

# HANDSHAKE COMMUNICATION BETWEEN MASTER AND CLIENT

## 1. PROJECT PURPOSE:

The purpose of this project is to exchange data from master to slave over serial communication (Ex: uart,spi).

## 2. FUNCTIONALITIES OF THE SYSTEM:

These functionalities have been implemented in the program:

- Data exchanging between master to slave over serial communication.
- Support for asynchronous and synchronous communication modes.
- Error detection and recovery mechanisms.
- Data integrity verification through checksums or other Techniques.

### Compiling the code:

- On slave's Terminal create an user space application that should invoke the respective driver.
- To run the code, First on the slave's Terminal: `./a.out driver_name`.
- Compile the Driver and insert the module ex: `sudo insmod driver_name`.
- Enter the choice either to send the data or receive etc.
- Closing The Communication remove the module ex: `sudo rmmod driver_name`.

## 3. OPERATING ENVIRONMENT:

Operating environment for Transferring data from master and client using Serial communication are:

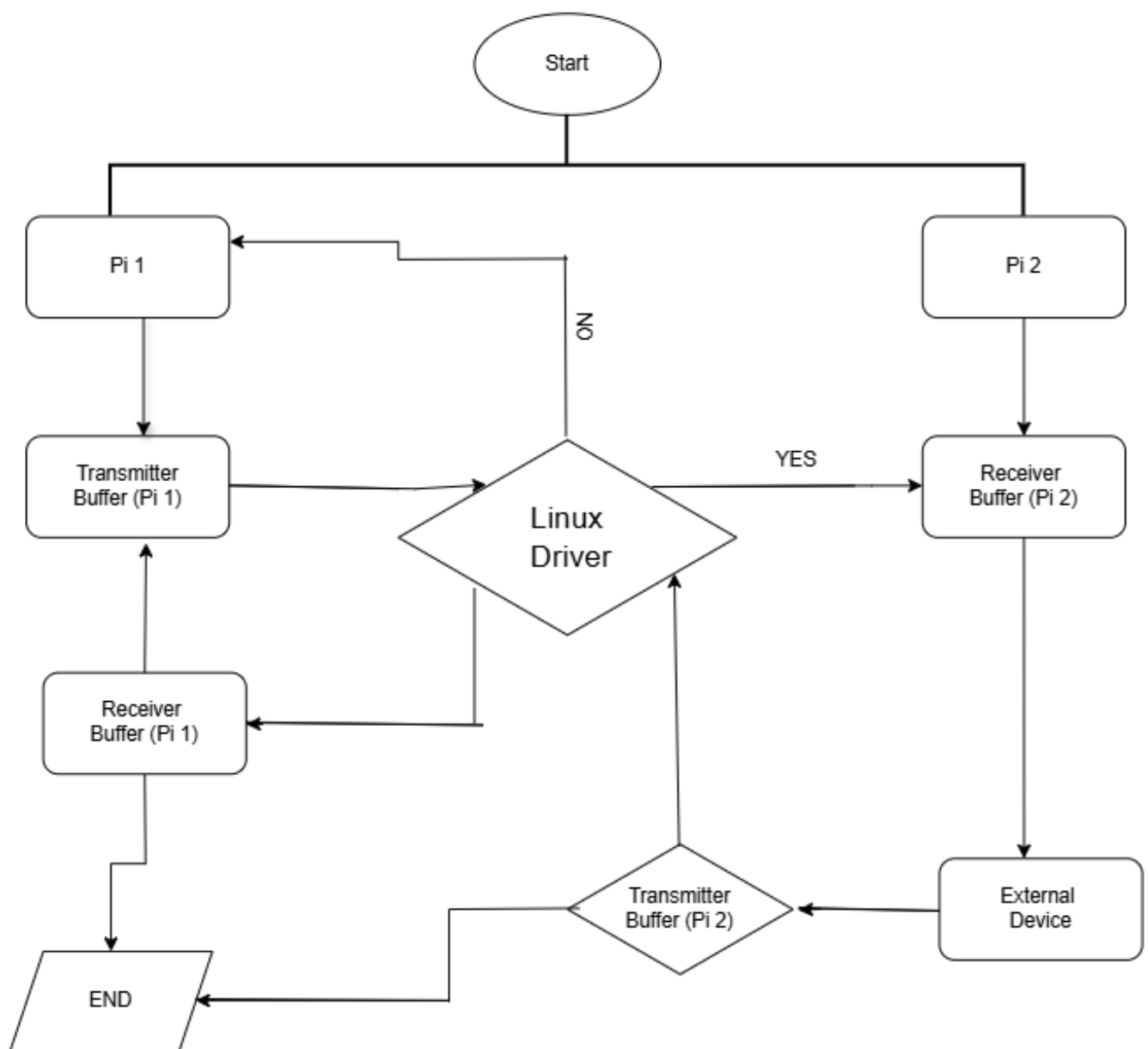
- Master and Client system
- Operating system: Linux
- Platform: Ubuntu/C

