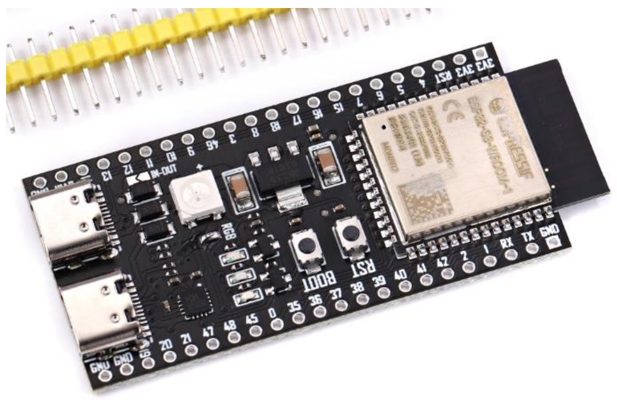


How to make a node of IoT by using ESP32-S3 DevKitC

Juozas Kimtys - How to make a node of IoT by using ESP32-S3 DevKitC – doc. ver.1.1

Table of contents:

Table of figures:

First impression

Information about similar module and various links is here:

https://docs.espressif.com/projects/esp-idf/en/latest/esp32s3/hw-reference/esp32s3/user-guide-devkitc-1.html

The module bought at AliExpress (MI YU KOUNG Official Store) is slightly different than shown on the site of Espressif – there is other type of USB connectors and other placement of parts. Look to picture below:

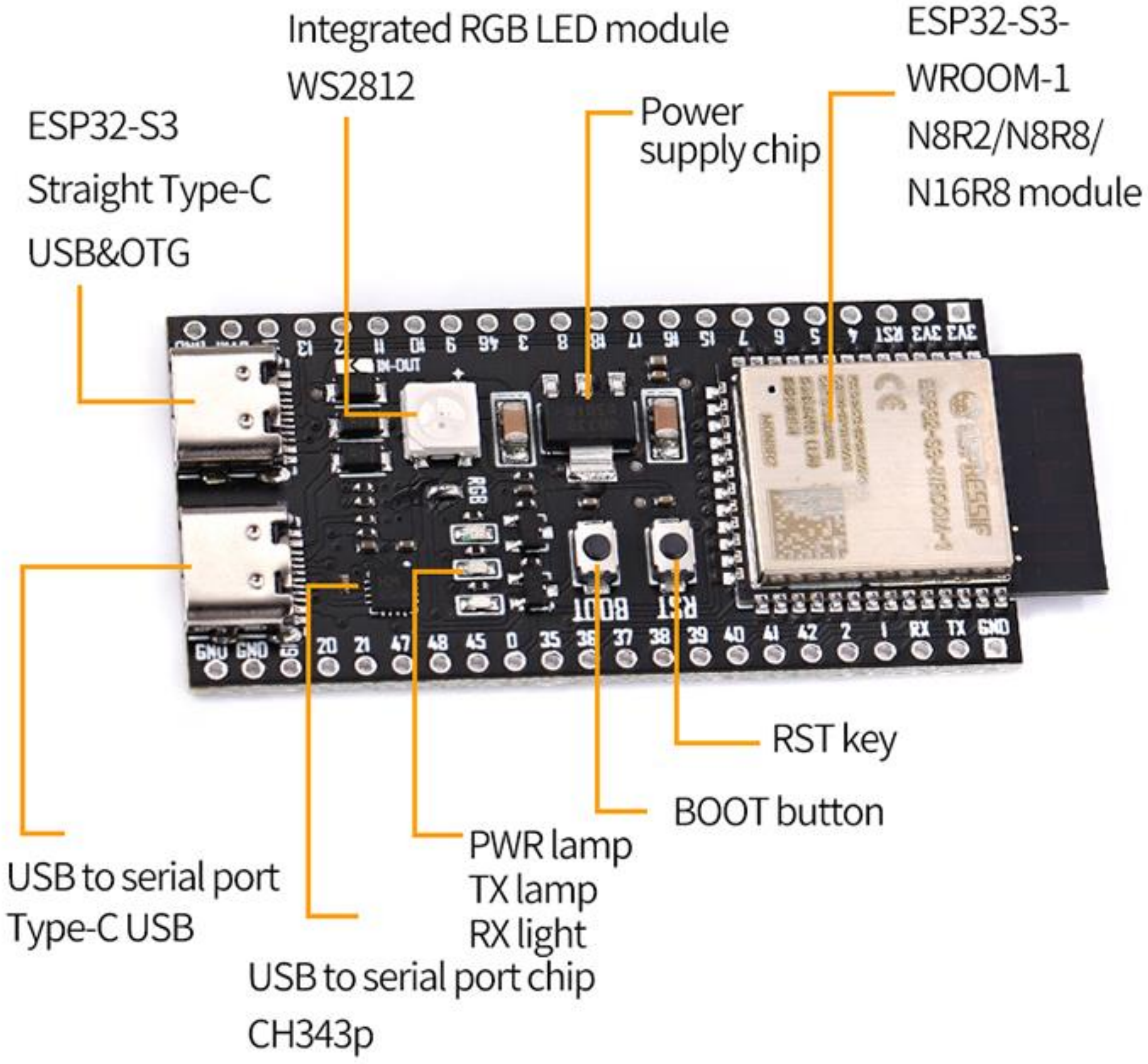

Integrated RGB LED module WS2812
ESP32-S3-WROOM-1 N8R2/N8R8/ N16R8 module
Power supply chip
ESP32-S3 Straight Type-C USB&OTG
RST key
BOOT button
PWR lamp TX lamp RX light
USB to serial port Type-C USB
USB to serial port chip CH343p

FIGURE 1 - ESP32-S3 DEVKITC FUNCTION DIAGRAM FROM SELLERS SITE.

Two USB connectors available

There are two USB connectors available on the module. One, marked as "COM" (marking is at bottom side) allows to have Virtual Serial port for communication with our application (lines RX and TX of microcontroller). Also, this port can control our program upload process by using RTS and DTR serial port lines to GPIO0 and to CHIP_PU (RST). Second, marked as "USB" (marking is at bottom side) allows to use integrated into ESP32-S3 microcontroller JTAG-to-USB debug adapter. These both USB ports can be used to upload our software from Espressif-IDE or from Arduino. It appears, that RST button also sometimes is required, regardless of additional control lines of COM port.

