Half Duplex Communication System using RS485 module

GROUP – 5

Name	Roll number
G. Navya Sree	22ec01019
D. Sai Sameeksha Vaishnavi	22ec01021
D. Keerthi Vijetha	22ec01048
V. Sujitha	22ec01047

INDEX

3.5 Potentiometer(x2)

3.6 Micro Servo motor(x1)

4. Software Requirements

4.1 Arduino IDE

4.2 Servo libraries

4.3 Serial monitor software

5. System Design

5.1 Circuit diagram

5.2 Description of communication protocol

6. Implementation

6.1 Programming Arduino for Half duplex communication

6.2 Pairing RS485 modules with Arduino boards

7. **Results and Testing**

7.1 Testing the half duplex communication

7.2 Range and reliability of communication

8. Applications

8.1 Potential applications of the half duplex communication system using RS485

8.2 Real-world scenarios where the system can be used

9. Challenges and Solutions

9.1 Common challenges faced during implementation

9.2 Solutions and workarounds adopted

10. Conclusion

10.1 Summary of the project

10.2 Achievements and limitations

10.3 Future enhancements and recommendations

11. References

List of books, articles and other resources referred to during the project

12. Appendices:

12.1 Code snippets

12.2 photographs and diagrams

1. Introduction:

1.1. Background:

RS-485 is a communication standard suitable for industrial, commercial, and home automation systems, offering multi-point connections, long-distance, high-speed data transmission, and reliable communication in noisy environments, while half-duplex systems enable data transmission in both directions.

1.2. Objective:

This project aims to create a half-duplex communication system using RS-485 modules, utilizing hardware and software components like Arduino boards, to establish a reliable and efficient network.

1.3. Scope of the Project:

The project focuses on designing and implementing a half-duplex communication system using RS-485 modules. It includes:

- Setting up hardware components (Arduino boards, RS-485 modules, power supply, and other components).
- Developing software for Arduino boards to enable half-duplex communication.
- Testing the communication system for range, reliability, and error handling.

1.4. Transmission modes of Computer Networks:

Transmission mode means data is transferred between two devices, also called communication mode. Buses and networks are designed to allow communication to occur between individual devices that are interconnected. There are 3 types of communication modes:

1. Simplex mode
2. Half duplex mode
3. Full duplex mode

Simplex mode: In simplex mode the communication is unidirectional. Only one of the 2 devices on a link can transmit, the other can only receive.

Half duplex mode: In half duplex mode, each station can transmit and receive, but not at the same time, if one is sending the other can only receive and vice versa.

Full Duplex mode: In full duplex mode, both the stations can transmit and receive data simultaneously.

2. Literature Review:

2.1. Half duplex communication:

Half-duplex communication is a communication method where data can be transmitted in both directions but not simultaneously, requiring devices to take turns transmitting and receiving data, making it suitable for systems with data flow in both directions.

Characteristics of Half-Duplex Communication:

Alternating Transmission and Reception: Devices communicate in turns, one transmitting while the other receives.

Use of Control Signals: In systems like RS-485, control signals may be used to enable and disable transmission and reception to manage the half-duplex mode.

Examples of Half-Duplex Communication:

Walkie-Talkies, Two-Wire RS-485 Bus, CSMA/CD Ethernet.

Advantages of Half-Duplex Communication:

Efficiency and Cost-Effective: Allows multiple devices to share the same communication channel and it reduces hardware costs due to fewer lines required and simplified transceiver design.

Long Distance: RS-485 and other similar systems can operate over long distances, making them suitable for industrial environments.

2.2. RS485 Module its Applications and Advantages:

RS-485, also known as TIA-485, is a communication standard for defining a physical layer of communication in a network. It is a widely used standard for serial data communication that supports long-distance, high-speed data transmission and multi-point connections.

Features and Characteristics:

Multi-point Communication: RS-485 is a versatile communication protocol that enables up to 32 devices to connect to the same bus, making it ideal for shared network applications.

Differential Signaling: RS-485 uses differential signaling over a twisted-pair cable to transmit data. This technique minimizes interference and allows for more reliable communication, even in noisy environments.

Long Distance Communication: RS-485 can support data transmission over long distances (up to 1.2 km or 4,000 ft) depending on the baud rate.

Data Rates: It supports high data rates, up to 10 Mbps at shorter distances, and lower rates at longer distances.

Two-Wire Interface: RS-485 uses two wires for communication (labeled A and B) and can optionally include a common ground wire for reference.

