CE406: Internet of Things

Lab 03: UART Communication on ESP32 & ESP8266

with a Comparative Performance Study

Instructor October 27, 2025

Objectives

- Explain UART fundamentals: TX/RX, baud, frame format (8N1), and cross-wiring with common GND.
- Implement board-to-board UART on **ESP32** (HardwareSerial with custom pins) and on **ESP8266** (SoftwareSerial on D5/D6).
- Collect performance data (throughput, message rate, error rate) across parameter sweeps.
- Compare ESP32 vs. ESP8266 UART behavior and write a concise, data-driven report.

1. Background

Both ESP32 and ESP8266 support UART. ESP32 provides three hardware UARTs (UART0/1/2) and allows flexible pin mapping via the GPIO matrix; we will keep UART0 for USB logging and use UART1/2 for inter-board links on custom pins. ESP8266 exposes a hardware UART shared with USB; for board-to-board we will use `SoftwareSerial` on general-purpose pins so the USB port remains free. See References for details and the original ESP32 lab handout used as the structural template.

2. Hardware & Software

Common: two boards (one pair ESP32 DevKit V1, one pair NodeMCU ESP8266), female–female jumpers (≤ 15 cm), two USB cables, Arduino IDE.
Install ESP32 boards support (Board Manager: *esp32 by Espressif Systems*) and ESP8266 core (Board Manager: *ESP8266 by ESP8266 Community*).

3. Part A — ESP32 Wiring & Test

Use custom-mapped UART pins and cross-wire TX→RX, RX→TX; join GNDs.

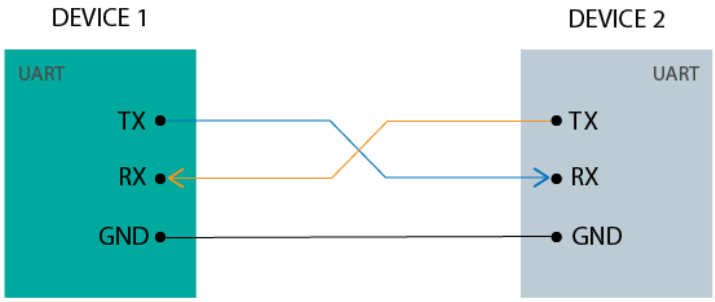

Figure 1: Concept: UART cross-over and common GND (ESP32 example).

