ESP32 to ESP8266 UART Communication to Control LEDs — Full Project Documentation

Components Required:

Component	Quantity
ESP32 Dev Board	1
ESP8266 NodeMCU	1
LEDs	3
220Ω Resistors	3
Jumper Wires	10+
Breadboard	1

Objective:

- Send commands like `"red on"`, `"green off"` etc. from **ESP32 (via Serial Monitor)**.
- ESP8266 receives these commands via **UART** and controls LEDs accordingly.

Wiring Diagram:

ESP32 (Sender) to ESP8266 (Receiver):

ESP32 Pin	ESP8266 NodeMCU Pin	Purpose
GPIO17 (TX)	D2 (GPIO4 RX)	UART TX → RX
GPIO16 (RX)	D1 (GPIO5 TX)	UART RX ← TX
GND	GND	Common GND

ESP8266 (LED Connections):

ESP8266 Pin	LED Color	Other End
D5 (GPIO14)	Red	220Ω resistor → LED → GND
D6 (GPIO12)	Yellow	220Ω resistor → LED → GND
D8 (GPIO15)	Green	220Ω resistor → LED → GND

Software Tools:

Development Board	IDE
ESP32	PlatformIO / VSCode
ESP8266	Arduino IDE

ESP32 Code (Sender):

```
#include <Arduino.h>
HardwareSerial MySerial(1);

void setup() {
  Serial.begin(115200);
  MySerial.begin(9600, SERIAL_8N1, 16,17);
  delay(5000);
  Serial.println("Type Colour");
}

void loop() {
  if(Serial.available())
  {
    String input = Serial.readStringUntil('\n');
    MySerial.println(input);
    Serial.println("Sent : " +input);
  }
}
```

ESP32 (Sender) Code Explanation:

```
#include <Arduino.h> // Include the core Arduino library
```

- This includes the Arduino library which is required to use standard Arduino functions like `Serial.begin()`, `Serial.println()`, etc.

```
HardwareSerial MySerial(1); // Declare UART1 for communication (ESP32 has 3 UARTs)
```

- ESP32 has 3 hardware serial ports (UART0, UART1, UART2). Here, UART1 is being used for communication with the ESP8266.

```
void setup() {
  Serial.begin(115200); // Initialize Serial Monitor communication at 115200 baud rate
  MySerial.begin(9600, SERIAL_8N1, 16, 17); // Initialize UART1 at 9600 baud using GPIO16 (RX)
and GPIO17 (TX)
```

```
  delay(5000); // Wait 5 seconds for stability (good for USB serial connection)
  Serial.println("Type Colour Command like 'red on' or 'green off'"); // Instruction on Serial
Monitor
}
```

- `Serial.begin(115200)` sets up the communication between the ESP32 and the computer (USB Serial Monitor).
- `MySerial.begin(9600, SERIAL_8N1, 16, 17)` configures UART1 with 9600 baud rate, 8 data bits, no parity, and 1 stop bit. TX is GPIO17 and RX is GPIO16.
- A delay of 5 seconds ensures everything initializes properly before communication begins.
- A message is printed to the Serial Monitor to inform the user about the command format.

```
void loop() {
  if (Serial.available()) { // Check if data is available from the Serial Monitor
    String input = Serial.readStringUntil('\n'); // Read the full string until newline character
    MySerial.println(input); // Send the string via UART1 to ESP8266
    Serial.println("Sent: " + input); // Print the sent command on Serial Monitor for
confirmation
  }
}
```

- The loop constantly checks for any input from the computer via the Serial Monitor.
- When a line is received (terminated by `\n`), it is read into the `input` string.
- This string is then sent to the ESP8266 using `MySerial.println()`.
- The same string is printed back to the Serial Monitor for user feedback.

