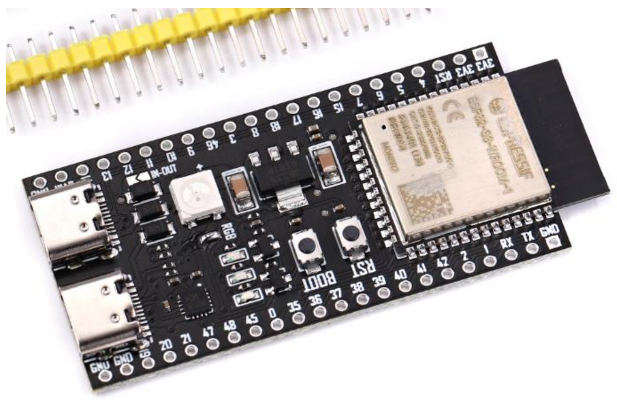

How to make a node of IoT by using ESP32-S3 DevKitC

Juozas Kimtys - How to make a node of IoT by using ESP32-S3 DevKitC – doc. ver.1.0

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:

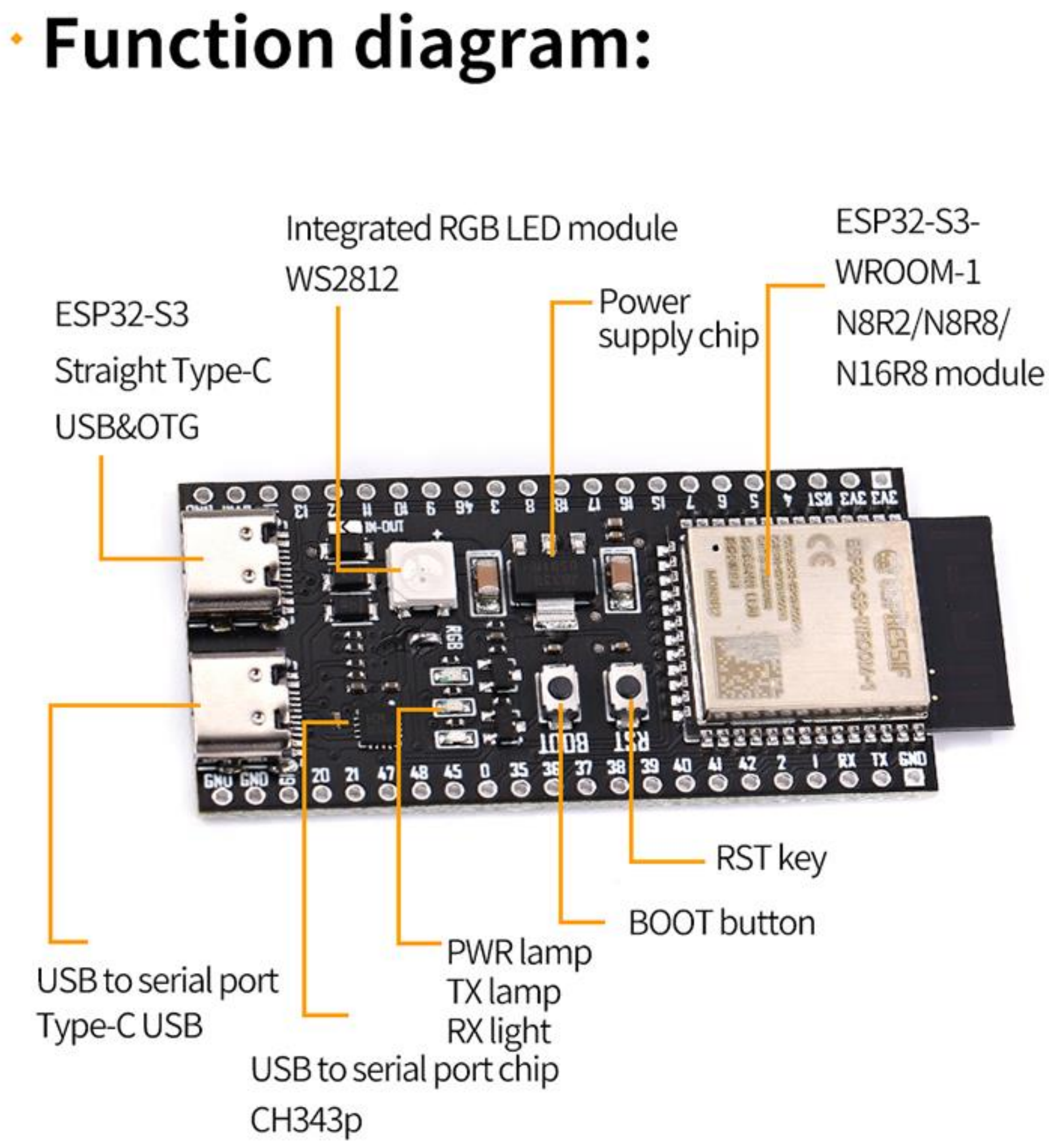
Function diagram:
Integrated RGB LED module WS2812
ESP32-S3-WROOM-1 N8R2/N8R8/ N16R8 module
ESP32-S3 Straight Type-C USB&OTG
Power supply chip
RST key
BOOT button
PWR lamp
TX lamp
RX light
USB to serial port Type-C USB
USB to serial port chip CH343p

FIGURE 1 - ESP32-S3 DEVKITC – FUNCTION DIAGRAM – PICTURE FROM THE ALIEXPRESS.

Two USB connectors available

There are two USB connectors available on the module. One, marked as “COM” (marked 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 other than TX and RX serial port lines. Second, marked as “USB” (marked 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.

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”.

Using Espressif-IDE to develop our applications

By using Espressif-IDE we can find a huge quantity of examples ready to review, modify and upload to our module from this IDE. Also, Espressif-IDE contains interface to debug with OpenOCD software together with JTAG adapter (integrated into the microcontroller or some external).

To use Espressif-IDE, we need Java SDK installed in to 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, 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 allows to run the command line tool idf.py with parameters. Firstly, in this 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 is to run (being in the project folder) idf.py with parameter “menuconfig” (idf.py menuconfig). During this session 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”.

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, 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 and, 2) during uploading or running of any of project, the Arduino do not touches partition and files inside the existing SPIFFS – we need a special project in case of necessity to manage SPIFFS creation, deletion, filling with files in Arduino, 3) sure, any of our projects may have functionality to manage SPIFFS from running Arduino project.

Blink LED example

Works well. The board contains NeoPixel RGB LED. The 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

Revision History

Version	Date	Comments
ver.1.0	2023.02.02	Initial release