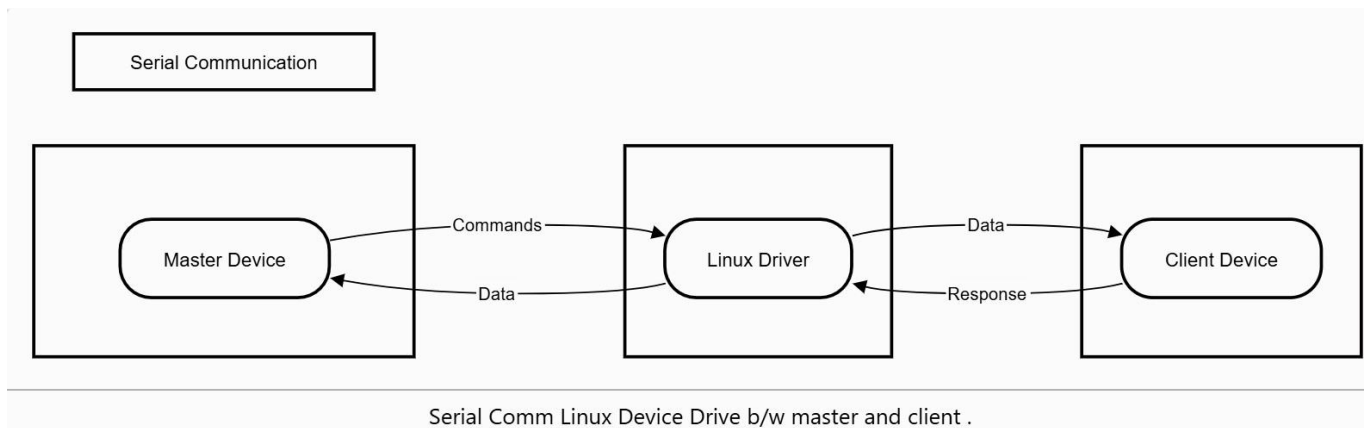
## 4. TESTING PLAN:

- **Test Strategy:** Develop unit tests, integration tests and systems tests for validating the data receiving across the serial communication.
- **Unit Tests:** Test individual handler functionality (Ex: **Spi,Uart**).
- **Integration Tests:** Test master-slave communication and data transfer, ensuring synchronization, error handling and throughput.

- **System Testing** : Test data transfer across the full system (master to client) under various conditions (Ex: different baud rates , data sizes).
- **Test cases:**
  - 1.verify successful data receive from master to client.
  - 2.simulate errors (Ex: buffer overflows ,lost data) and test error handling.
  - 3.performance testing under different loads.

## 5.FLOW DIAGRAM





## 6.BLOCK DIAGRAM

### 1. Master Device:

The master device initiates communication by sending the commands. These commands are instructions or request meant to interact with the client device.

### 2. Driver Receives commands:

The commands from the master device are passed to the Linux driver. The driver acts as an intermediary layer that processes and translates these commands for the client device.

### 3. Driver Sends to Client:

After processing the commands the Linux driver sends the appropriate data or instruction to the client device via serial communication channel.

### 4. Client Device Processes:

The Client device receives and interprets the commands from the Linux driver based on the commands, the client device performs specific actions or computations.

### 5. Client Sends Response:

Once the client device completes the required action. It generates a response this could be a status message, output data or any other result of the operation.

### 6. Driver Receives Response:

The Linux Driver collects the response from the client device and prepares it for transmission back to the master device.

### 7. Master Responds to Driver:

The response is send back to the master device through the Linux driver this completes the communication cycle.

## **5. CONCLUSION:**

The project successfully developed and tested a Linux device driver capable of facilitating communication between a master and client over serial communication (Ex: spi,uart)The driver ensures reliable data transfer, robust error handling and correct protocol compliance through rigorous testing and modular design.