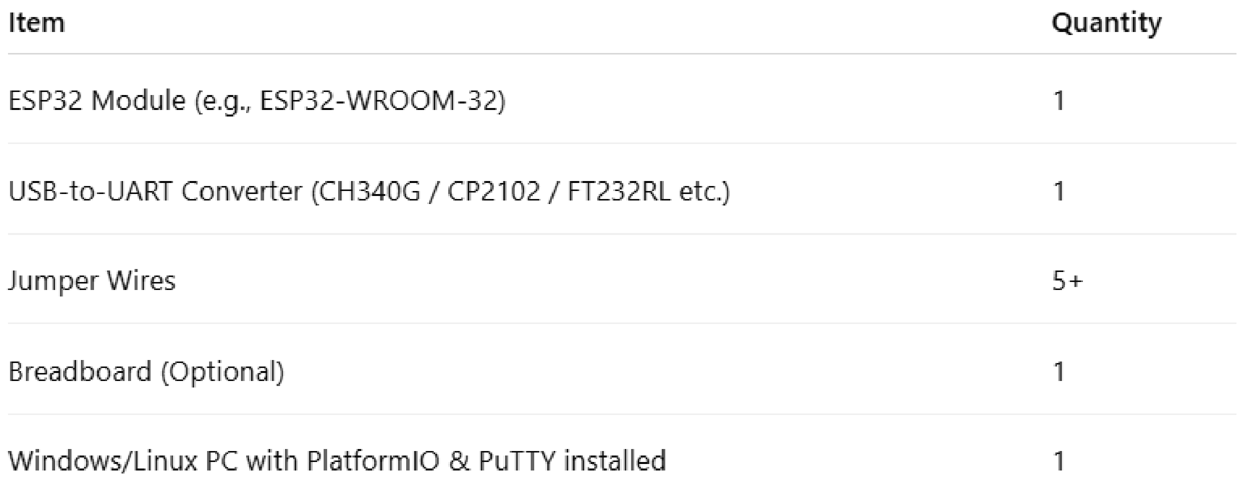
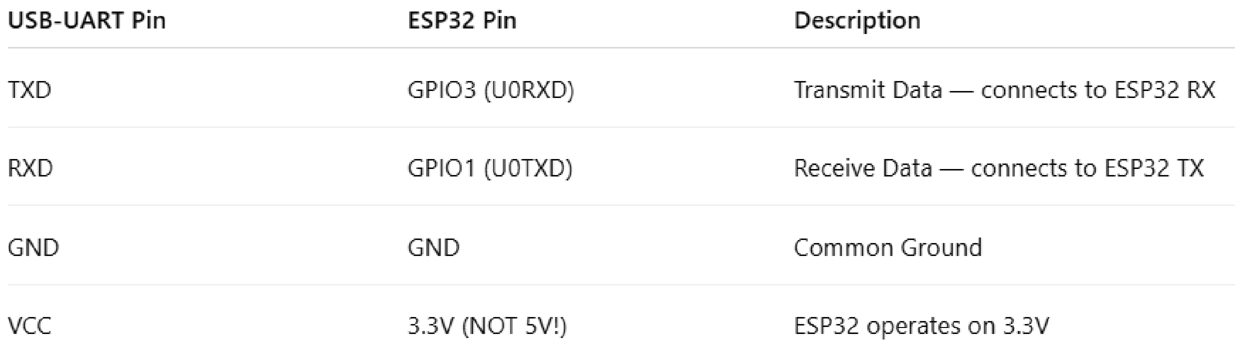
ESP32 + USB-to-UART Converter: Complete Setup & Communication Guide (Using PlatformIO and PuTTY)

