MICRO ROBOTICS
Centurion Tel 012 665 5713
Stellenbosch Tel 021 007 2971
info@microgroup.co.za

Introduction:

The ESP32-S3-N16R8 uses the ESP32-S3 chip, suitable for IoT application testing prototypes and actual applications. It is equipped with two USB ports, one is a hardware USB to serial port (CH343P WCH), and the other is the ESP32-S3 USB port.

This guide will help you quickly get started with the ESP32-S3-N16R8 and provide detailed information about this development board.

The ESP32-S3-N16R8 is an entry-level development board with a Wi-Fi + Bluetooth® LE module ESP32-S3-WROOM-1.

Most of the module's pins are exposed on both sides of the development board, allowing developers to easily connect various peripherals through jumper wires or insert the development board into a breadboard.

1. This is a minimal core board for the ESP32-S3, using Espressif's ESP32-S3 module.
2. The board has a dedicated LDO circuit for wireless functionality, ensuring sufficient current (power).
3. It is equipped with a WS2812-RGB LED (note: it is not directly controlled by GPIO).
4. The RST button is for external reset, while the boot button (used with the RST button) can guide into bootloader mode and act as a user button after reset (GPIO0).
5. The board has two TYPE-C ports (one is directly connected to USB (GPIO19, GPIO20), and the other is a USB to serial port). It includes a hardware USB to serial chip (CH343).

Ensure Correct Driver Installation: Install driver. Double-check if the installation was successful and that the driver is correctly associated with your board in the Device Manager (for Windows) or equivalent on your OS. If necessary, try reinstalling the driver from this link.

Board Connection and Detection:

- Make sure you are using the "COM" marked USB-C port for communication.
- Sometimes, USB cables can be faulty. Try using a different, data-capable USB cable.
- Ensure that the board is receiving power, check for any indicator LEDs.

Pads:

• **RGB LED Interface**: If you want to interface the RGB LED, you need to bridge the pads marked as "RGB." Please note that once this is done, the specific pin (GPIO48) can only be used to interface the RGB LED.

• **IN-OUT Pads**: These pads are used to configure the 5V pin as either an input or an output. When unbridged, it is set as an input, and when bridged, it is set as an output. This configuration is necessary if you want to use the USB-C connector port "COM" to power the ESP32 while also having a 5V output on the 5V pin.

• **USB-OTG Function**: The pads underneath the module marked as "USB-OTG" are used to enable the USB-OTG function on the "USB" USB-C connector.

Correct Board Settings in Arduino IDE:

- Open the Arduino IDE.
- Go to `Tools` -> `Board` -> `ESP32S3 Dev Module`.
- Ensure the correct port is selected under `Tools` -> `Port`.
- Try uploading a basic blink example to see if the IDE detects the board.

Board Upload Mode: The board should be upload-ready out of the box. However, if it isn't detected, you might need to put it into download mode manually:

- Hold the `BOOT` button.
- Press the `RESET` button while holding `BOOT`.
- Release the `RESET` button.
- Release the `BOOT` button. This sequence should put the board into bootloader mode.

Alternative Resources and Documentation:

- Check the GitHub repository for any updates or additional instructions: YD-ESP32-S3 GitHub
- Refer to the datasheet and pinout documentation you can find in the repository or provided with the board.

Hardware Overview:

Main Component	Description
ESP32-S3-WROOM-1	A general-purpose Wi-Fi + low-power Bluetooth MCU module with rich peripheral interfaces, powerful neural network computing, and signal processing capabilities, designed for AI and AIoT markets. It uses a PCB onboard antenna.
5 V to 3.3 V LDO	Power converter with 5 V input and 3.3 V output, 1A current
Pin Headers	All available GPIO pins (except the flash SPI bus) are exposed to pin headers on the development board.
USB-to-UART Port	Type-C USB interface used for power supply, firmware flashing, and communication via the onboard USB to UART bridge.
Boot Button	Download button. Press and hold Boot while pressing Reset to enter "firmware download" mode via UART. Can be used as a regular input button (GPIO0) after startup.
Reset Button	Reset button.
USB Port	ESP32-S3 USB OTG interface, supporting full-speed USB 1.1 standard. Used for power supply, firmware flashing, USB communication, and JTAG debugging.
USB-to-UART Bridge	CH343P chip from WCH.
RGB LED	Addressable RGB LED driven by GPIO48 (WS2812).
PWR LED	Power indicator that lights up when the board is powered, not controllable by software.
TX LED	LED on the ESP32-S3 UART TX line that blinks with serial data transmission. Can be used as GPIO if the serial function is not needed (GPIO43).
RX LED	LED on the ESP32-S3 UART RX line that blinks with serial data reception. Can be used as GPIO if the serial function is not needed (GPIO44).