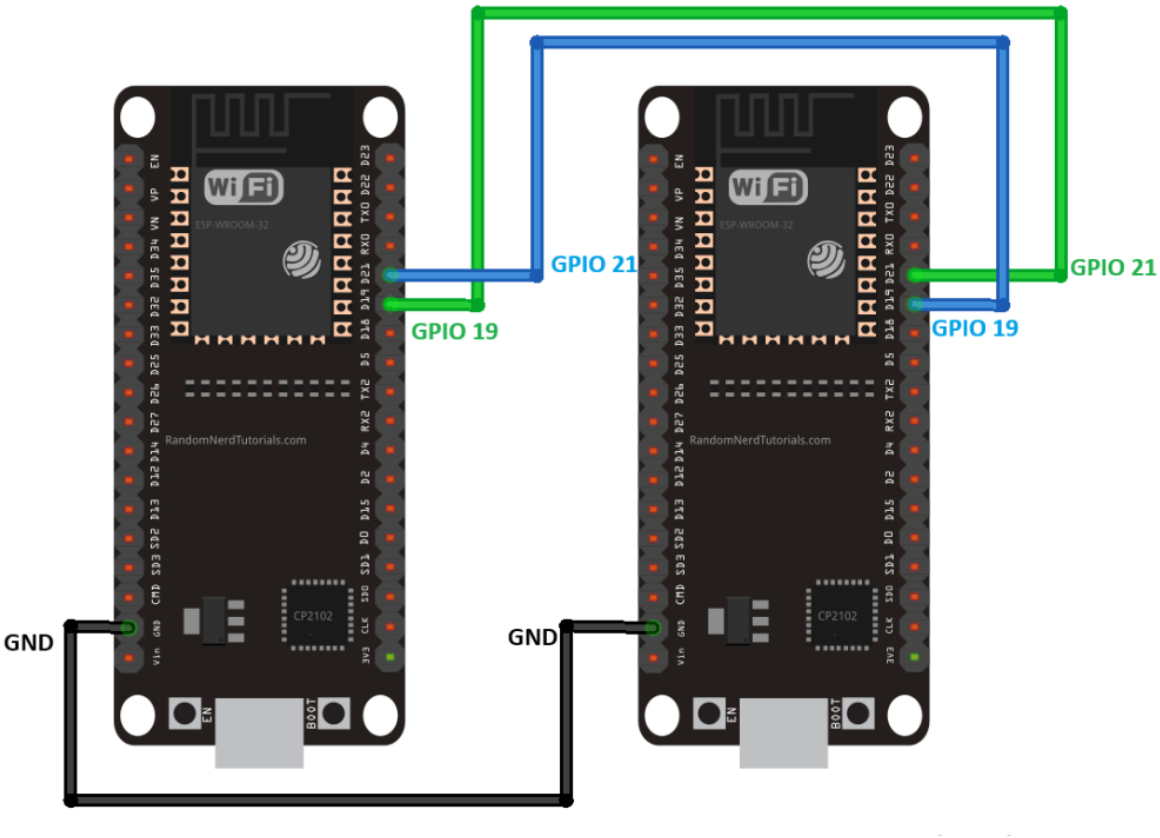

Figure 2: ESP32-to-ESP32 direct jumpers (example with GPIO19 as TX and GPIO21 as RX).

Sample Codes (ESP32)

Receiver (UART2).

```
/*********
  Rui Santos & Sara Santos - Random Nerd Tutorials
  Complete instructions at https://RandomNerdTutorials.com/esp32-uart-
    communication-serial-arduino/
  Permission notice retained.
*********/
#define TXD1 19
#define RXD1 21
```

```
HardwareSerial mySerial(2);  // UART2

void setup() {
  Serial.begin(115200);
  mySerial.begin(9600, SERIAL_8N1, RXD1, TXD1);
  Serial.println("ESP32 UART Receiver");
}

void loop() {
  if (mySerial.available()) {
    String message = mySerial.readStringUntil('\n');
    Serial.println("Received: " + message);
  }
}
```

Sender (UART1).

```
/*********
  Rui Santos & Sara Santos - Random Nerd Tutorials
  Complete instructions at https://RandomNerdTutorials.com/esp32-uart-
    communication-serial-arduino/
  Permission notice retained.
*********/
#define TXD1 19
#define RXD1 21
HardwareSerial mySerial(1);  // UART1

int counter = 0;

void setup() {
  Serial.begin(115200);
  mySerial.begin(9600, SERIAL_8N1, RXD1, TXD1);
  Serial.println("ESP32 UART Transmitter");
}

void loop() {
  mySerial.println(String(counter));
  Serial.println("Sent: " + String(counter));
  counter++;
  delay(1000);
}
```

4. Part B — ESP8266 Wiring & Test

Use `SoftwareSerial` on D5 (TX=GPIO14) and D6 (RX=GPIO12). Cross-wire and share GND.