1. Required Components



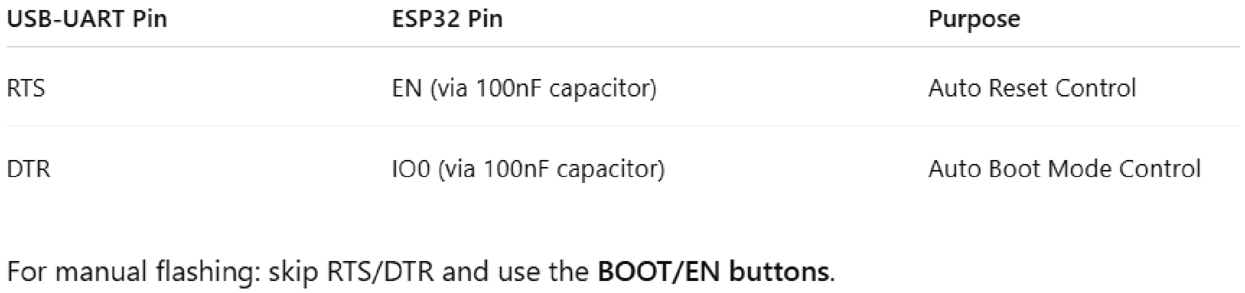
Item	Quantity
ESP32 Module (e.g., ESP32-WROOM-32)	1
USB-to-UART Converter (CH340G / CP2102 / FT232RL etc.)	1
Jumper Wires	5+
Breadboard (Optional)	1
Windows/Linux PC with PlatformIO & PuTTY installed	1

2. Hardware Connection (ESP32 ↔ USB-UART Converter)



USB-UART Pin	ESP32 Pin	Description
TXD	GPIO3 (U0RXD)	Transmit Data — connects to ESP32 RX
RXD	GPIO1 (U0TXD)	Receive Data — connects to ESP32 TX
GND	GND	Common Ground
VCC	3.3V (NOT 5V!)	ESP32 operates on 3.3V

Optional for automatic flashing:



USB-UART Pin	ESP32 Pin	Purpose
RTS	EN (via 100nF capacitor)	Auto Reset Control
DTR	IO0 (via 100nF capacitor)	Auto Boot Mode Control

For manual flashing: skip RTS/DTR and use the **BOOT/EN buttons**.

3. Power Notes:

- ESP32 is sensitive to **5V** on its 3.3V pin — ensure your USB-UART adapter provides 3.3V if directly powering ESP32.

- For **bare ESP32 modules**, you **must provide stable 3.3V power** (some USB-UART modules have jumpers for this).

4. PlatformIO Setup (Using Arduino Framework)

Step 1: Install PlatformIO IDE Extension (in VS Code).

Step 2: Create New Project:

- **Board**: ESP32 Dev Module (esp32dev)
- **Framework**: Arduino

Step 3: Modify `platformio.ini`:

```
[env:upesy_wroom]
platform = espressif32
board = upesy_wroom
framework = arduino
monitor_speed = 115200
upload_speed = 115200
upload_port = COM4
```

Find **COM port** via **Device Manager** → **Ports (COM & LPT)**.

5. Sample Code (src/main.cpp)

```
#include <Arduino.h>

void setup() {
  Serial.begin(115200);
}

void loop() {
  if(Serial.available())
  {
    String data = Serial.readStringUntil('\n');
    if(data)
    {
      Serial.println(data);
    }
  }
}
```

6. Manual Flashing Process (No Auto-Reset Circuit)

Before clicking Upload in PlatformIO:

1. **Press and hold the BOOT button** on ESP32.
2. **Tap the EN (Reset) button once.**
3. **Release BOOT button** after 1 second.
4. In PlatformIO, click **Upload** (or run: `pio run --target upload`).
5. Wait for a successful upload.

7. Serial Monitor Options (View ESP32 Output)

PuTTY Terminal:

Step-by-Step PuTTY Setup:

1. Open **PuTTY**.
2. Select **Serial**.
3. Configure:
   - **Serial line**: `COMx` (Find in Device Manager)
   - **Speed (baud)**: `115200`
4. Go to **Connection** → **Serial**:
   - Data bits: **8**
   - Stop bits: **1**
   - Parity: **None**
   - Flow Control: **None**
5. Click **Open**.

8. Viewing Startup Message in PuTTY

- To see `"ESP 32 UART TEST"` line in PuTTY:
  1. **Open PuTTY FIRST**.
  2. **Reset ESP32 manually (press EN button)**.
  3. ESP32 restarts and sends the startup message — PuTTY will now catch it.