Layout of GPIO connector

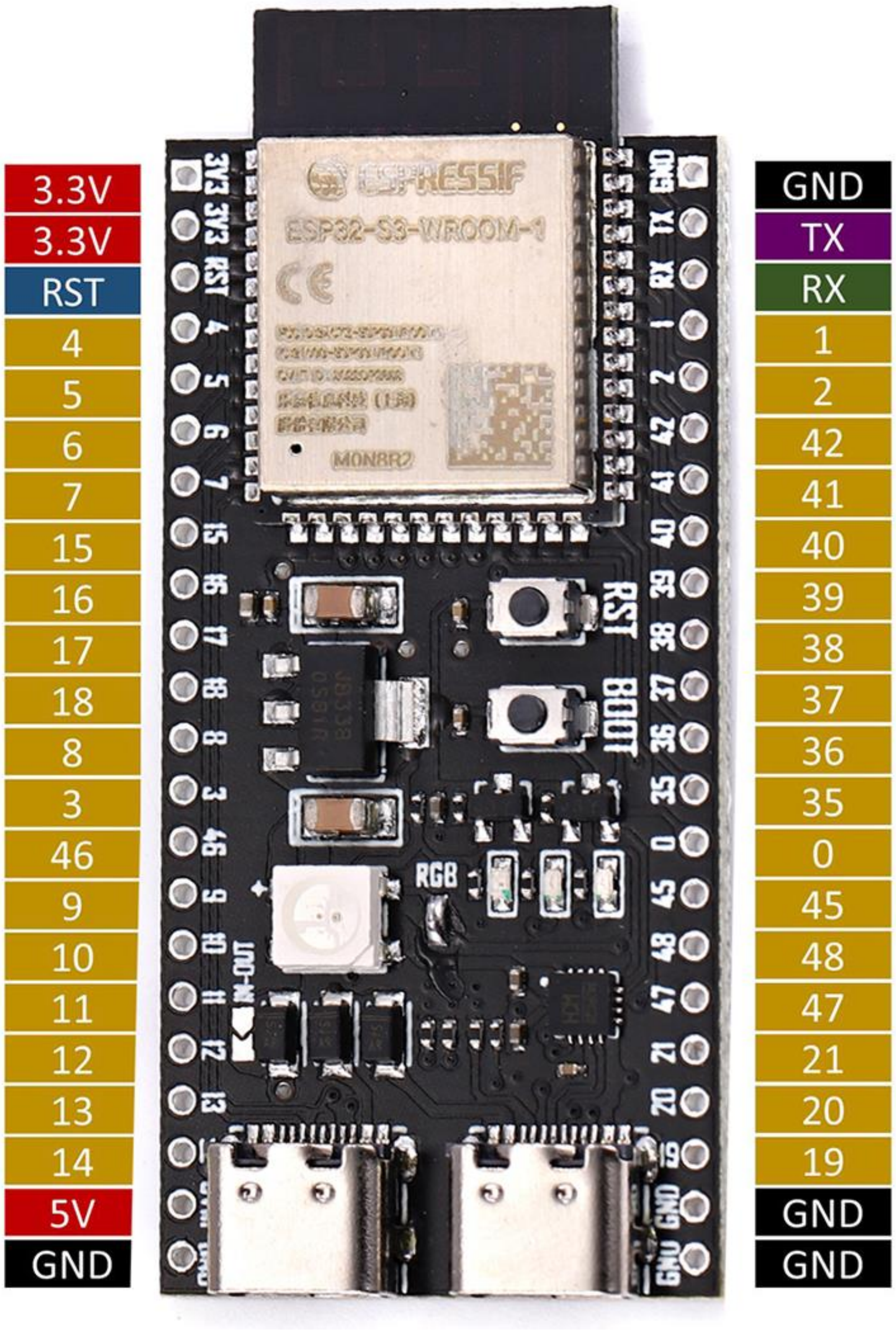


FIGURE 2 - LAYOUT OF GPIO CONNECTOR FROM SELLERS SITE. PINS NUMBERED AS GPIO NUMBERS.

Power supply options

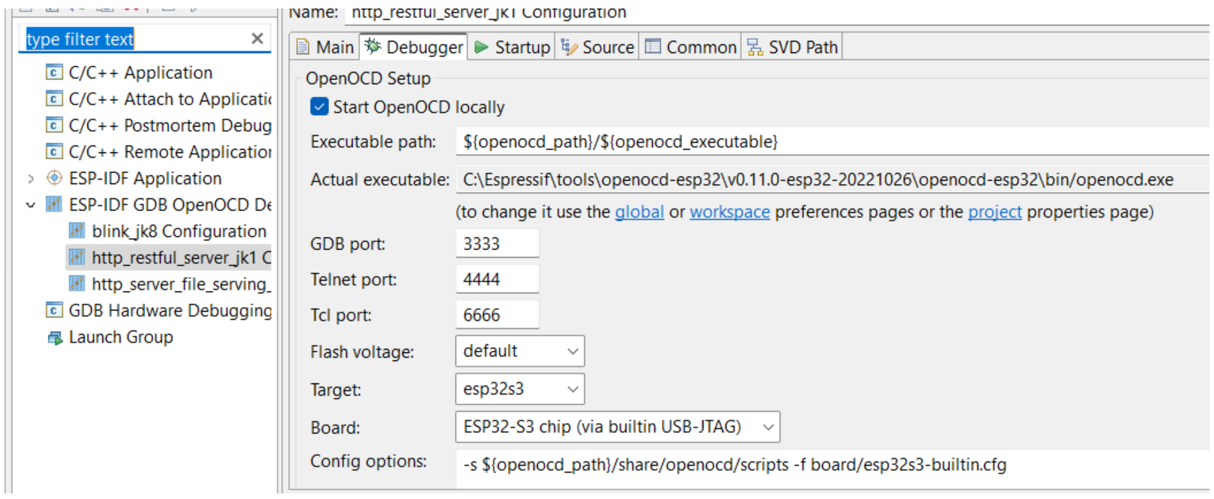
The module must be powered with 5V. Any one of two USB ports can supply the module. Also, we can use external 5V power connected to GPIO connectors pin marked as “5V in”. All three power inputs can be used independently and simultaneously, because this module contains 3 Schottky type diodes. Note1: one of these diodes can be shorted by using SMD solder joint. In such case pin “5V in” on GPIO connector becomes as “5V out”. Note2: official schematic of similar module on the site of Espressif does not contain that third diode, but contains 6 pieces of ESD protection diodes. Two pins on GPIO connector marked as “3V3” can supply external devices.