ESP8266 Code (Receiver + LED Control):

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(D2,D1);
#define red D5
#define yellow D6
#define green D8

void setup() {
  Serial.begin(115200);
  mySerial.begin(9600);
  pinMode(red,OUTPUT);
  pinMode(yellow,OUTPUT);
  pinMode(green,OUTPUT);
}

void loop() {

  if(mySerial.available())
  {
    String command = mySerial.readStringUntil('\n');
    command.trim();
    Serial.print(" Command = " + command);

    if(command == "red on") digitalWrite(D5,HIGH);

    if(command == "red off") digitalWrite(D5,LOW);

    if(command == "yellow on") digitalWrite(D6,HIGH);

    if(command == "yellow off") digitalWrite(D6,LOW);

    if(command == "green on") digitalWrite(D8,HIGH);

    if(command == "green off") digitalWrite(D8,LOW);
  }

}
```

ESP8266 (Receiver) Code Explanation:

```
#include <SoftwareSerial.h> // Include SoftwareSerial library for UART on arbitrary pins
```

- The ESP8266 generally has only one hardware serial port, so SoftwareSerial is used to create another serial port on GPIO4 and GPIO5.

```
SoftwareSerial mySerial(D2, D1); // Declare SoftwareSerial: RX=D2 (GPIO4), TX=D1 (GPIO5)
```

- Defines a SoftwareSerial object named `mySerial` on pins D2 (RX) and D1 (TX).
- D2 will receive data coming from ESP32's TX.

```
#define red D5     // Define 'red' LED on pin D5 (GPIO14)

#define yellow D6  // Define 'yellow' LED on pin D6 (GPIO12)

#define green D8   // Define 'green' LED on pin D8 (GPIO15)
```

- These are easy-to-read names assigned to each LED pin.

```
void setup() {

  Serial.begin(115200);    // Initialize Serial Monitor for debugging

  mySerial.begin(9600);    // Initialize SoftwareSerial (UART) to receive from ESP32

  pinMode(red, OUTPUT);    // Set red LED pin as output

  pinMode(yellow, OUTPUT); // Set yellow LED pin as output

  pinMode(green, OUTPUT);  // Set green LED pin as output

}
```

- Serial communication to the PC (USB) and ESP32 (via SoftwareSerial) is initialized.
- The LED pins are set as outputs, meaning they will provide voltage to the LEDs.

```
void loop() {

  if (mySerial.available()) { // Check if any data is available from ESP32

    String command = mySerial.readStringUntil('\n'); // Read the incoming command string till
newline
```

```
    command.trim(); // Remove unwanted whitespace or carriage return characters

    Serial.println(" Command = " + command); // Print the received command on Serial Monitor for debugging
```

- The program waits until a complete command is received from ESP32 via UART.
- The string is then cleaned (trimmed) to remove extra characters like `\r`.

```
   if (command == "red on") digitalWrite(red, HIGH);    // Turn red LED ON

    else if (command == "red off") digitalWrite(red, LOW);  // Turn red LED OFF

    else if (command == "yellow on") digitalWrite(yellow, HIGH);  // Turn yellow LED ON

    else if (command == "yellow off") digitalWrite(yellow, LOW);  // Turn yellow LED OFF

    else if (command == "green on") digitalWrite(green, HIGH);  // Turn green LED ON

    else if (command == "green off") digitalWrite(green, LOW);  // Turn green LED OFF

    else Serial.println("Unknown Command"); // Print this if an unexpected command arrives

