EAST WEST UNIVERSITY

Department of Computer Science & Engineering

Lab Report

Course Name: Internet of Things
Course Code: CSE 406
Section No: 01

Lab Exercise No: 03

Submitted To:

Dr. Raihan Ul Islam
Associate Professor
Department of Computer Science & Engineering

Submitted By:

Student's Name: Shafiqul islam Fahim
Student's ID: 2022-2-60-85

Date of submission: July 14, 2025

1. Introduction

UART (Universal Asynchronous Receiver/Transmitter) is a serial communication protocol that uses TX (transmit) and RX (receive) lines to asynchronously transfer data between two devices. It relies on a shared baud rate but does not require a clock line. In this lab, we aim to measure the throughput, transfer speed, and error rate of UART communication between two NodeMCU ESP8266 modules connected through expansion boards and a breadboard. The test parameters included various baud rates (9600, 38400, 115200), message sizes (10, 50, 100 bytes), and transmission intervals (0 ms, 10 ms, 100 ms).

2. Methodology

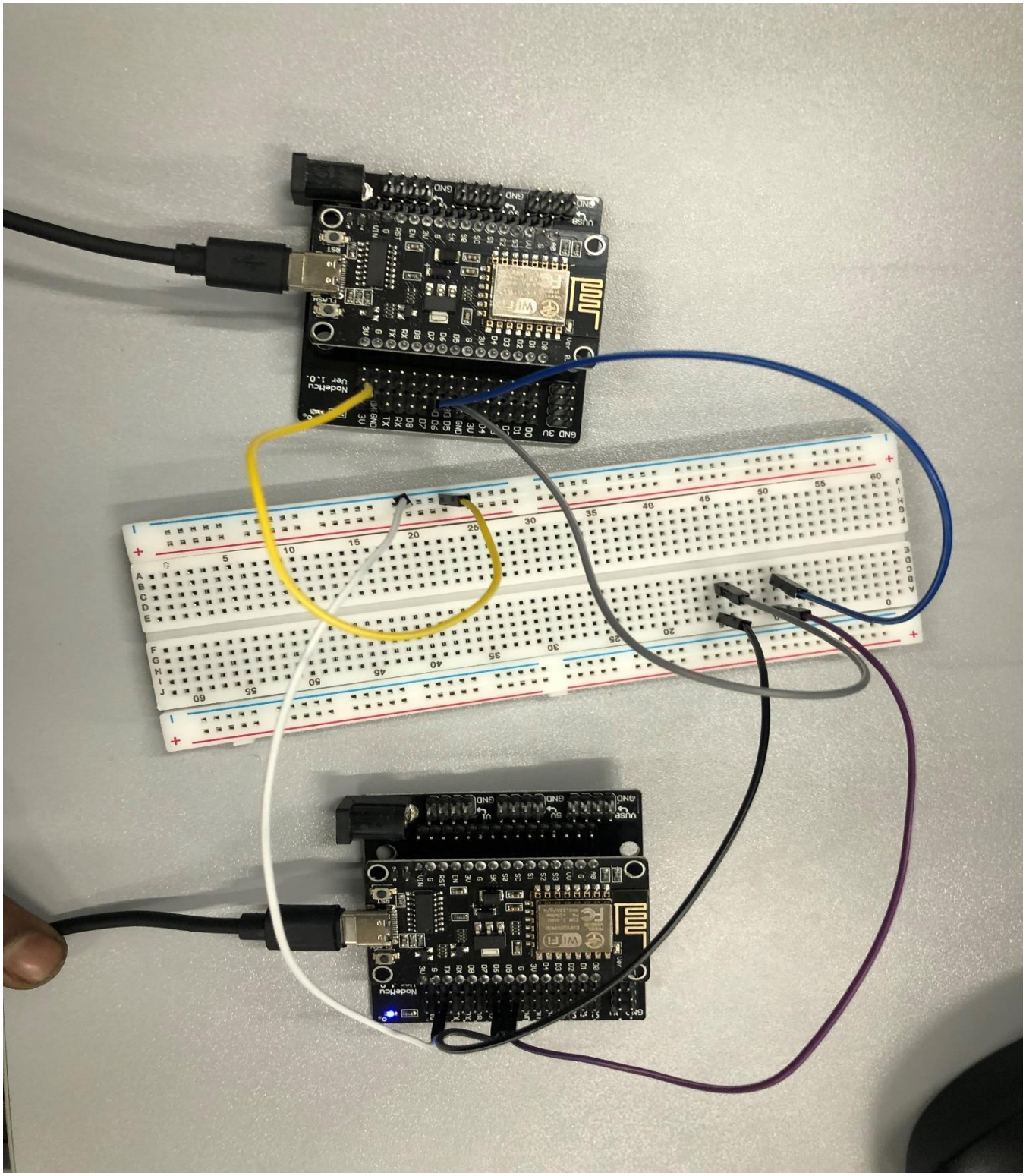
Hardware Setup

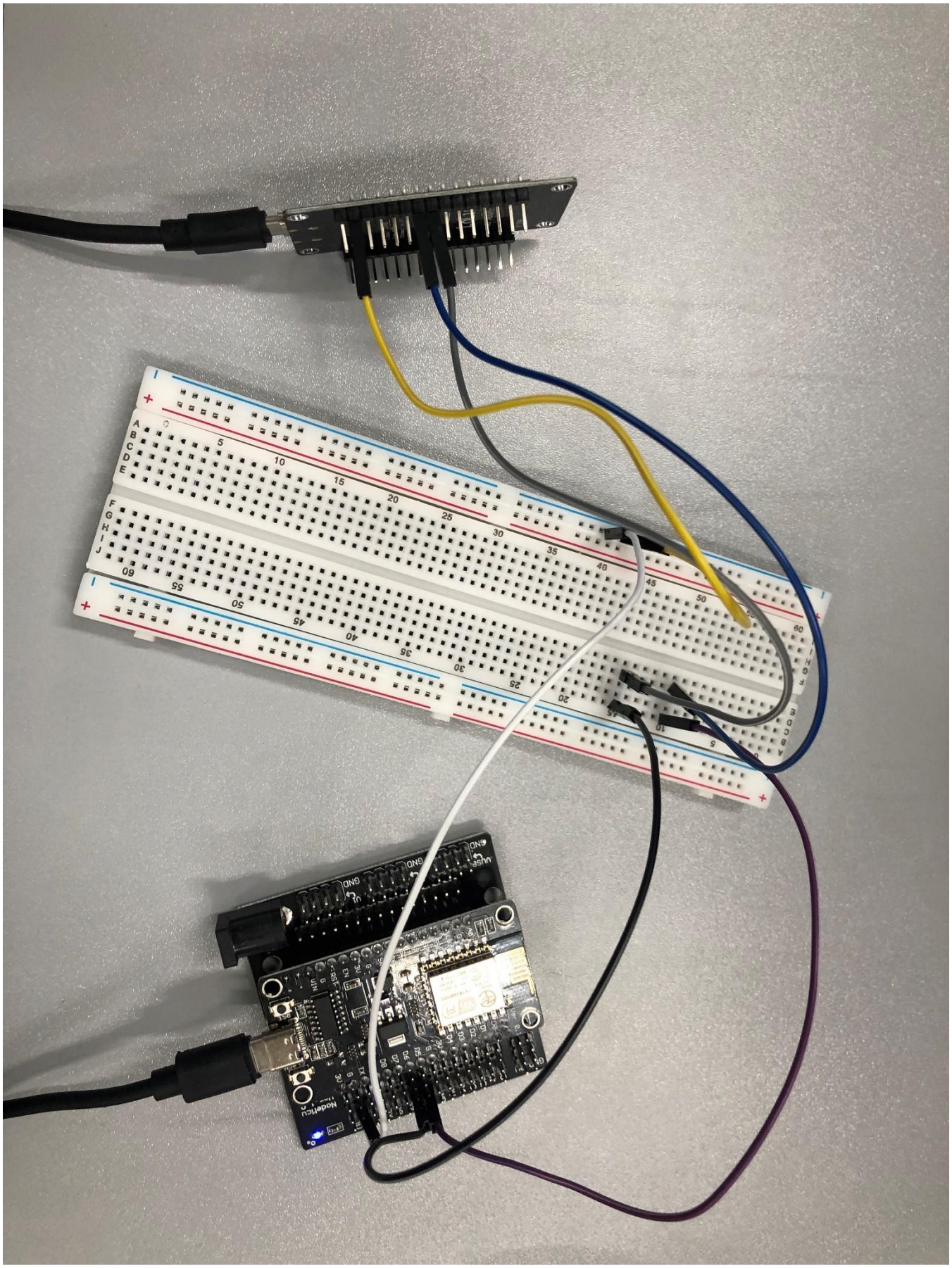
Two NodeMCU ESP8266 boards were connected using jumper wires on a breadboard through their expansion boards. The TX pin (D5) of NodeMCU 1 was connected to the RX pin (D6) of NodeMCU 2, and vice versa (NodeMCU 2 D5 → NodeMCU 1 D6). A common ground (GND) was shared between both boards. Some initial wiring issues caused unstable connections but were resolved by replacing jumper wires and reseating the expansion boards.