ESP-IDF (Espressif IoT Development Framework)

Development of our application, independently of tool in use, will be based on ESP-IDF (Espressif IoT Development Framework) : https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html#introduction Actual version now is “5”.

Development by using Espressif-IDE

By using Espressif-IDE, we can find a huge quantity of examples ready to review, modify and upload to our module from this IDE. Examples are located in the “examples” directory – as in following case: C:\Espressif\frameworks\esp-idf-v5.0\examples\protocols\http_server\restful_server Also, Espressif-IDE contains interface to debug with OpenOCD software together with JTAG adapter (integrated into the microcontroller or some external).



Name: http_restful_server_jk1 Configuration

OpenOCD Setup

☑ Start OpenOCD locally

Executable path: ${openocd_path}/${openocd_executable}

Actual executable: C:\Espressif\tools\openocd-esp32\v0.11.0-esp32-20221026\openocd-esp32\bin\openocd.exe

(to change it use the global or workspace preferences pages or the project properties page)

GDB port: 3333

Telnet port: 4444

Tcl port: 6666

Flash voltage: default

Target: esp32s3

Board: ESP32-S3 chip (via builtin USB-JTAG)

Config options: -s ${openocd_path}/share/openocd/scripts -f board/esp32s3-builtin.cfg

FIGURE 3 - CONFIGURATION OF DEBUG IN ESPRESSIF-IDE

To use Espressif-IDE, we need Java SDK installed into our Windows computer. Get it from https://www.oracle.com/java/technologies/downloads/#jdk19-windows

Also, it must be installed Python – to run various scripts.

Using ESP-IDF 5.0 PowerShell

Before editing, compiling, building, debugging, and uploading our project in Espressif-IDE, for each work session we must run this special tool and related script. Shortcut to this tool appears on our Windows desktop after installation of Espressif-IDE. The script configures required Windows environmental variables and opens the window of Windows Power Shell, which appears as configured in right way to run the command line tool idf.py with parameters. Firstly, in each shell session, we must navigate to our project folder (by using Windows command line commands). Note: project folder and all required scripts and configuration files appears after creating new project. One of required actions for new project is to run (being in the project folder) idf.py with parameter “menuconfig” (idf.py menuconfig). During this first session of new project we must enter, at least, Wi-Fi connection credentials and save them. Note: It is also possible to make configuration of our project from within of Project Explorer of Espressif-IDE – by double clicking on the file “sdkconfig”.

Development by using Arduino IDE

By using Arduino IDE, we can find a huge quantity of examples ready to review, modify and upload to our module from this IDE. Also, Arduino IDE contains interface to debug with debug software integrated in Arduino IDE together with JTAG adapter (integrated into the microcontroller or some external). In “Preferences->Additional boards manager URLs” we must enter “https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json ”. In the “Tools->Partition scheme” we must select memory variant matching to our module (our is “8R2”). We can upload and debug our module by using two options: 1) Virtual Serial port or 2) JTAG debug adapter integrated in the ESP32-S3 chip. Selection of these options we are doing by making selections in the “Tools-> Port”. Sure, USB cable/cables must be connected.