  }

}
```

- The command is compared to expected strings such as `"red on"`, `"yellow off"`, etc.
- When a match is found, the appropriate LED pin is turned ON or OFF.
- If no match is found, it prints `"Unknown Command"` to help debugging.

Steps to Implement:

1. Hardware Connections:

Connect ESP32 UART TX/RX to ESP8266 UART RX/TX (cross-connected).
Common Ground between ESP32 & ESP8266.
Connect 3 LEDs with 220Ω resistors to D5, D6, D8 of ESP8266.

2. Flash ESP8266:

Open **Arduino IDE** → Select:

- **Board:** "NodeMCU 1.0 (ESP-12E Module)"
- **Port:** Your ESP8266 COM port.
- Paste **ESP8266 Code**, compile & upload.

3. Flash ESP32:

Open **PlatformIO (VSCode)** → Select:

- **Board:** "ESP32 Dev Module"
- Paste **ESP32 Code**, compile & upload.

4. Serial Monitor Settings:

ESP32 Serial Monitor: 115200 baud → Type:
`red on`, `red off`, `yellow on`, `green off`, etc.

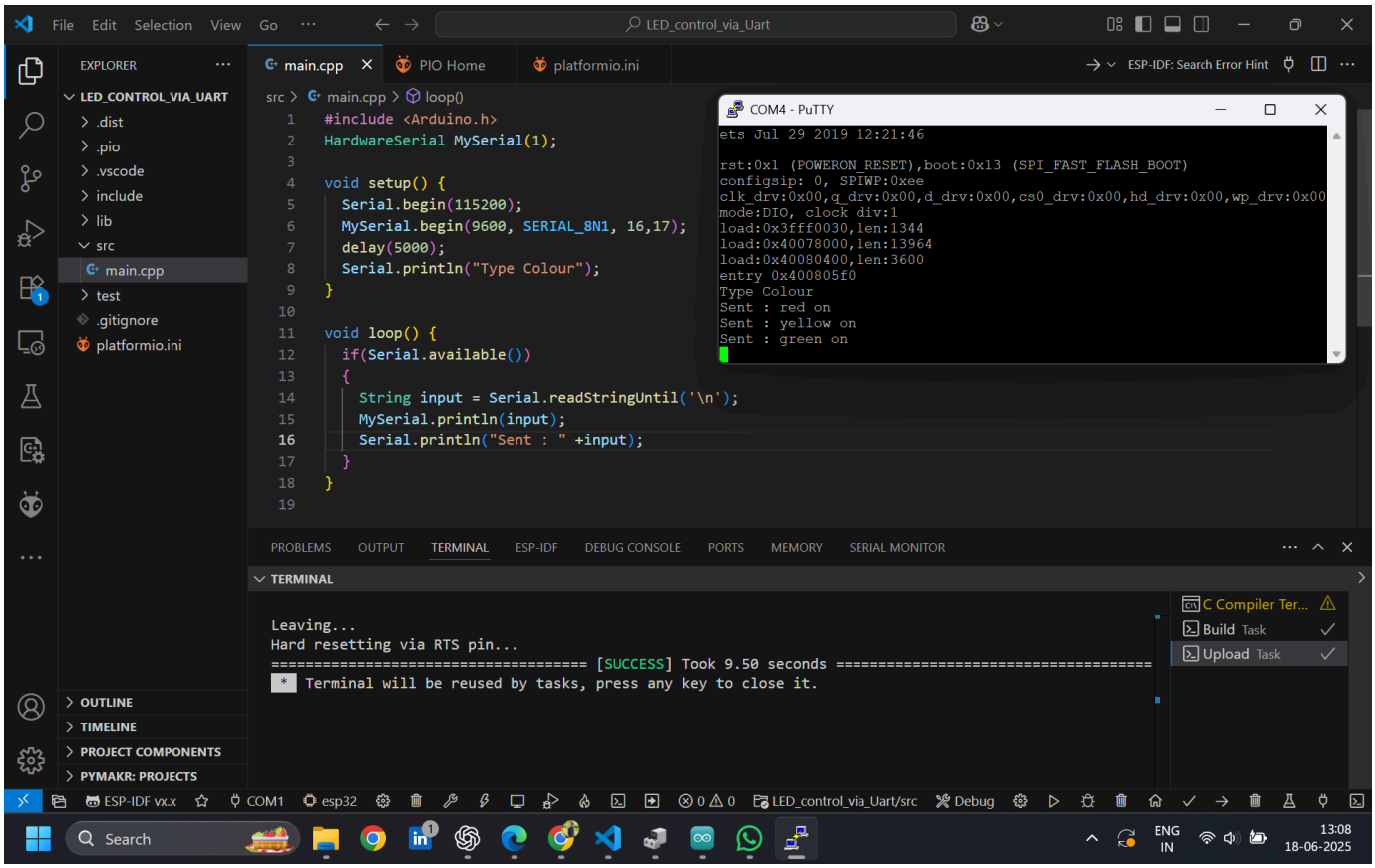
ESP8266 Serial Monitor: 115200 baud (Optional for debugging).

