ESP32 ↔ STM32 UART Communication (Sourya Board)

1. Overview

This project demonstrates **UART communication between ESP32 and STM32 (Sourya Board)**.

- ESP32 sends commands (1 or 0)
- STM32 receives commands and:
  - Turns **LED ON (1)**
  - Turns **LED OFF (0)**
- STM32 sends back response:
  - "LED ON" or "LED OFF"

Communication is done using **USART1 internally**, without external crossing hardware.

2. Hardware Connections(Just Memorize, No need of connections)

STM32 (Sourya Board) ↔ ESP32

STM32	Pin Function	ESP32 Pin
PA9	USART1_TX	GPIO16 (RXD2)
PA10	USART1_RX	GPIO17 (TXD2)

ST-Link V2 Connection (For Flashing STM32)

ST-Link	Sourya Board Pin
SWDIO	Pin 7
SWCLK	Pin 9
3.3V	Pin 1
GND	Pin 12

ESP32 Upload (Using TTL Adapter)

TTL Adapter	ESP32
RX	TX
TX	RX
GND	GND
5V	5V

3. STM32 Configuration (STM32CubeIDE)

Step 1: Create Project

- Open **STM32CubeIDE**
- Create new project for your STM32 board

Step 2: Clock Configuration

- Set:
    - **RCC → HSE → Crystal/Ceramic Resonator**

Step 3: Pin Configuration

- **PA9 → USART1_TX**
- **PA10 → USART1_RX**
- **PA6 → GPIO Output (LED)**

Step 4: USART Settings

- Enable **USART1**
- Mode: **Asynchronous**
- Keep default:
    - Baud Rate: 115200
    - Parity: None
    - Stop Bits: 1

Step 5: Generate Code

- Press **Ctrl + S**
- Code will be generated (main.c)

STM32 Code Implementation

Global Variables (Add above main)

```
uint8_t ch;
uint8_t rx_data[10];
int idx = 0;
```

Add Inside while(1)

```
HAL_UART_Receive(&huart1, &ch, 1, HAL_MAX_DELAY);

if (ch == '\n')   // end of message
{
   rx_data[idx] = '\0';

   if (strstr((char*)rx_data, "1") != NULL)
   {
      HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);
      HAL_UART_Transmit(&huart1, (uint8_t*)"LED ON\r\n", 8, HAL_MAX_DELAY);
   }
   else if (strstr((char*)rx_data, "0") != NULL)
   {
      HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET);
      HAL_UART_Transmit(&huart1, (uint8_t*)"LED OFF\r\n", 9, HAL_MAX_DELAY);
   }

   idx = 0;
   memset(rx_data, 0, sizeof(rx_data));
}
else
{
   if (idx < sizeof(rx_data) - 1)
      rx_data[idx++] = ch;
}
```

Step 6: Build & Flash

- Click **Build**
- Flash using **ST-Link V2**

ESP32 Code (Arduino IDE)

Code

```
#define RXD2 16
#define TXD2 17

HardwareSerial mySerial(2);

void setup() {
  Serial.begin(115200);          // PC Serial Monitor
  mySerial.begin(115200, SERIAL_8N1, RXD2, TXD2);  // UART2 to STM32

  Serial.println("Type 1 for ON or 0 for OFF:");
}

void loop() {

  if (mySerial.available()) {
    String data = mySerial.readString();
    Serial.println("From STM32: " + data);
  }

  if (Serial.available()) {
    String command = Serial.readStringUntil('\n');
    command.trim();

    if (command == "1") {
      mySerial.println("1");
      Serial.println("Sent ON to STM32");
    }
    else if (command == "0") {
      mySerial.println("0");
      Serial.println("Sent OFF to STM32");
    }
  }
}
```

Step 7: Upload ESP32 Code

- Select correct **COM port**
- Click **Upload**

LED Connection

- Connect LED to:
    - **PA6 (GPIO Output) –D0**

- Through resistor to **GND**

Working Flow

1. User types 1 or 0 in Serial Monitor
2. ESP32 sends command via UART
3. STM32 receives command:
    - 1 → LED ON
    - 0 → LED OFF
4. STM32 sends response:
    - "LED ON" / "LED OFF"
5. ESP32 displays response on Serial Monitor

Output Example

Type 1 for ON or 0 for OFF:

1

Sent ON to STM32

From STM32: LED ON

0

Sent OFF to STM32

From STM32: LED OFF

Key Notes

- Ensure **baud rate is same (115200)** on both sides
- Proper **TX ↔ RX connection**
- Use **\n (newline)** as message terminator
- Buffer size should not overflow