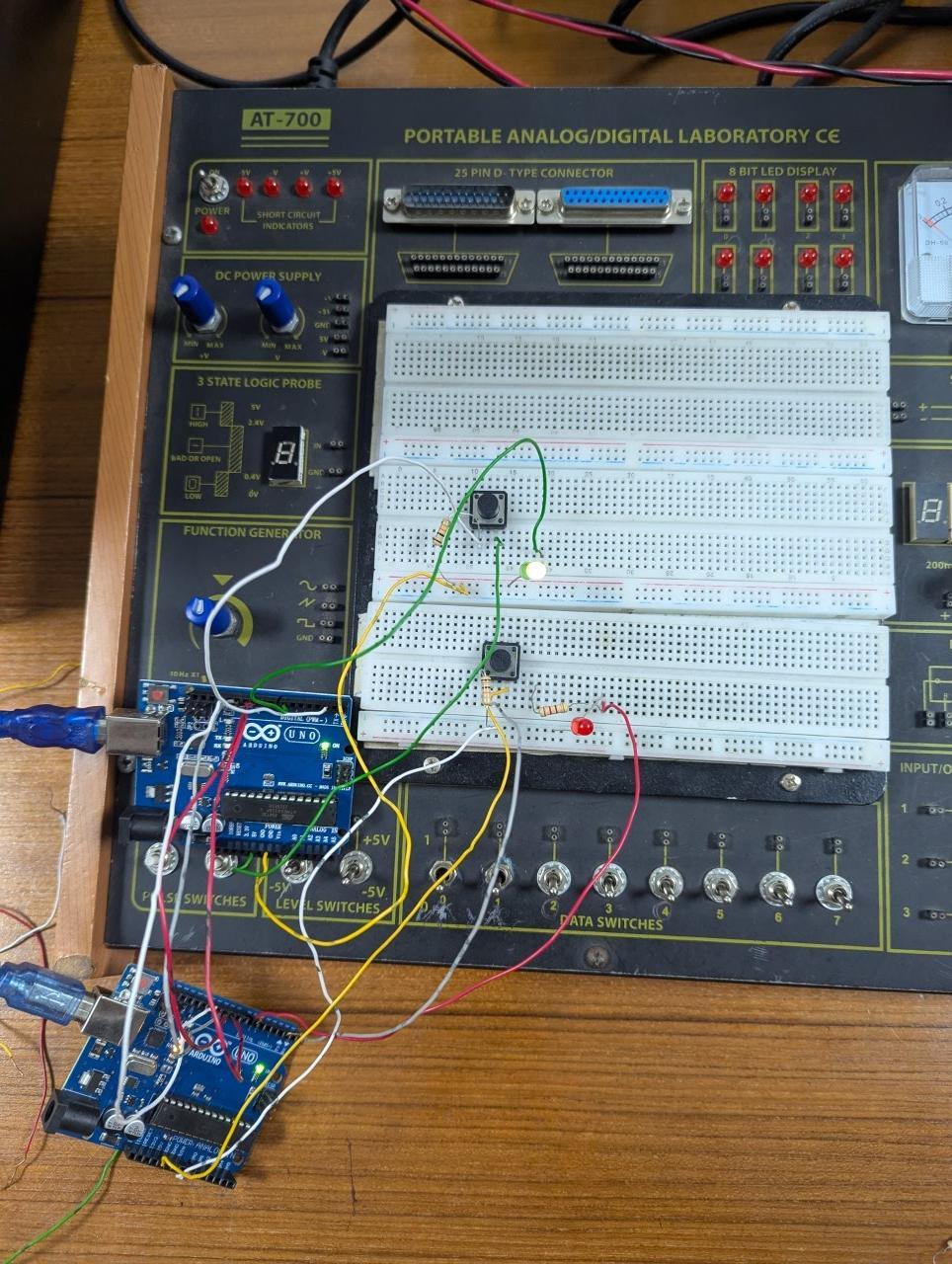
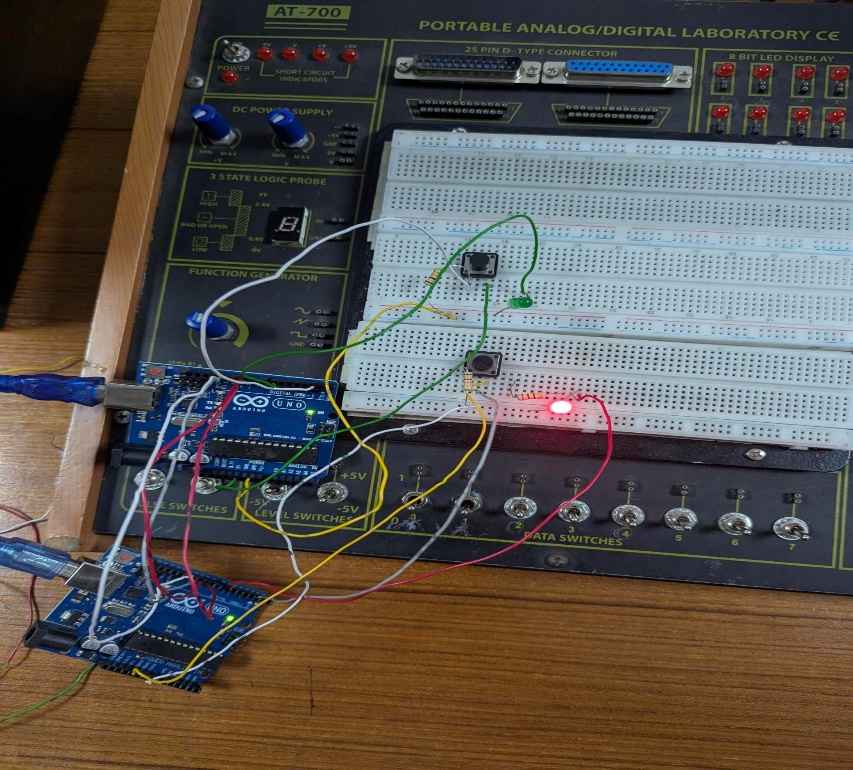
|  |
| --- |
| **Code for Master Device** |
| #include<SPI.h> //Library for SPI  #define LED 7  #define ipbutton 2  int buttonvalue;  int x;  void setup (void){  Serial.begin(115200); //Starts Serial Communication at Baud Rate 115200  pinMode(ipbutton,INPUT); //Sets pin 2 as input  pinMode(LED,OUTPUT); //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);  } |