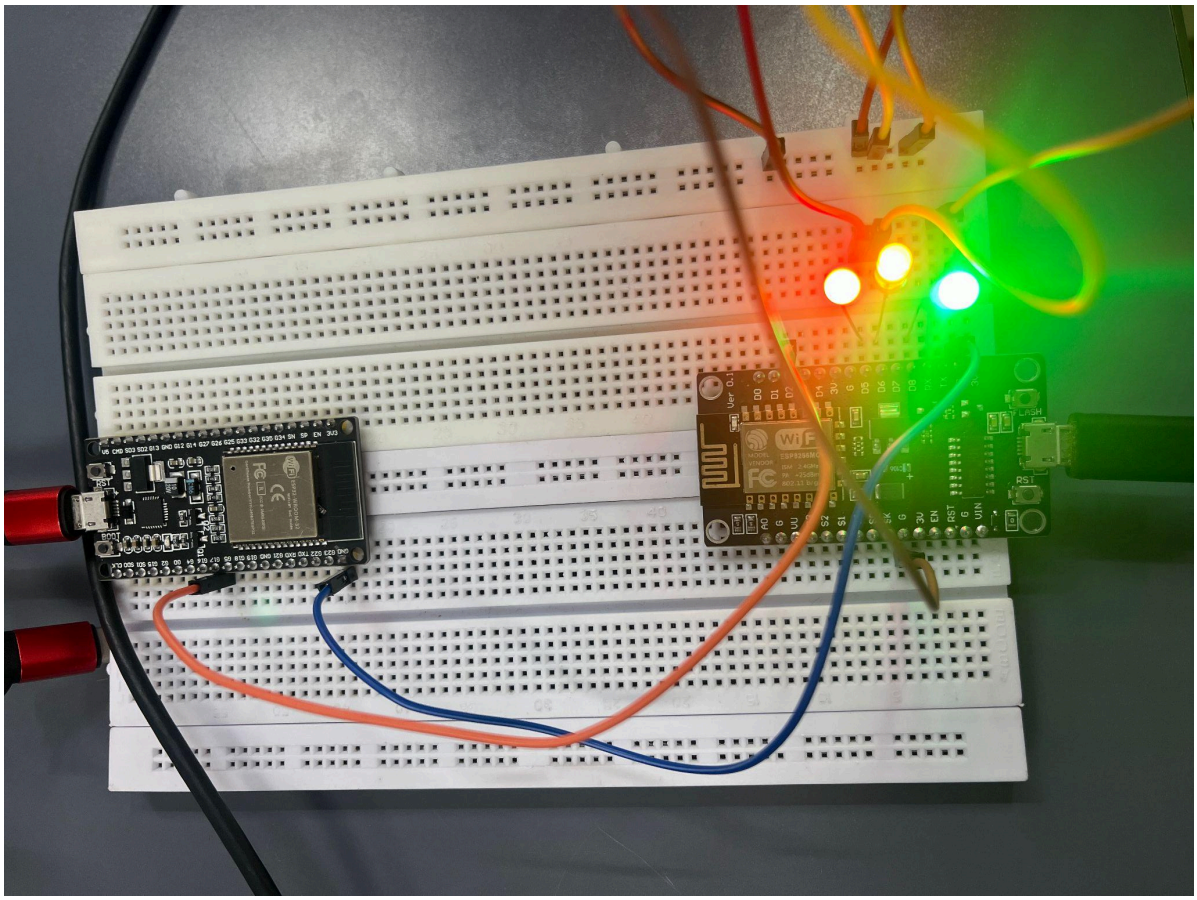
5. Test:

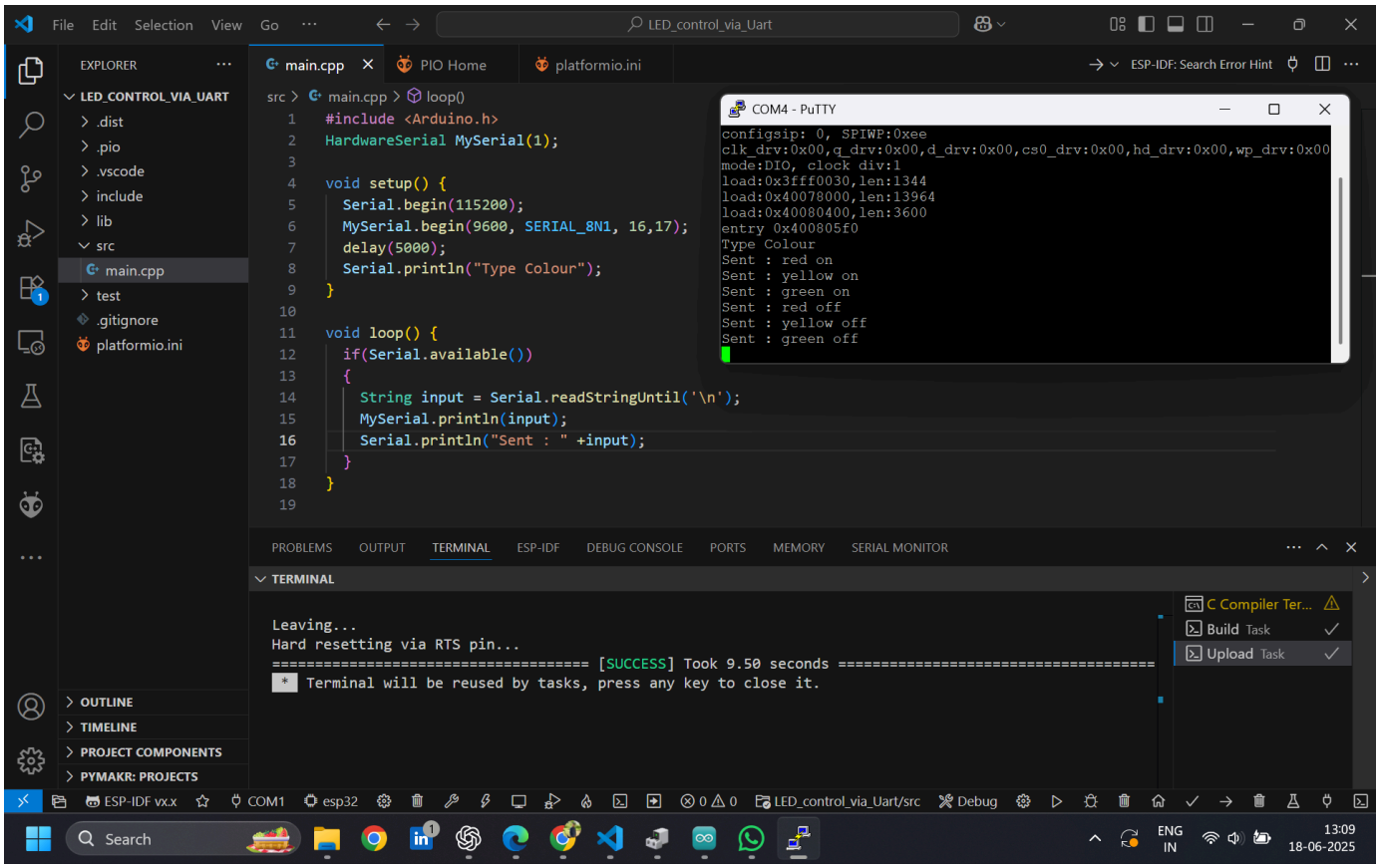
Type `red on` → **Red LED turns ON**.
Type `yellow off` → **Yellow LED turns OFF**.
Works for Green as well.

Notes:

1. **Important: Always connect the GND of ESP32 and ESP8266 together.**
2. Use **separate USB cables** for programming each board.
3. Remove ESP32's UART connection from ESP8266 during **flashing** to avoid interference.
4. After flashing, reconnect wires for UART.


File Edit Selection View Go
LED_control_via_Uart
EXPLORER
LED_CONTROL_VIA_UART
.dist
.pio
.vscode
include
lib
src
main.cpp
test
.gitignore
platformio.ini
main.cpp
PIO Home
platformio.ini
ESP-IDF: Search Error Hint
src > main.cpp > loop()
#include <Arduino.h>
HardwareSerial MySerial(1);
void setup() {
Serial.begin(115200);
MySerial.begin(9600, SERIAL_8N1, 16,17);
delay(5000);
Serial.println("Type Colour");
}
void loop() {
if(Serial.available())
{
