COMMUNICATION PROTOCOLS IN MICROCONTROLLERS: A COMPARATIVE STUDY OF UART, I^2C AND SPI

Brief Theory of Protocols

A. UART (Universal Asynchronous Receiver-Transmitter)

UART is an asynchronous serial communication protocol that does not require a shared clock signal. Instead, it relies on pre-defined baud rates and start/stop bits to synchronize data transfer between a transmitter and a receiver.

- **Important Terms:** Baud Rate, Start/Stop Bits, TX (Transmit), RX (Receive).

B. I^2C (Inter-Integrated Circuit)

I^2C is a synchronous, multi-master, multi-slave, packet-switched, single-ended, serial communication bus. It uses only two bidirectional lines: Serial Data (SDA) and Serial Clock (SCL).

- **Important Terms:** SDA, SCL, Master-Slave, Device Addressing, Acknowledgement (ACK/NACK).

C. SPI (Serial Peripheral Interface)

SPI is a high-speed, synchronous serial communication protocol used for short-distance communication. It operates in full-duplex mode using a master-slave architecture with a dedicated clock line.

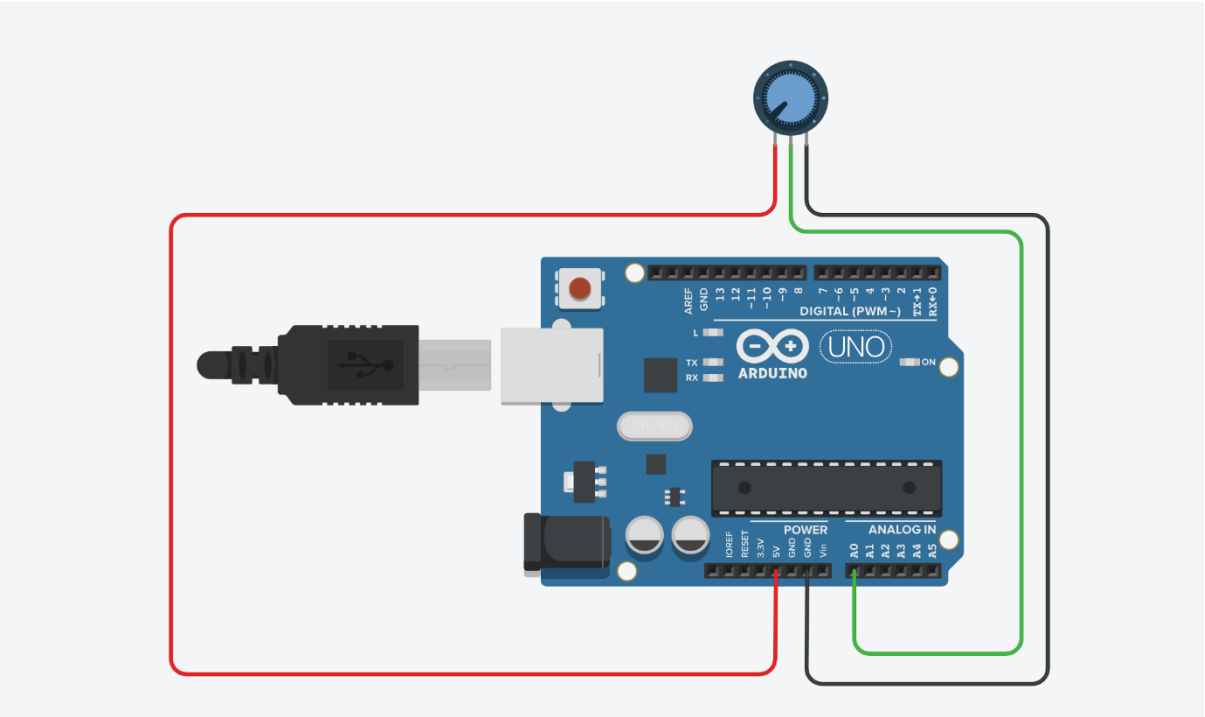
- **Important Terms:** MOSI (Master Out Slave In), MISO (Master In Slave Out), SCK (Serial Clock), SS/CS (Slave Select/Chip Select).

Hands-On Project Observations

Task 1: UART Communication Simulation

- **Aim:** To simulate UART serial communication between an Arduino Uno and a PC.
- **Components:** Arduino Uno, Potentiometer.
- **Observations:** Successful asynchronous data transmission was verified. Analog values from the potentiometer were read via analogRead() and transmitted to the Tinkercad Serial Monitor in real-time using Serial.begin(9600).