Half-Duplex and Full-Duplex Modes: RS-485 supports both half-duplex (one direction at a time) and full-duplex (both directions simultaneously) communication modes, depending on the transceiver design and the number of wires used.

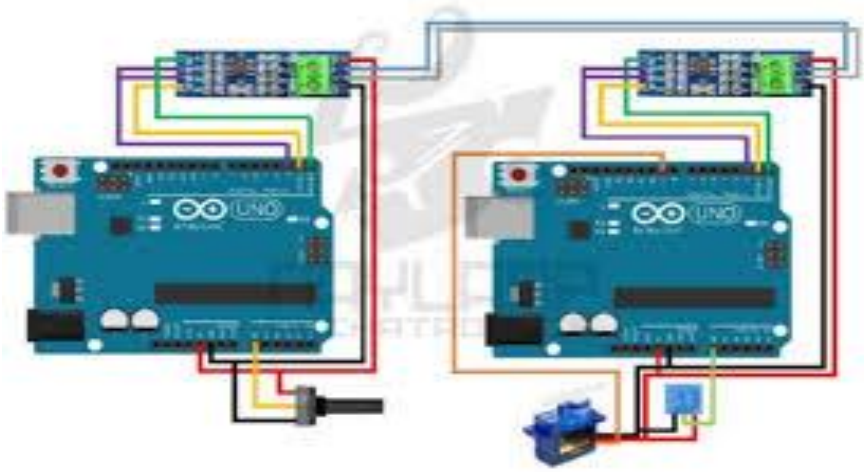
Common RS-485 Transceiver Chips:

MAX485: A commonly used RS-485 transceiver chip that can support half-duplex and full-duplex communication.

SN75176: Another popular transceiver chip for RS-485 communication.

5. System Design:

5.1. Circuit diagram:



5.2. Description of communication protocol:

In a half-duplex communication system using an RS485 module, the communication protocol governs how data is transmitted and received over the RS485 bus. Here's a description of the key aspects of such a protocol:

A) Frame Structure: Data is organized into frames, including start bit, data payload, optional parity bit, and stop bit(s).

B) Addressing: Each device in a multi-drop RS485 network is assigned a unique address. The protocol outlines how devices address each other in data frames.

The communication protocol for a half-duplex RS485 system establishes rules and procedures for efficient data exchange among devices connected to the bus, ensuring orderly and reliable communication.

6. Implementation:

6.1. Programming Arduino for Half duplex communication:

Arduino can be programmed for half-duplex communication using protocols like UART or software serial communication.

A) Connect the Hardware: Connect the communication lines (TX and RX) of our Arduino to the corresponding lines of the device we're communicating with another Arduino.

B) Initialize Serial Communication: In the setup function of our Arduino sketch, initialize the serial communication with the desired baud rate using the Serial.begin() function.

C) Write and Read Data: In the loop function, we can write data to the serial port using the Serial.write() function.

We can read incoming data from the serial port using the Serial.available() and Serial.read() functions.

D) Testing: Upload the code to our Arduino board using the Arduino IDE and open the serial monitor to view the communication.

Send data from the serial monitor, and we should see the Arduino echoing back the same data

6.2. Pairing RS485 modules with Arduino boards:

To pair RS485 modules with an Arduino board, follow these steps:

A) Choose RS485 Modules: We Selected RS485 modules compatible with our Arduino board and the communication requirements of our project.

RS485 modules typically include a transceiver chip (such as the MAX485) and terminal blocks for connecting wires.

B) Connecting Hardware: Connect the RS485 module to the Arduino board, connecting the DE (Driver Enable) and RE (Receiver Enable) pins to digital pins 2 and 3. Connect the TX (Transmit) and RX(Receive) pins to corresponding pins on Arduino. Connect the A and B terminals of the RS485 module to the A and B terminals of the communication network.

C) Writing Arduino Code: Write code in the Arduino IDE to initialize and communicate with the RS485 module. Below is a simple example employing the Software Serial library:

D) Test Communication: We Uploaded the code to our Arduino board and connected the RS485 module to the RS485 network and ensured that all connections are secure. Monitor the communication using a terminal program or serial monitor in the Arduino IDE.

Send and receive data to verify that communication is working as expected.

7. Results and Testing:

7.1. Testing the Half Duplex Communication:

Testing half-duplex communication involves verifying that data can be transmitted successfully in both directions, though not simultaneously. The process to test half-duplex communication is as follows:

A) Setup: Ensure that the devices or systems we're testing are properly connected and configured for half-duplex communication. This typically involves connecting them via a communication medium such as a serial port, RS-485, or a half-duplex Ethernet connection.