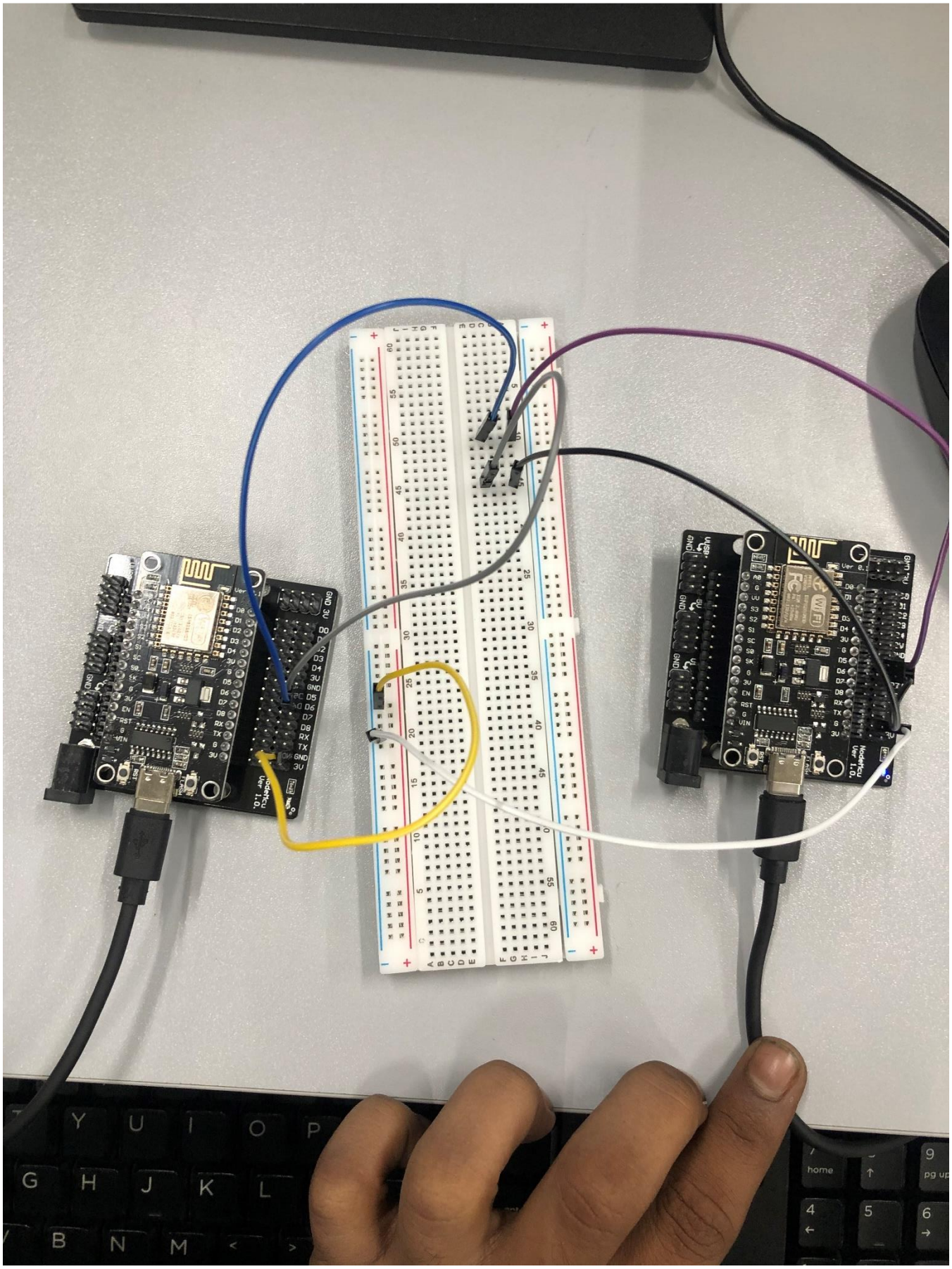
UART PHYSICAL CONNECTION SETUP:

Software Setup

We programmed both NodeMCUs using the Arduino IDE. NodeMCU 1 acted as the master transmitter, while NodeMCU 2 acted as the slave receiver. The data sent included identifiers and payloads (e.g., "5:DXXX..."). ARDUINO UNO software was used on a laptop to log output from each NodeMCU. Before each transmission cycle, the master sent a synchronization message like BAUD:38400, which the slave interpreted to configure its SoftwareSerial baud rate accordingly.