Unsuccess of development by using Platformio IDE

After a few hours of attempts, we can conclude that Platformio IDE does not contain any thing that is not presented in Arduino IDE. This is only additional layer on top of Arduino IDE or on top of Espressif-IDE. Painful and not useful.

It's built on top of Microsoft's Visual Studio Code – free, open source, and MIT licensed editor. Web sites containing information:

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/third-party-tools/platformio.html

https://platformio.org/platformio-ide?utm_source=docs.espressif.com

In Visual Studio Code go to Extensions and search for platformio ide:

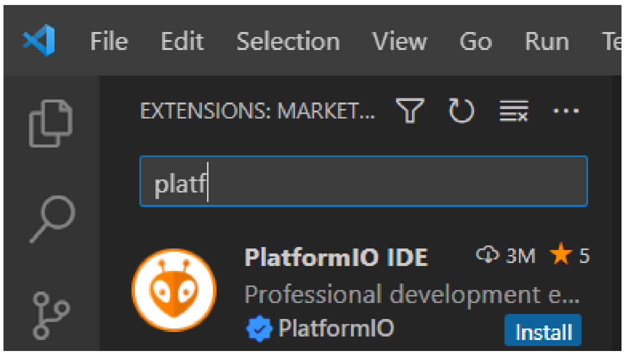


FIGURE 4 - IN VISUAL STUDIO EXTENSIONS MANAGER SEARCH FOR PLATFORMIO IDE

https://docs.platformio.org/en/latest/integration/ide/vscode.html#quick-start

Using of FreeRTOS

It seems that all examples in Espressif-IDE use preconfigured FreeRTOS. It seems that we don’t need special actions to allow and to configure the FreeRTOS (if we don’t need any special).

Using of bootloader

It seems that all examples in Espressif-IDE use preconfigured bootloader. It seems that we don’t need special actions to allow and to configure the bootloader (if we don’t need any special). Also, we will see, that each time we upload our project to chip, the bootloader is uploaded together.

Using of SPIFFS

SPIFFS is a file system well adopted to use with Flash memory. It seems that all examples, that requires files, in Espressif-IDE, by default, use preconfigured SPIFFS. It seems that we don’t need special actions to allow and to configure the SPIFFS (if we don’t need any special). But now, at least for us, it is not clear how to manage SPIFFS creation, deletion, filling with files in Espressif-IDE. If to use Arduino, this seems is clearer – 1) during compiling Arduino must know memory configuration of our module (“8R2” for our module) and, 2) during uploading or running of any of project, the Arduino does not touches partition and files inside the existing SPIFFS, 3) we need an another Arduino project from examples in case of necessity to manage SPIFFS creation, deletion, filling with files in Arduino, 4) sure, any of our Arduino projects may have functionality to manage SPIFFS.

Selection of module type in Arduino IDE

We will not find exact module matching to our in the list of boards of Arduino IDE. Use “ESP32-S3-USB-OTG”.

Blink LED example

On Arduino we will need the library: #include <Adafruit_NeoPixel.h> Works well. Similar example on Espressif-IDE knows which pin is connected to NeoPixel LED on the module. But the example on Arduino needs to edit pin number to #define PIN_NEOPIXEL 48.