String input = Serial.readStringUntil('\n');
MySerial.println(input);
Serial.println("Sent : " +input);
}
}
COM4 - PuTTY
ets Jul 29 2019 12:21:46
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1344
load:0x40078000,len:13964
load:0x40080400,len:3600
entry 0x400805f0
Type Colour
Sent : red on
Sent : yellow on
Sent : green on
PROBLEMS OUTPUT TERMINAL ESP-IDF DEBUG CONSOLE PORTS MEMORY SERIAL MONITOR
TERMINAL
Leaving...
Hard resetting via RTS pin...
[SUCCESS] Took 9.50 seconds
Terminal will be reused by tasks, press any key to close it.
C Compiler Ter...
Build Task
Upload Task
OUTLINE
TIMELINE
PROJECT COMPONENTS
PYMAKR: PROJECTS
ESP-IDF vx.x
COM1
esp32
LED_control_via_Uart/src
Debug
Search
ENG
IN
13:08
18-06-2025




File Edit Selection View Go
LED_control_via_Uart
EXPLORER
LED_CONTROL_VIA_UART
.dist
.pio
.vscode
include
lib
src
main.cpp
test
.gitignore
platformio.ini
main.cpp
PIO Home
platformio.ini
ESP-IDF: Search Error Hint
src > main.cpp > loop()
#include <Arduino.h>
HardwareSerial MySerial(1);
void setup() {
Serial.begin(115200);
MySerial.begin(9600, SERIAL_8N1, 16,17);
delay(5000);
Serial.println("Type Colour");
}
void loop() {
if(Serial.available())
{
String input = Serial.readStringUntil('\n');
MySerial.println(input);
Serial.println("Sent : " +input);
}
}
COM4 - PuTTY
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1344
load:0x40078000,len:13964
load:0x40080400,len:3600
entry 0x400805f0
Type Colour
Sent : red on
Sent : yellow on
Sent : green on
Sent : red off
Sent : yellow off
Sent : green off
PROBLEMS OUTPUT TERMINAL ESP-IDF DEBUG CONSOLE PORTS MEMORY SERIAL MONITOR
TERMINAL
Leaving...
Hard resetting via RTS pin...
[SUCCESS] Took 9.50 seconds
Terminal will be reused by tasks, press any key to close it.
C Compiler Ter...
Build Task
Upload Task
OUTLINE
TIMELINE
PROJECT COMPONENTS
PYMAKR: PROJECTS
ESP-IDF vx.x
COM1
esp32
LED_control_via_Uart/src
Debug
Search
ENG
IN
13:09
18-06-2025