OR

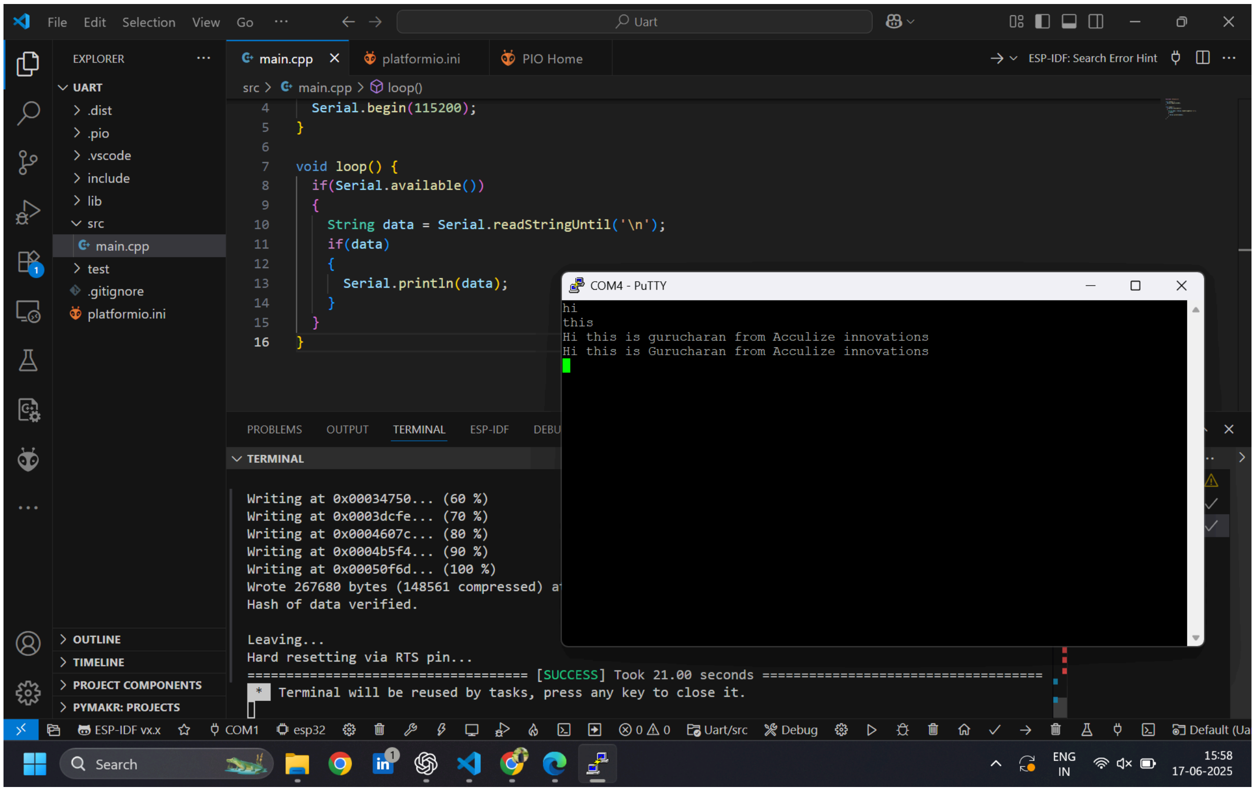
- Use `delay(2000);` in `setup()` to allow PuTTY time to catch output after ESP32 reset.

9. Important Troubleshooting Tips

Problem	Reason/Solution
Startup message missing in PuTTY	Add `delay(2000);` in setup OR reset ESP32 after PuTTY is open
ESP32 not uploading	Enter flashing mode manually (BOOT + EN method)
Garbage output	Wrong baud rate — use `115200`
COM port not found	COM port busy — close other Serial Monitors (PlatformIO etc.)
"Brownout Detector" reset error	Insufficient power — provide reliable 3.3V