B) Test Plan Creation: Develop a test plan outlining the scenarios we want to test. This could include basic connectivity, data transmission in both directions, error handling, and performance testing.

C) Basic Connectivity Test: Verify that both devices can establish a connection and exchange data. This confirms that the physical and basic communication settings are configured correctly.

D) Data Transmission Test: Send data from one device to the other and vice versa. Ensure that each device can transmit and receive data successfully. This demonstrates that the half-duplex communication is functioning as expected.

By following these steps, we can effectively test half-duplex communication and ensure that it meets the required functionality and performance criteria.

7.2. Range and Reliability of Communication:

The range and reliability of half-duplex communication using an RS-485 module can vary depending on several factors:

A) Cable Quality: RS-485 uses twisted pair cables for good noise immunity and signal integrity. Higher quality cables with better shielding can extend range and improve reliability.

B) Transceiver Quality: Higher-quality transceivers can handle longer cable lengths and provide better noise immunity, leading to more reliable communication.

C) Termination: Proper termination of the RS-485 bus at both ends is crucial for preventing signal reflections and ensuring signal integrity.

D) Environmental Factors: Shielding cables and proper grounding can mitigate the impact of environmental conditions.

Overall, with proper configuration, RS-485 modules can support communication over distances of up to several kilometers and provide reliable data transmission in industrial and commercial applications.

8. Applications:

8.1. Potential applications of the half duplex communication system using RS485:

RS485 communication offers various applications due to its robustness, reliability, and noise immunity. Here are some potential applications:

A) Supervisory Control and Data Acquisition (SCADA): RS485 is commonly used in SCADA systems for remote monitoring and control of distributed equipment in industries like oil and gas, water management, and utilities.

B) Access Control Systems: RS485 is utilized in access control systems for communication between access control panels, card readers, and other components to manage entry and exit points in buildings and facilities.

C) Renewable Energy Systems: RS485 is used in solar and wind energy systems for communication between inverters, controllers, and monitoring devices to optimize energy production and monitor system performance.

8.2. Real-world scenarios where the system can be used:

Some real-world scenarios where RS485 communication systems are commonly used:

A) Building Management Systems: RS485 is employed in building management systems to integrate and control HVAC systems, lighting, security systems, and access control devices. This allows building operators to optimize energy usage, maintain occupant comfort, and enhance security.

B) Traffic Management Systems: RS485 is utilized in traffic management systems to connect traffic lights, sensors, cameras, and variable message signs. This facilitates traffic monitoring, signal coordination, and incident detection to improve traffic flow and safety on roads.

C) Oil and Gas Industry: RS485 communication is essential in the oil and gas industry for supervisory control and data acquisition (SCADA) applications. It

enables remote monitoring and control of pipelines, wellheads, and production facilities, enhancing operational efficiency and safety.

9. Challenges and Solutions:

9.1. Common challenges faced during implementation:

Implementing Half-Duplex Communication using RS-485 modules presents several challenges, including:

A) Signal Integrity: Maintaining signal integrity over long distances is challenging due to signal attenuation, impedance mismatches, and environmental noise.

B) Collision Detection: Devices share the communication medium, leading to potential collisions.

C) Synchronization: Devices must synchronize their transmission and reception timings to avoid data corruption and collisions.

D) Error Detection and Handling: The deployment of half-duplex communication using RS-485 modules requires robust error detection and handling mechanisms to ensure data integrity and reliability.

9.2. Solutions and workarounds Adopted:

To mitigate the challenges faced during the implementation of half-duplex communication using RS-485 modules, several solutions can be adopted.

A) Signal Integrity Solutions:

- Use high-quality twisted-pair cables with proper shielding.
- Implement proper termination at both ends of the communication bus.

B) Collision Detection and Resolution:

- Implement collision detection algorithms or protocols like Carrier Sense Multiple Access with Collision Detection (CSMA/CD).

C) Synchronization Solutions:

- Implement clock synchronization protocols for accurate timing.
- Use synchronization signals or preamble sequences in the communication protocol.

D) Error Detection and Handling:

- Use robust error detection techniques like cyclic redundancy checks (CRCs) or checksums.
- Implement error correction protocols or retransmission mechanisms.