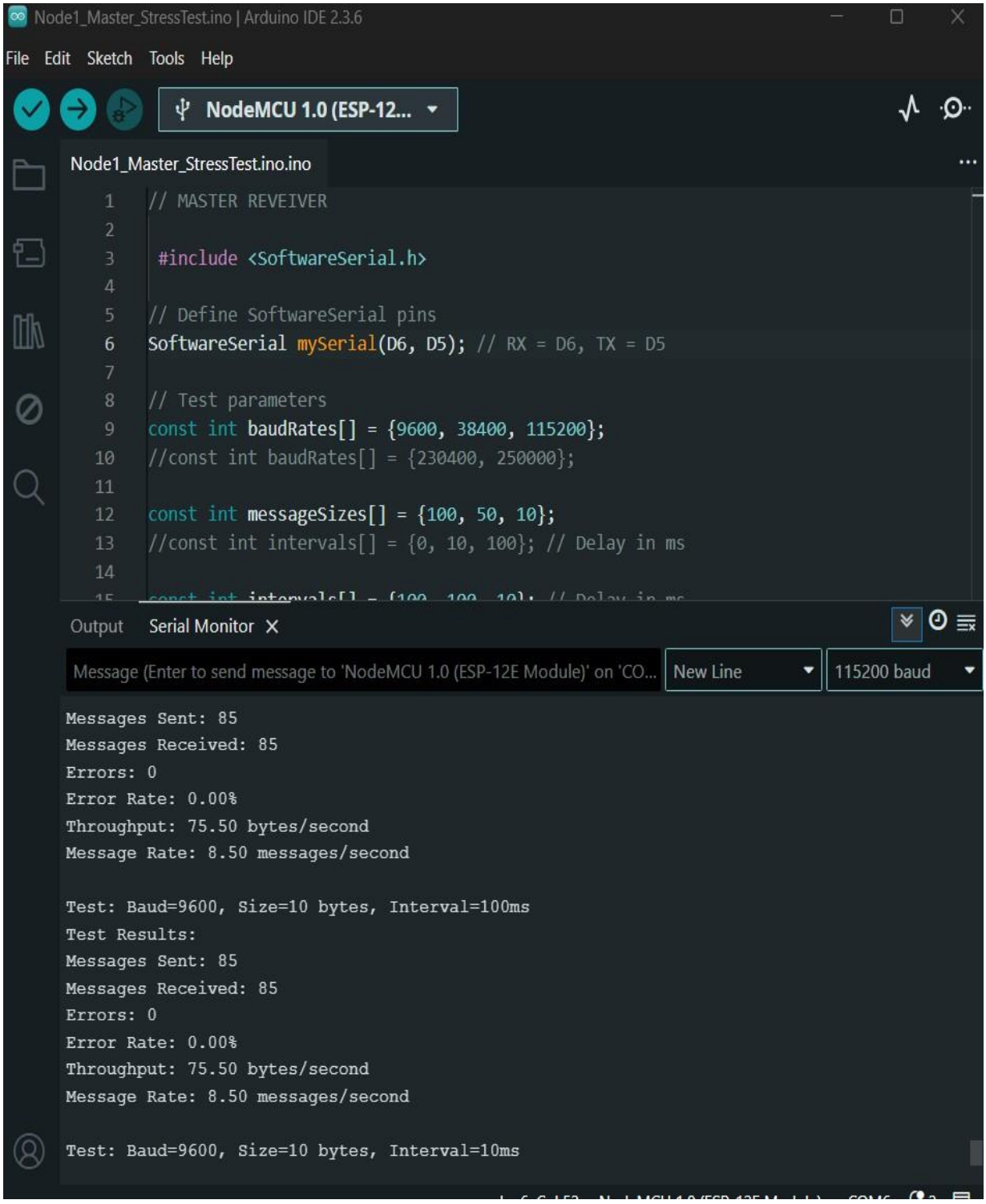


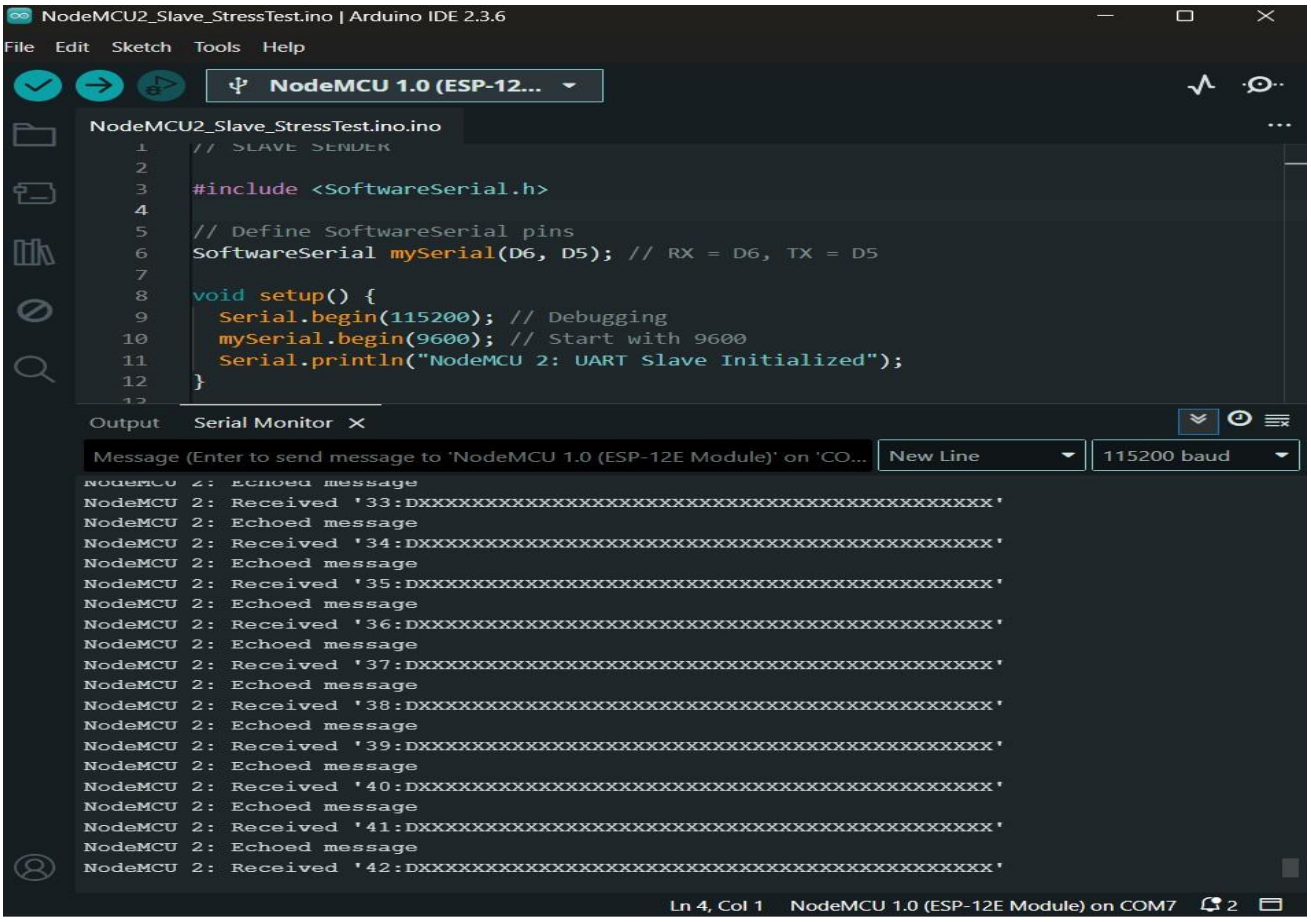