10. Summary Flow

`ESP32 Wiring ➡ PlatformIO Code Upload ➡ Manual Reset ➡ PuTTY View`



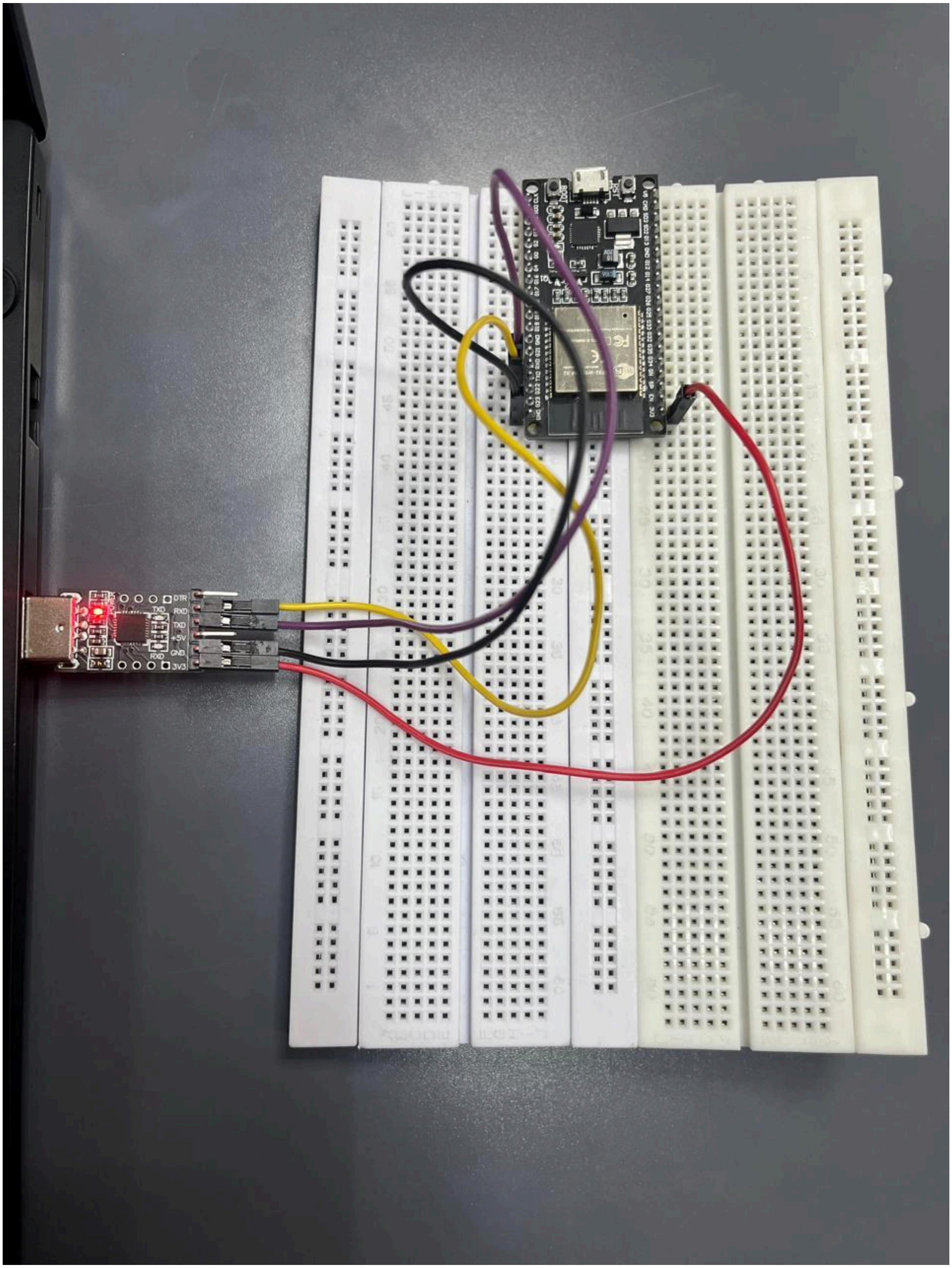
```cpp
  Serial.begin(115200);
}

void loop() {
  if(Serial.available())
  {
    String data = Serial.readStringUntil('\n');
    if(data)
    {
      Serial.println(data);
    }
  }
}
```

```
hi
this
Hi this is gurucharan from Acculize innovations
Hi this is Gurucharan from Acculize innovations
```

```
Writing at 0x00034750... (60 %)
Writing at 0x0003dcfe... (70 %)
Writing at 0x0004607c... (80 %)
Writing at 0x0004b5f4... (90 %)
Writing at 0x00050f6d... (100 %)
Wrote 267680 bytes (148561 compressed) a
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
==================================== [SUCCESS] Took 21.00 seconds ====================================
 *  Terminal will be reused by tasks, press any key to close it.
```


TXD
DTR
RXD
TXD
+5V
GND
RXD
3V3