Notes: For the onboard ESP32-S3-WROOM-1 module series (using an 8-line SPI flash/PSRAM), GPIO35, GPIO36, and GPIO37 are used for internal communication between the ESP32-S3 chip and SPI flash/PSRAM and are not available for external use.

Starting Development: Before powering up, ensure the development board is in good condition.

Power Options: You can power the board using one of the following methods:

1. USB-to-UART port or ESP32-S3 USB port (recommended).
2. 5V and G (GND) pin headers.
3. 3V3 and G (GND) pin headers.

Pin Headers: The following tables list the names and functions of the pin headers (P1 and P2) on both sides of the development board. The pin names are shown in the YD-ESP32-S3 front view, and the pin numbers correspond to the development board schematic (PDF).

P1

No.	Name	Type	Function
1	3V3	P	3.3 V Power
2	3V3	P	3.3 V Power
3	RST	I	EN
4	4	I/O/T	RTC_GPIO4, GPIO4, TOUCH4, ADC1_CH3
5	5	I/O/T	RTC_GPIO5, GPIO5, TOUCH5, ADC1_CH4
6	6	I/O/T	RTC_GPIO6, GPIO6, TOUCH6, ADC1_CH5
7	7	I/O/T	RTC_GPIO7, GPIO7, TOUCH7, ADC1_CH6
8	15	I/O/T	RTC_GPIO15, GPIO15, U0RTS, ADC2_CH4, XTAL_32K_P
9	16	I/O/T	RTC_GPIO16, GPIO16, U0CTS, ADC2_CH5, XTAL_32K_N
10	17	I/O/T	RTC_GPIO17, GPIO17, U1TXD, ADC2_CH6
11	18	I/O/T	RTC_GPIO18, GPIO18, U1RXD, ADC2_CH7, CLK_OUT3
12	8	I/O/T	RTC_GPIO8, GPIO8, TOUCH8, ADC1_CH7, SUBSPICS1
13	3	I/O/T	RTC_GPIO3, GPIO3, TOUCH3, ADC1_CH2
14	46	I/O/T	GPIO46
15	9	I/O/T	RTC_GPIO9, GPIO9, TOUCH9, ADC1_CH8, FSPIHD, SUBSPIHD
16	10	I/O/T	RTC_GPIO10, GPIO10, TOUCH10, ADC1_CH9, FSPICS0, FSPIIO4, SUBSPICS0
17	11	I/O/T	RTC_GPIO11, GPIO11, TOUCH11, ADC2_CH0, FSPID, FSPIIO5, SUBSPID
18	12	I/O/T	RTC_GPIO12, GPIO12, TOUCH12, ADC2_CH1, FSPICLK, FSPIIO6, SUBSPICLK

No.	Name	Type	Function
19	13	I/O/T	RTC_GPIO13, GPIO13, TOUCH13, ADC2_CH2, FSPIQ, FSPIIO7, SUBSPIQ
20	14	I/O/T	RTC_GPIO14, GPIO14, TOUCH14, ADC2_CH3, FSPIWP, FSPIDQS, SUBSPIWP
21	5V	P	5 V Power
22	G	G	Ground

P2

No.	Name	Type	Function
1	G	G	Ground
2	TX	I/O/T	U0TXD, GPIO43, CLK_OUT1
3	RX	I/O/T	U0RXD, GPIO44, CLK_OUT2
4	1	I/O/T	RTC_GPIO1, GPIO1, TOUCH1, ADC1_CH0
5	2	I/O/T	RTC_GPIO2, GPIO2, TOUCH2, ADC1_CH1
6	42	I/O/T	MTMS, GPIO42
7	41	I/O/T	MTDI, GPIO41, CLK_OUT1
8	40	I/O/T	MTDO, GPIO40, CLK_OUT2
9	39	I/O/T	MTCK, GPIO39, CLK_OUT3, SUBSPICS1
10	38	I/O/T	GPIO38, FSPIWP, SUBSPIWP
11	37	I/O/T	SPIDQS, GPIO37, FSPIQ, SUBSPIQ
12	36	I/O/T	SPIIO7, GPIO36, FSPICLK, SUBSPICLK
13	35	I/O/T	SPIIO6, GPIO35, FSPID, SUBSPID
14	0	I/O/T	RTC_GPIO0, GPIO0
15	45	I/O/T	GPIO45
16	48	I/O/T	GPIO48, SPICLK_N, SUBSPICLK_N_DIFF, RGB LED
17	47	I/O/T	GPIO47, SPICLK_P, SUBSPICLK_P_DIFF

No.	Name	Type	Function
18	21	I/O/T	RTC_GPIO21, GPIO21
19	20	I/O/T	RTC_GPIO20, GPIO20, U1CTS, ADC2_CH9, CLK_OUT1, USB_D+
20	19	I/O/T	RTC_GPIO19, GPIO19, U1RTS, ADC2_CH8, CLK_OUT2, USB_D-
21	G	G	Ground
22	G	G	Ground

P: Power; I: Input; O: Output; T: Tri-state.