3. Results

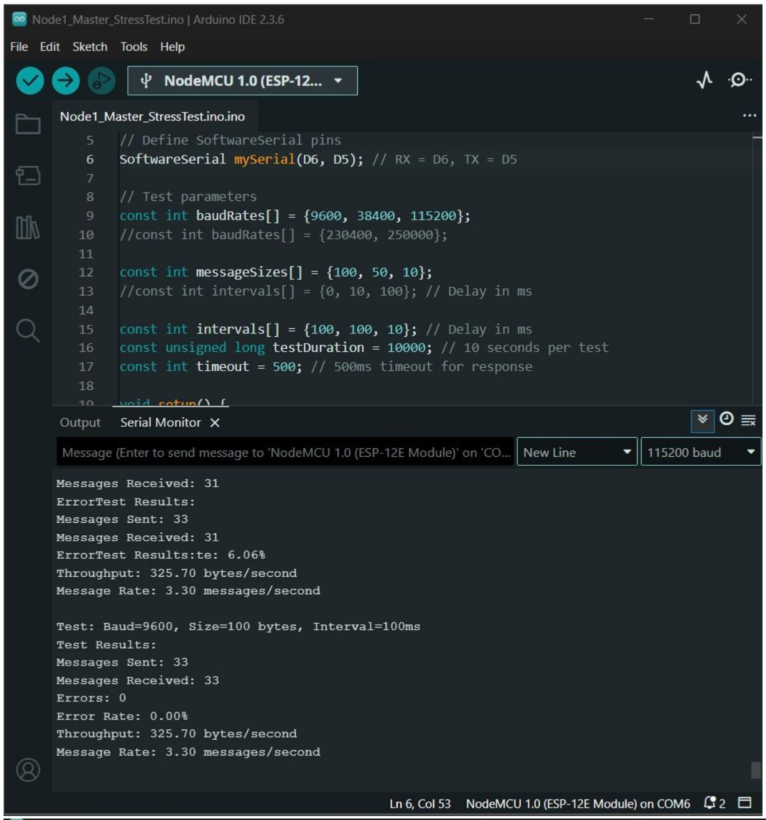
Data Collection