E) Grounding and Electrical Noise Mitigation:

- Ensure proper grounding practices to minimize ground-potential differences and reduce electrical noise.

By implementing these solutions and best practices, developers can mitigate the challenges associated with half-duplex communication using RS-485 modules and ensure reliable and robust communication in various applications.

10. Conclusion:

10.1. Summary of the project:

The project utilizes RS485, a popular communication standard for industrial applications, to enable half-duplex communication between multiple devices. Half-duplex communication means that data can be transmitted and received, but not simultaneously.

Key Components:

A) RS485 Module: facilitating serial communication over long distances with noise immunity.

B) Communication Protocol: Establishes rules for data exchange, ensuring reliable and synchronized communication between devices.

C) Wiring Infrastructure: Proper wiring setup is crucial for RS485 communication, including twisted pair cables for data lines and termination resistors to minimize signal reflections**.**

Functionality:

A) Data Transmission: The microcontroller sends data packets over the RS485 bus to other connected devices.

B) Data Reception: It listens for incoming data packets and processes them accordingly.

C) Synchronization: Devices on the RS485 bus adhere to a predefined communication protocol to ensure synchronized data exchange.

D) Error Handling: Mechanisms are in place to detect and mitigate errors, ensuring data integrity during transmission.

Overall, the project harnesses the robustness and efficiency of RS485 for half-duplex communication, making it suitable for various applications requiring reliable data exchange over long distances**.**

10.2. Achievements and limitations:

Achievements:

A) Reliable Communication: By utilizing the RS485 module, the project achieves reliable communication over long distances, even in noisy industrial environments.

B) Cost-Effectiveness: RS485 modules are relatively inexpensive and readily available, making the project cost-effective for implementing communication between multiple devices.

C) Scalability: The project can easily scale to accommodate additional devices on the RS485 bus, allowing for flexible expansion in industrial or other applications.

D) Customizability: The microcontroller-based approach allows for customization of the communication protocol and functionality to suit specific project req **Limitations:**

A) Half-Duplex Operation: While half-duplex communication is sufficient for many applications, it limits the speed of data transmission compared to full-duplex communication.

B) Wiring Complexity: Proper wiring and termination are crucial for RS485 communication, which can add complexity to the installation process, especially over long distances.

C) Synchronization Challenges: Ensuring synchronized communication between devices, especially in large networks, can be challenging and may require careful protocol design.

D) Limited Data Rate: RS485 has a maximum data rate, which may not be suitable for applications requiring very high-speed communication.

10.3. Future enhancements and recommendations:

Future enhancements in half-duplex communication using RS485 modules, consider:

A) Improved Error Detection and Correction: Implement robust error detection and correction mechanisms to ensure data integrity, such as CRC checks or checksums.

B) Enhanced Noise Immunity: Explore techniques like adaptive equalization or noise filtering to improve communication reliability, especially in noisy environments.

C) Higher Data Rates: Investigate ways to increase data rates without sacrificing reliability, perhaps through better modulation techniques or optimized transmission protocols.

D) Power Efficiency: Develop methods to minimize power consumption during communication, particularly for battery-operated devices or low-power applications.

11. References:

S. H. Alnabelsi, H. B. Salameh and K. A. Darabkh, "A Comparative Study for Half-duplex and Full-duplex Multi-hop Routing in Software Defined Networks," 2022 Ninth International Conference on Software Defined Systems (SDS), Paris, France, 2022, pp. 1-5, doi: 10.1109/SDS57574.2022.10062917. keywords: {Statistical analysis;Wireless networks;Half-duplex system;Spread spectrum communication;Full-duplex system;Channel allocation;Throughput;Half-duplex;Full-duplex;Routing;Throughput;Cognitive Radio},

12. Appendices:

12.1. Code Snippets:

```
//       Teacher Code
const int ledPin = 13;  // Built-in LED
const int EnTxPin = 2;  // HIGH:Transmitter, LOW:Receiver
void setup()
{
  Serial.begin(9600);
  Serial.setTimeout(100);
  pinMode(ledPin, OUTPUT);
  pinMode(EnTxPin, OUTPUT);
  digitalWrite(ledPin, LOW);
```