WiFi Web Server LED Blink on Arduino

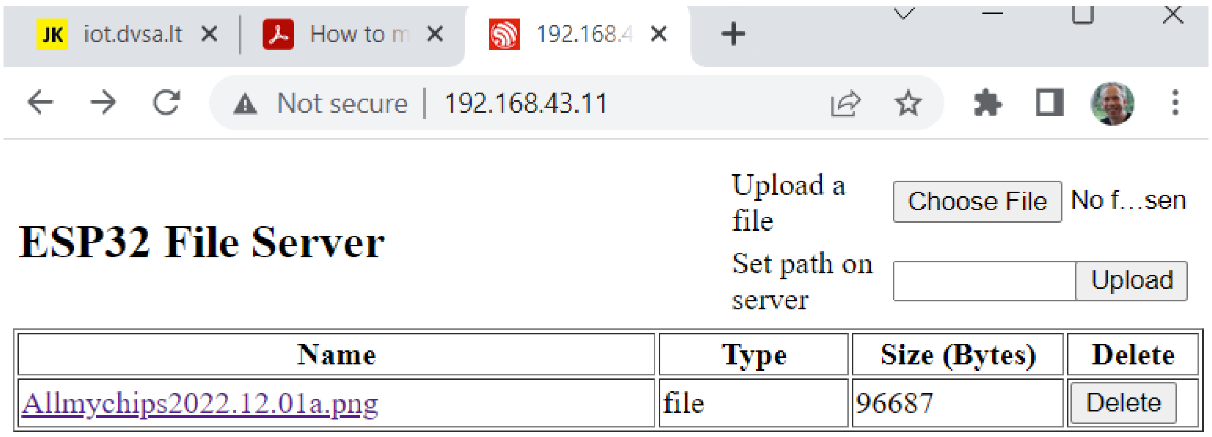
“Examples->WiFi->SimpleWiFiServer”. Works well. Very simple web interface is generated by code of the example.

Unsuccess of “HTTP Restful API Server Example” on Espressif-IDE

This complex example uses files for web presentation created with vue. But this example does not contain already generated files for web – we must prepare our environment to compile vue files. Not success to generate these files, vue compilation does not works, we need much work to find correct version of vue and related libraries. Attempt to use loaded by using other tools other web files instead of using project generated web content – not success. Without correct files, matching to server code, this example can be debugged only – in debugger we can look how our Rest interface reacts to HTTP requests. But, after restart, we have an endless restart loop.

HTTP File Server Example on Espressif-IDE

This example works fine. We can view web interface of this example application, upload files to SPIFFS, browse uploaded files. Also, we can observe, that during working with other application, files in SPIFFS, previously loaded by other application, will remain.


iot.dvsa.lt
How to m
192.168.4
Not secure | 192.168.43.11

ESP32 File Server

Upload a file [Choose File] No f…sen

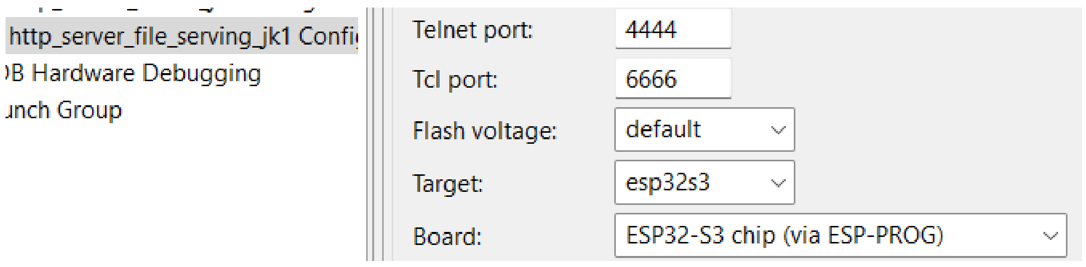
Set path on server [Upload]

Name	Type	Size (Bytes)	Delete
Allmychips2022.12.01a.png	file	96687	Delete

FIGURE 5 - BROWSER WINDOW SHOWING THIS APPLICATION WEB INTERFACE

Is a big advantage of using built-in JTAG adapter?

We can see that Virtual Serial port on Arduino IDE and on Espressif-IDE allows to make all required tasks.


http_server_file_serving_jk1 Confi
)B Hardware Debugging
ınch Group

Telnet port: 4444
Tcl port: 6666
Flash voltage: default
Target: esp32s3
Board: ESP32-S3 chip (via ESP-PROG)

Revision History

Version	Date	Comments
ver.1.1	2023.02.08	New chapters, various text improvements
ver.1.0	2023.02.02	Initial release