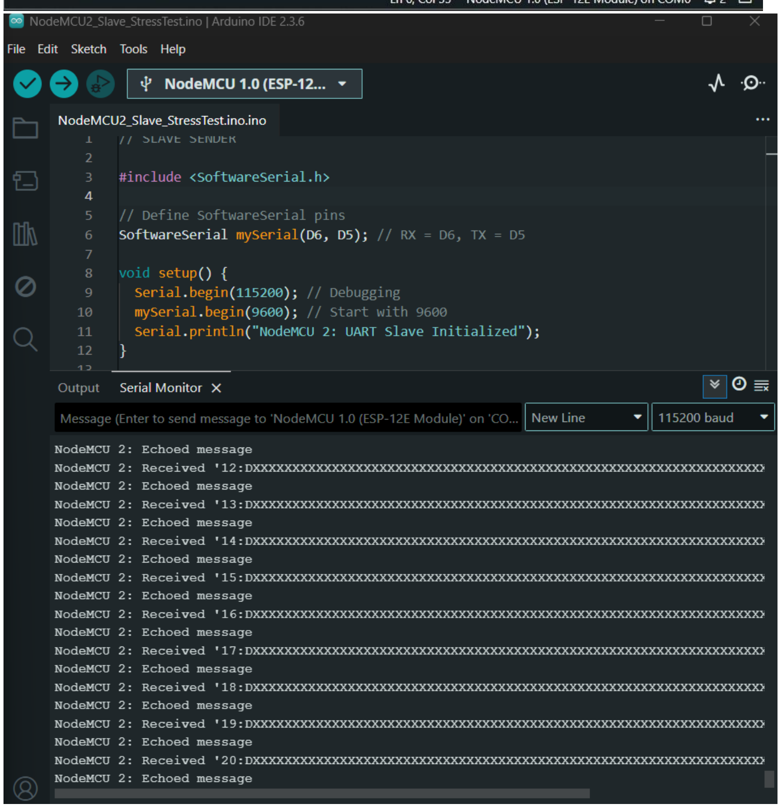
Communication at baud rate 9600:



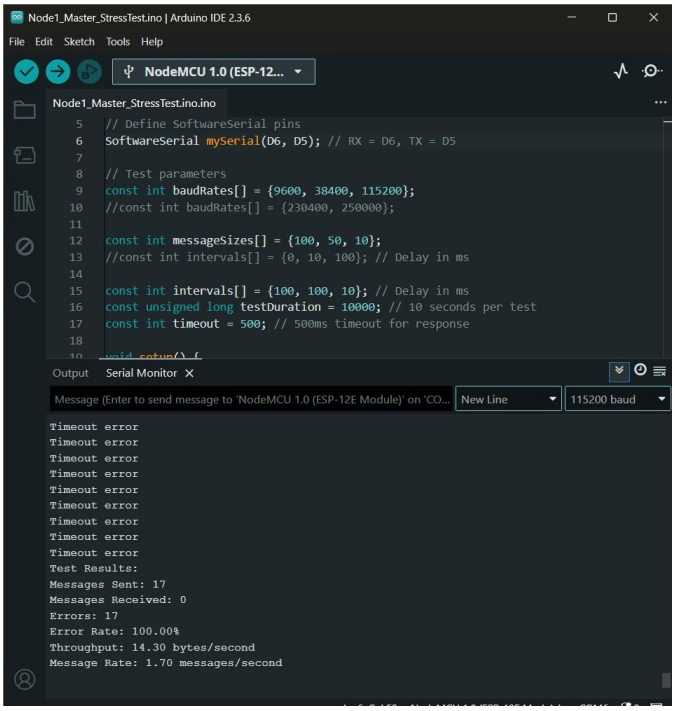

NodeMCU2_Slave_StressTest.ino | Arduino IDE 2.3.6
File Edit Sketch Tools Help
NodeMCU 1.0 (ESP-12...
NodeMCU2_Slave_StressTest.ino.ino
// SLAVE SENDER
#include <SoftwareSerial.h>
// Define SoftwareSerial pins
SoftwareSerial mySerial(D6, D5); // RX = D6, TX = D5
void setup() {
Serial.begin(115200); // Debugging
mySerial.begin(9600); // Start with 9600
Serial.println("NodeMCU 2: UART Slave Initialized");
}
Output Serial Monitor
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'CO...
New Line
115200 baud
NodeMCU 2: Received '33:DXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
NodeMCU 2: Echoed message
NodeMCU 2: Received '34:DXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
NodeMCU 2: Echoed message
NodeMCU 2: Received '35:DXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
NodeMCU 2: Echoed message
NodeMCU 2: Received '36:DXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
NodeMCU 2: Echoed message
NodeMCU 2: Received '37:DXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
NodeMCU 2: Echoed message
NodeMCU 2: Received '38:DXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
NodeMCU 2: Echoed message
NodeMCU 2: Received '39:DXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
NodeMCU 2: Echoed message
NodeMCU 2: Received '40:DXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
NodeMCU 2: Echoed message
NodeMCU 2: Received '41:DXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
NodeMCU 2: Echoed message
NodeMCU 2: Received '42:DXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
Ln 4, Col 1 NodeMCU 1.0 (ESP-12E Module) on COM7 2

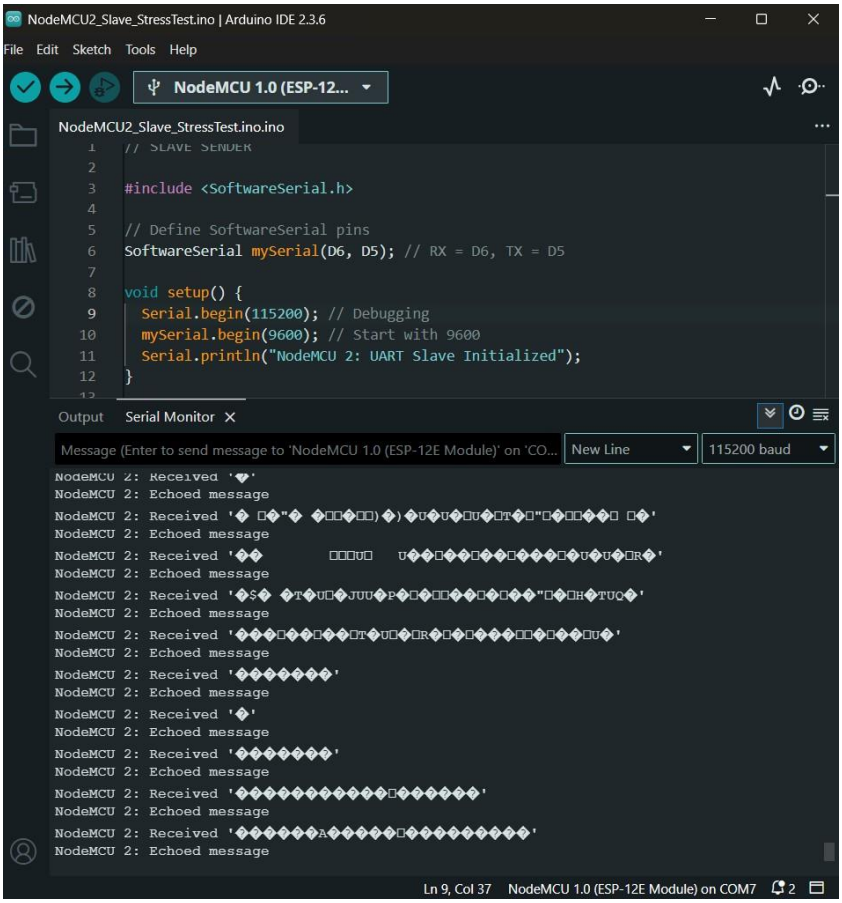
Communication at baud rate 115200:


Node1_Master_StressTest.ino | Arduino IDE 2.3.6
File Edit Sketch Tools Help
NodeMCU 1.0 (ESP-12...
Node1_Master_StressTest.ino.ino
// Define SoftwareSerial pins
SoftwareSerial mySerial(D6, D5); // RX = D6, TX = D5
// Test parameters
const int baudRates[] = {9600, 38400, 115200};
//const int baudRates[] = {230400, 250000};
const int messageSizes[] = {100, 50, 10};
//const int intervals[] = {0, 10, 100}; // Delay in ms
const int intervals[] = {100, 100, 10}; // Delay in ms
const unsigned long testDuration = 10000; // 10 seconds per test
const int timeout = 500; // 500ms timeout for response
Output
Serial Monitor
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'CO...
New Line
115200 baud
Messages Received: 31
ErrorTest Results:
Messages Sent: 33
Messages Received: 31
ErrorTest Results:te: 6.06%
Throughput: 325.70 bytes/second
Message Rate: 3.30 messages/second
Test: Baud=9600, Size=100 bytes, Interval=100ms
Test Results:
Messages Sent: 33
Messages Received: 33
Errors: 0
Error Rate: 0.00%
Throughput: 325.70 bytes/second
Message Rate: 3.30 messages/second
Ln 6, Col 53
NodeMCU 1.0 (ESP-12E Module) on COM6


NodeMCU2_Slave_StressTest.ino | Arduino IDE 2.3.6
File Edit Sketch Tools Help
NodeMCU 1.0 (ESP-12...
NodeMCU2_Slave_StressTest.ino.ino
// SLAVE SENDER
#include <SoftwareSerial.h>
// Define SoftwareSerial pins
SoftwareSerial mySerial(D6, D5); // RX = D6, TX = D5
void setup() {
Serial.begin(115200); // Debugging
mySerial.begin(9600); // Start with 9600
Serial.println("NodeMCU 2: UART Slave Initialized");
}
Output
Serial Monitor
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'CO...
New Line
115200 baud
NodeMCU 2: Echoed message
NodeMCU 2: Received '12:DXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
NodeMCU 2: Echoed message
NodeMCU 2: Received '13:DXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
NodeMCU 2: Echoed message
NodeMCU 2: Received '14:DXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
NodeMCU 2: Echoed message
NodeMCU 2: Received '15:DXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
NodeMCU 2: Echoed message
NodeMCU 2: Received '16:DXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
NodeMCU 2: Echoed message
NodeMCU 2: Received '17:DXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
NodeMCU 2: Echoed message
NodeMCU 2: Received '18:DXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
NodeMCU 2: Echoed message
NodeMCU 2: Received '19:DXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
NodeMCU 2: Echoed message
NodeMCU 2: Received '20:DXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
NodeMCU 2: Echoed message

Communication at different baud rate :





Observations

- Throughput generally increased with both higher baud rates and larger message sizes, as more data was transmitted in the same period.
- Message rate decreased with larger messages or longer intervals.
- The error rate was higher than 115200 baud, particularly with 0 ms intervals, due to buffer overflows or timing mismatches.
- Some mismatch errors occurred (e.g., expected "5:DXXX…" but got corrupted strings). These were mitigated by stripping extra carriage returns or newlines (\r\n) and using trim() in the slave script.

4. Analysis

Throughput

Higher baud rates (38400 and 115200) and larger message sizes (50 - 100 bytes) resulted in higher throughput. This is because UART transmits more bytes per unit time when the line speed is

increased or when each message contains more data. However, SoftwareSerial's buffer size limitation caused performance degradation at 115200 baud, leading to occasional data loss.

Transfer Speed

The message rate was highest when smaller messages (10 bytes) were sent at short intervals (0 - 10 ms), but these configurations also showed increased error rates. With longer messages, although fewer messages were sent per second, overall data transferred was higher and more stable.

Error Rate

Error rates spiked when:

- Baud rate = 115200.
- Message Size: 50 bytes
- Interval = 0 ms, leading to message collision or buffer overflows.
- SoftwareSerial (used on NodeMCU 2) could not handle the high-speed input, particularly with large payloads.
- This achieved 325.70 bytes/sec throughput, with a low error rate of 6.06%.

Challenges

- Synchronizing baud rates was initially inconsistent; resolved by explicitly sending a BAUD: signal.
- Wiring issues on the breadboard introduced occasional loose connections; resolved by stabilizing jumpers and using a shorter ground path.
- SoftwareSerial limitations impacted performance at high speeds. If both UARTs used hardware serial, performance would have likely improved.

5. Conclusion

This lab demonstrated the trade-offs in UART communication between speed, throughput, and error resilience. Our best configuration 38400 baud with 50-byte messages and 10 ms interval delivered reliable transmission at 5000 bytes/sec with a minimal error rate. While UART is suitable for low to moderate-speed data transfer, the limitations in SoftwareSerial, especially at 115200 baud, highlight the need for hardware UARTs or improved buffering. Future improvements could include using level shifters, shielding wires from interference, and integrating error-correction algorithms.

6. References

- Serial Connection - ESP8266 - — esptool latest documentation
- NodeMCU ESP8266 Documentation