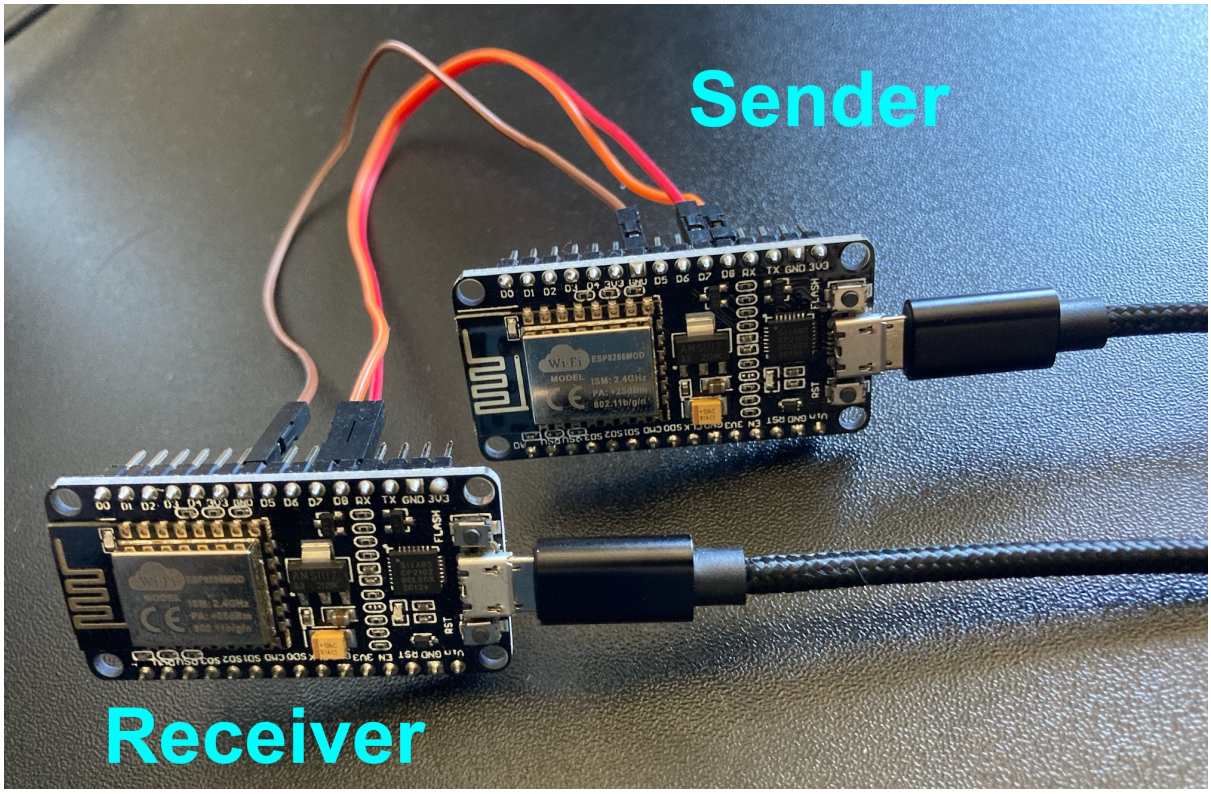

Figure 3: Real wiring photo: NodeMCU *Sender* (D5=TX) to *Receiver* (D6=RX) with common GND.

Sample Codes (ESP8266)

NodeMCU 1 — Master.

```
#include <SoftwareSerial.h>

// Define SoftwareSerial pins
SoftwareSerial mySerial(D6, D5); // RX = D6, TX = D5

void setup() {
  Serial.begin(115200); // Debugging
  mySerial.begin(9600); // UART
  Serial.println("NodeMCU 1: UART Master");
}

void loop() {
  mySerial.println("Hello from NodeMCU 1");
  Serial.println("Sent: Hello from NodeMCU 1");

  if (mySerial.available()) {
    String receivedData = mySerial.readStringUntil('\n');
    Serial.print("Received: ");
    Serial.println(receivedData);
  }

  delay(2000); // Wait 2 seconds
}
```

NodeMCU 2 — Slave.

```
#include <SoftwareSerial.h>

// Define SoftwareSerial pins
SoftwareSerial mySerial(D6, D5); // RX = D6, TX = D5

void setup() {
  Serial.begin(115200); // Debugging
  mySerial.begin(9600); // UART
  Serial.println("NodeMCU 2: UART Slave");
}

void loop() {
  if (mySerial.available()) {
    String receivedData = mySerial.readStringUntil('\n');
    Serial.print("Received: ");
    Serial.println(receivedData);

    mySerial.println("Hi from NodeMCU 2");
    Serial.println("Sent: Hi from NodeMCU 2");
  }

  delay(100); // Small delay
}
```

Notes. `SoftwareSerial` on ESP8266 is CPU-driven; start at 9600 or 38400 for reliability. Test 115200 only with short, clean wiring.

5. Part C — Comparative Stress Test

5.1. Parameter Grid

Run the following combinations on each platform (ESP32 pair and ESP8266 pair):

- Baud: 9600, 38400, 115200
- Message size: 10, 50, 100 bytes (pad with ASCII)
- Interval: 0 ms, 10 ms, 100 ms

5.2. Metrics & Formulas

For a test window of duration T seconds:

- **Throughput (B/s)** = total payload bytes received$T$
- **Message rate (msg/s)** = messages received$T$
- **Error rate (%)** = $100 \times$ sent $-$ valid receivedsent

5.3. Results Templates

Table 1: ESP32 results

Baud	Size	Interval	Throughput (B/s)	Msg/s	Error (%)

Table 2: ESP8266 results

Baud	Size	Interval	Throughput (B/s)	Msg/s	Error (%)

Table 3: Head-to-head comparison (ESP32 vs ESP8266)

Baud	Size	Interval	ESP32 Throughput	ESP8266 Throughput	Winner

5.4. Guiding Questions

- At the same baud/size/interval, which platform delivers higher throughput and lower error?
- Does ESP8266 `SoftwareSerial` saturate earlier (e.g., at 115 200 baud) than ESP32 HardwareSerial?
- What is the "sweet spot" for reliable messages with moderate speed on each board?

6. Submission

- **Code:** upload four sketches (`ESP32 Sender/Receiver`, `ESP8266 Master/Slave`).
- **Data:** raw logs or CSV; filled tables above.
- **Report (4–6 pages):** brief methods, results (tables/plots), and a **comparison discussion** with your recommended configuration for each platform.

7. Marking (summary)

Wiring correctness (cross-over + GND), clean code and logging, correct metric calculations, completeness of the parameter sweep, clarity of the comparative analysis, and reproducibility (pin choices documented).

8. References

- Random Nerd Tutorials: *ESP32 UART — Serial Communication, Send and Receive Data (Arduino IDE)*. https://randomnerdtutorials.com/esp32-uart-communication-serial-