|  |
| --- |
| **Code for Slave Device** |
| #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)  {  x = 1;  }  else  {  x = 0;  }  Slavesend = x;  SPDR = Slavesend;  //To set the value of x to send to Master  //Sends the x value to the Master via SPDR  delay(1000);  }  } |

**Experimental Results:**

****

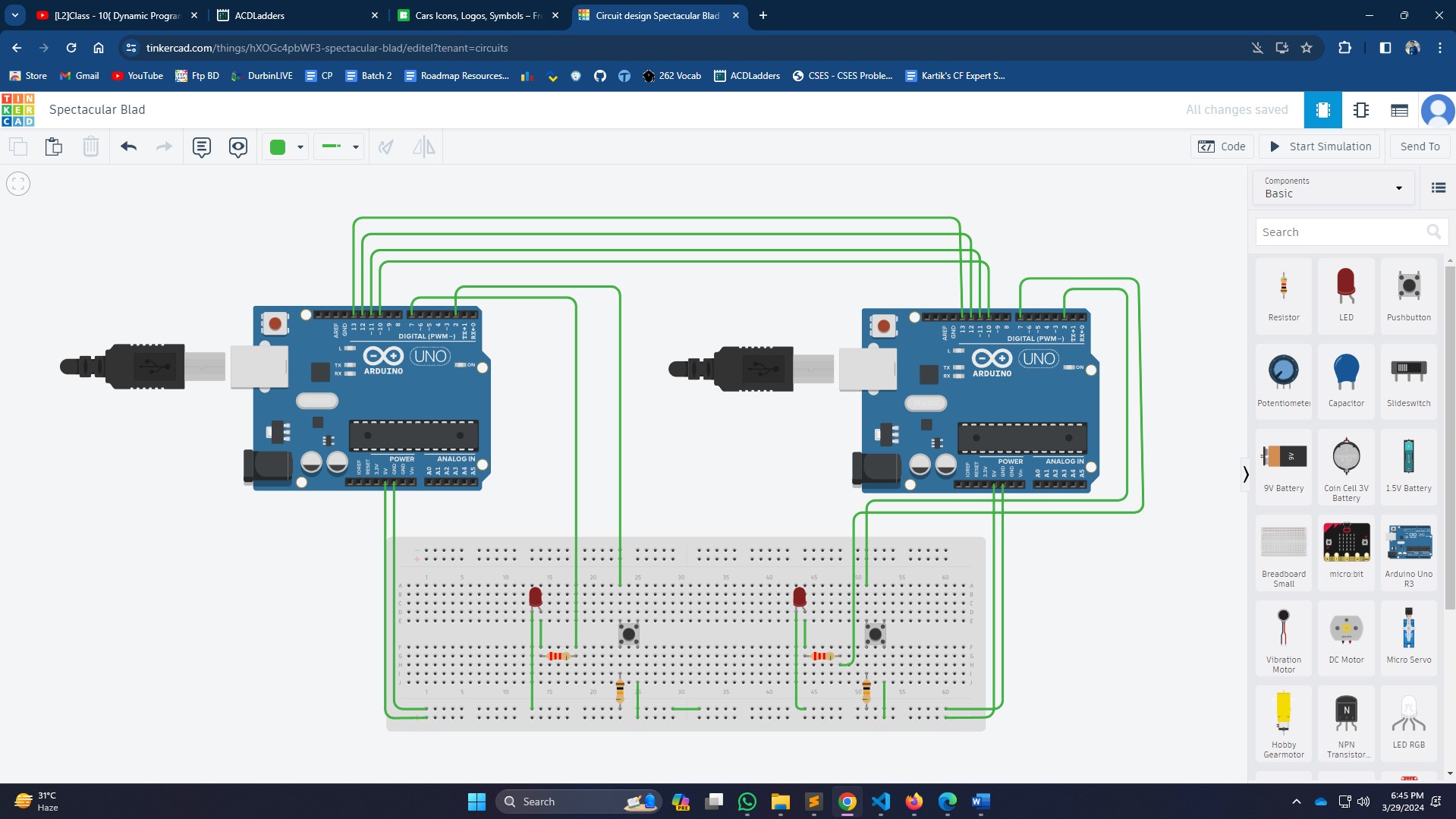
**Fig-3**: Connection between master and slave device

****

**Fig-4:** Clicked master device button and light **Fig-5:** Clicked salve device button and light on for

On for slave device Master device

**Simulation Setup:**



**Fig-6**: Simulation of SPI communication

**Discussion and Conclusion:**

Implementing communication between Arduino boards using SPI offers several advantages. SPI is a synchronous serial communication protocol that enables high-speed data transfer between devices, making it suitable for real-time applications. By designating one Arduino as the master and the other as the slave, precise control over data transmission is achieved, facilitating reliable communication. Additionally, SPI requires minimal hardware overhead, utilizing only a few pins for communication, which conserves resources and simplifies circuitry. This experiment enhances understanding of SPI configuration, including settings such as clock polarity, phase, and data order, which are crucial for successful communication.

However, there are limitations and considerations to acknowledge. SPI communication is typically point-to point, restricting the number of devices that can be connected directly. This can pose challenges in scenarios requiring communication among multiple nodes, necessitating additional hardware or alternative communication protocols. Moreover, while SPI offers high-speed communication, longer distances between devices may introduce signal degradation, requiring signal conditioning or the use of additional components.

In conclusion, the experiment demonstrates the effectiveness of SPI for inter-board communication in Arduino-based systems. Participants gain valuable insights into configuring SPI communication and leveraging its benefits for efficient data exchange. By understanding the nuances of SPI configuration and addressing its limitations, developers can effectively integrate SPI into their projects to enable seamless communication between Arduino boards. Overall, the experiment underscores the importance of selecting appropriate communication protocols and mastering their implementation for building robust and scalable embedded systems

**Reference(s):**

1. https://www.arduino.cc/.
2. ATMega328 manual
3. https://www.avrfreaks.net
4. http://maxembedded.com