Circuit:



Code:

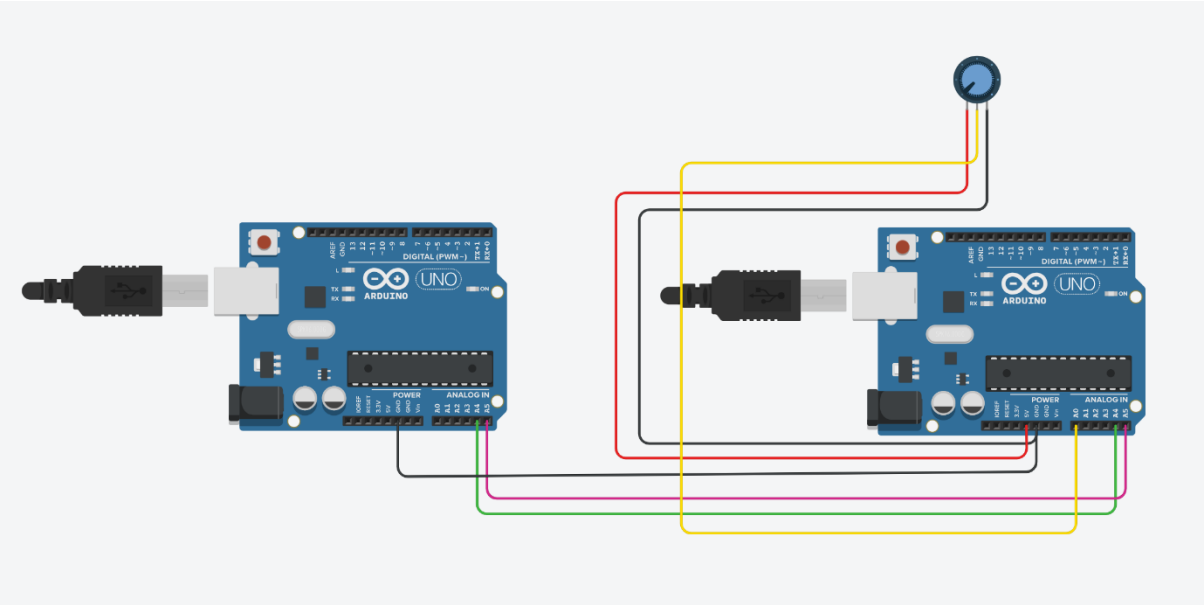
```
void setup() {
  Serial.begin(9600);   // Initialize UART communication
}

void loop() {
  int sensorValue = analogRead(A0); // Read potentiometer
  Serial.print("Potentiometer Value: ");
  Serial.println(sensorValue);      // Send data via UART
  delay(500);
}
```

Task 2: I²C Communication Simulation

- **Aim:** To simulate I²C communication between two Arduino Uno boards (Master-Slave).

- **Components:** Arduino Uno (Master), Arduino Uno (Slave), Potentiometer (connected to Slave).
- **Observations:** The Master successfully requested and received sensor data from the Slave using the Wire.h library. Proper I^2C addressing was observed, ensuring the Master communicated only with the intended Slave.

Circuit:



Code:

Master-

```
#include <Wire.h>

void setup() {
  Wire.begin();       // Initialize as Master
  Serial.begin(9600);  // UART to PC
}

void loop() {
  Wire.requestFrom(0x08, 1); // Request 1 byte from Slave
  if (Wire.available()) {
    int receivedValue = Wire.read();
    Serial.print("Received Value: ");
    Serial.println(receivedValue);
  }
```

```
  delay(500);
}
```

Slave-

```
#include <Wire.h>
int sensorValue = 0;
void setup() {
  Wire.begin(0x08);          // Initialize as Slave
  Wire.onRequest(sendData);  // Function to send data
}

void loop() {
  sensorValue = analogRead(A0); // Read potentiometer
  delay(100);
}
void sendData() {
  Wire.write(sensorValue >> 2); // Send 1 byte (0–255)
}
```

Task 3: SPI Communication Simulation

- **Aim:** To simulate SPI communication between two Arduino Uno boards.
- **Components:** Arduino Uno (Master), Arduino Uno (Slave), Push Button (connected to Slave).
- **Observations:** Synchronous data transfer was verified through a shared SCK line. The Master polled the Slave for the button state. It was observed that the Master must initiate the clock for the Slave to send its data via the MISO line.

Conclusion

Through these simulations, the fundamental differences in microcontroller communication were demonstrated. UART proved effective for simple PC-to-controller links, I^2C offered an efficient way to connect multiple devices with minimal wiring, and SPI provided the high-speed synchronous performance necessary for rapid data exchange.