Pin Diagram:

CH340 Chip Driver Official Links:

- [CH340 Chip Driver (English)](http://www.wch-ic.com/products/CH340.html?ENGLISH)
- [CH340 Chip Driver (Chinese)](https://www.wch.cn/products/CH340.html?from=list 中文)

Micropython Firmware Download:

Use the flash_download_tool_3.9.2_0 on Windows for ESP32-S3 download and erase tools. No installation is required; just unzip and use it. Double-click the gear icon, select ESP32-S3, develop, USART, and follow the instructions. Ensure the start address is 0x00 and check the boxes as shown. If downloading fails, ensure the USB to serial driver is properly installed first.

Note:

- Do not use Thonny's built-in ESP32 downloader for downloading micropython firmware to ESP32-S3. It is designed for ESP32, not ESP32-S3, and uses the incorrect address (0x1000 instead of 0x00 for ESP32-S3).
- Do not use Micropython's official firmware with SPRAM. Use Espressif's flash tool and our custom firmware with the start address 0x00.
- Ensure CH343 USB-to-serial hardware driver is updated before use.

For TASMOTA firmware download:

- TASMOTA Documentation

For custom firmware download:

- Espressif Official Site

ESP32-S3 Documentation and Resources:

- Basic information including hardware serial CH343 drivers, YD custom Micropython firmware, firmware download tools, Micropython IDE, schematics, and dimensions: YD-ESP32-S3 Resources

Official IDF-C Programming Detailed Documentation:

- ESP32-S3 IDF Documentation

Arduino Programming Documentation:

- Arduino ESP32 Documentation

Micropython Programming Documentation:

- Micropython ESP32 Quick Reference

Flexible Pin Assignment: The various peripheral communication functions (e.g., I2C, I2S, UART, SPI) can be assigned to any GPIO pin, as detailed in the ESP32 series documentation.

Arduino IDE Code:

```
#include <Adafruit_NeoPixel.h>

// Define the pins for the LEDs
#define LED_TX_PIN 43
#define LED_RX_PIN 44
#define RGB_LED_PIN 48

// Define the number of RGB LEDs (assuming 1 WS2812 LED)
#define NUM_RGB_LEDS 1

// Create an instance of the Adafruit_NeoPixel class
Adafruit_NeoPixel rgb_led = Adafruit_NeoPixel(NUM_RGB_LEDS, RGB_LED_PIN, NEO_GRB +
NEO_KHZ800);

void setup() {
  // Initialize the serial communication
  Serial.begin(115200);

  // Initialize the LED pins
  pinMode(LED_TX_PIN, OUTPUT);
  pinMode(LED_RX_PIN, OUTPUT);

  // Initialize the RGB LED
  rgb_led.begin();
  rgb_led.show(); // Initialize all pixels to 'off'

  // Start a test sequence
  Serial.println("Starting LED test sequence...");
}

void loop() {
  // Blink the TX LED
  digitalWrite(LED_TX_PIN, HIGH);
  delay(500);
  digitalWrite(LED_TX_PIN, LOW);
  delay(500);

  // Blink the RX LED
  digitalWrite(LED_RX_PIN, HIGH);
  delay(500);
  digitalWrite(LED_RX_PIN, LOW);
  delay(500);

  // Test the RGB LED
  testRGBLED();
}

void testRGBLED() {
  // Set RGB LED to red
  rgb_led.setPixelColor(0, rgb_led.Color(255, 0, 0)); // Red
  rgb_led.show();
  delay(500);

  // Set RGB LED to green
  rgb_led.setPixelColor(0, rgb_led.Color(0, 255, 0)); // Green
```

```
  rgb_led.show();
  delay(500);

  // Set RGB LED to blue
  rgb_led.setPixelColor(0, rgb_led.Color(0, 0, 255)); // Blue
  rgb_led.show();
  delay(500);

  // Turn off the RGB LED
  rgb_led.setPixelColor(0, rgb_led.Color(0, 0, 0)); // Off
  rgb_led.show();
  delay(500);
}
```