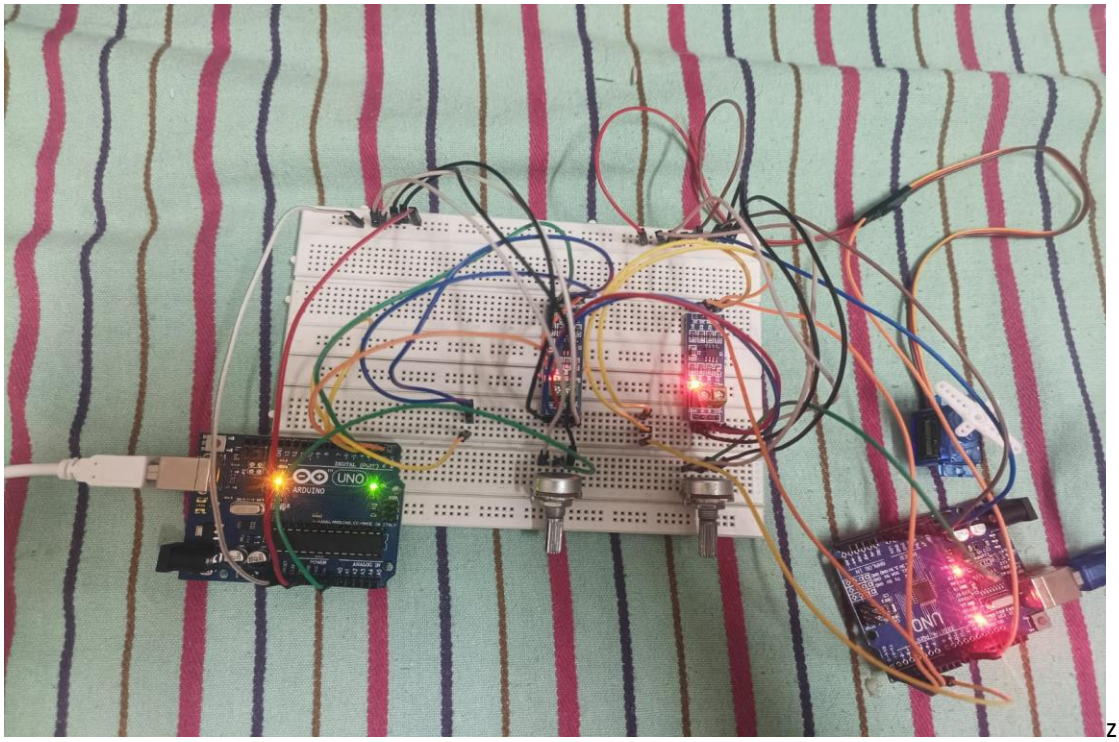
```
  digitalWrite(EnTxPin, HIGH);
}
void loop()
{
  int rdata = analogRead(0); //data from potentiometer
  int angle= map(rdata, 0, 1023, 0, 180);
  //transmitter data packet
  Serial.print("I"); //initiate data packet
  Serial.print("S"); //code for servo
  Serial.print(angle); //servo angle data
  Serial.print("F"); //finish data packet
  delay(50);
  //receiver data packet
  Serial.print("I"); //initiate data packet
  Serial.print("L"); //code for sensor
  Serial.print("F"); //finish data packet
  Serial.flush();
  digitalWrite(EnTxPin, LOW); //RS485 as receiver
  if(Serial. find("i"))
  {
      int data=Serial.parseInt();
      if(Serial.read()=='f') onLED(data);
  }
```

```
  digitalWrite(EnTxPin, HIGH); //RS485 as transmitter
}
void onLED(int data)
{
  if(data>500) digitalWrite(ledPin, HIGH);
  else digitalWrite(ledPin, LOW);
}

// Slave Code
#include <Servo.h>
Servo myservo;
const int EnTxPin = 2;
void setup () {
  Serial.begin (9600);
  myservo.attach (9);
  pinMode(EnTxPin, OUTPUT );
  digitalWrite (EnTxPin, LOW );
}
void loop (){
  if ( Serial.available ()){
    if ( Serial.read () == 'I' ){
      char function = Serial.read ();
      if (function == 'S' ){
```

```
        int angle = Serial.parseInt ();
        if ( Serial.read () == 'F' ){
          if (angle <= 180) myservo.write (angle);
         }
      }
      else if (function == 'L' ){
        if ( Serial.read () == 'F' ){
          int val = analogRead (0);
          digitalWrite (EnTxPin, HIGH ); //enable to transmit
          Serial.print ( "i" );
          Serial.print (val);
          Serial.println ( "f" );
          Serial.flush ();
          digitalWrite (EnTxPin, LOW ); //enable to receive
        }
      }
    }
  }
  delay (10);
}
```

12.2. Photographs and Diagram:



z