WT32-SC01 plus
Getting Started Guide for AWS IoT Core

Table of Contents

1 Document information

1.1 Document revision history

Date	Modified by	Description
November 14, 2024	Vans	First release

1.2 Applicable operating systems for this guide

This guide is applicable to the ESP32S3 series chips with FreeRTOS system.

2 Overview

WT32-SC01 plus is a development board for visual touch screen, the board is equipped with GUI platform firmware developed independently, support graphic drag and drop programming to help users complete the development of customized control platform.

The main controller of this development board adopts WT32-S3-WROVER-N16R2 module, this module is a general-purpose Wi-Fi and BLE MCU module, 16MB SPI flash and 2MB PSRAM are configured in.

WT32-SC01 plus development board can also develop and debug functions such as buttons,

voice and camera through the expansion interfaces on both sides, greatly shorten the development cycle of users.

3 Hardware description

3.1 Datasheet

The link to the product datasheet: *https://en.wireless-tag.com/product-item-26.html*

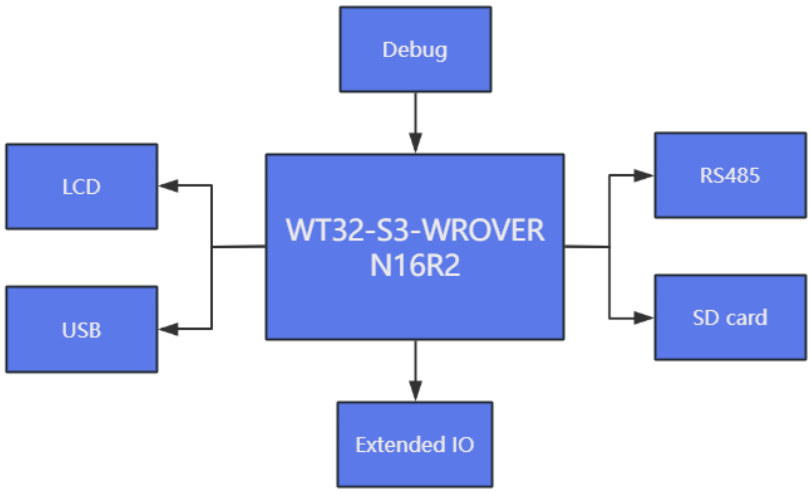
3.2 Standard kit contents

The contents of the standard shipping hardware package as indicated below:

- Module: WT32-S3-WROVER-N16R2, with 16MB flash onboard
- Memory: 512kB RAM, 384kB ROM, 2MB PSRAM
- Supports seamless switching between Arduino and MicroPython programming.
- Resolution: 480*320
- Power supply: 5V or 3.3V

Please links to the page on our company website for more detail:
https://en.wireless-tag.com/product-item-26.html

3.3 Additional hardware references



4 Set up your development environment

4.1 Tools installation (IDEs, Toolchains, SDKs)

1. IDE based
   - Eclipse Plugin
   - VSCode Extension
   - Arduino

2. idf.py

The idf.py command-line tool provides a front-end for easily managing your project builds, deployment and debugging, and more. It manages several tools, for example:

- CMake, which configures the project to be built.
- Ninja, which builds the project.
- Esptool.py, which flashes the target.

Can read more about configuring the build system using idf.py here.

3. ESP-IDF

For the manual procedure, please select according to your operating system.

- Windows Installer
- Linux and macOS

4. Optimizing the compiler

In ESP-IDF, you can force compiler optimization by modifying the compiler options. You can set the compiler optimization options in the CMakeLists.txt file.

In your project's CMakeLists.txt file, add the following lines to set the compiler optimization options

```
set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -O3")
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -O3")
```

5. SDK

Esp-idf , A tutorial on setting up the sdk environment can be found in the Getting Started guide. ESP-IDF is Espressif's official IoT Development Framework for the ESP32, ESP32-S, ESP32-C and ESP32-H series of SoCs.

It provides a self-sufficient SDK for any generic application development on these platforms, using programming languages such as C and C++.

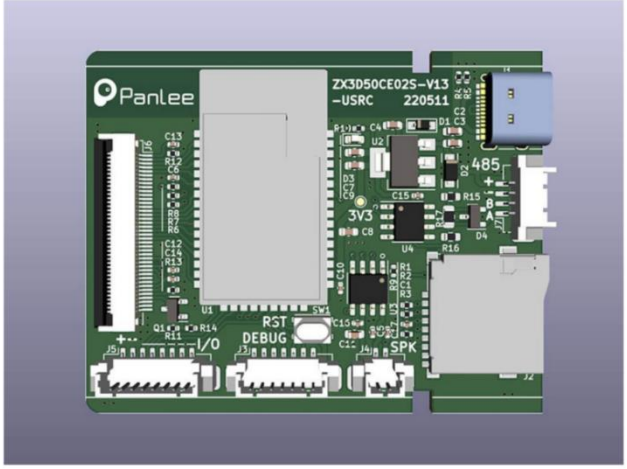
4.2 Additional software references

ESP32S3 Chip Manual: Provides specifications for the ESP32S3 chip.

ESP-IDF Programming Guide for ESP32S3: Extensive documentation for the ESP-IDF development framework.

5 Set up device hardware

5.1 Product Images



5.2 Interface Description

Pin	Description	Module Pin	Voltage Range	Remark
1	+5V	-	5V	
2	+3.3V	-	3.3V	For reference, not for power in put
3	ESP_TXD	TXD0	3.3V TTL	
4	ESP_RXD	RXD0	3.3V TTL	
5	EN	EN	0-3.3V	Chip enable
6	BOOT	GPIO 0	0-3.3V	
7	GND	GND	0V	GROUND

table 1 Debug Interface

Pin	Description	Module Pin	Voltage Range	Remark
1	+5V	-	5V±5%	Power supply or output voltage
2	GND	-	0V	Ground
3	EXT_IO1	GPIO10	0-3.3V	Extended IO
4	EXT_IO2	GPIO11	0-3.3V	
5	EXT_IO3	GPIO12	0-3.3V	
6	EXT_IO4	GPIO13	0-3.3V	
7	EXT_IO5	GPIO14	0-3.3V	
8	EXT_IO6	GPIO21	0-3.3V	

table 2 Extended IO Interface

Description	Module Pin	Remark
SD_CS	GPIO41	SD card chip selection
SD_DI (MOSI)	GPIO40	SD card data input
SD_CLK	GPIO39	SD card clock
SD_DO (MISO)	GPIO38	SD card data output

table 3 SD Card Interface

Pin	Description	Remark
1	SPK+	Speaker positive
2	SPK-	Speaker negative

table 4 Speaker Connector

Description	Module Pin	Remark
BL_PWM	GPIO45	Backlight control, active high
LCD_RESET	GPIO4	LCD reset, multiplexed with touch reset
LCD_RS	GPIO0	Command/Data selection
LCD_WR	GPIO47	Write clock
LCD_TE	GPIO48	Frame sync
LCD_DB0	GPIO9	LCD data interface, 8bit MCU (8080)
LCD_DB1	GPIO46	
LCD_DB2	GPIO3	
LCD_DB3	GPIO8	
LCD_DB4	GPIO18	
LCD_DB5	GPIO17	LCD data interface, 8bit MCU (8080)
LCD_DB6	GPIO16	
LCD_DB7	GPIO15	
TP_INT	GPIO7	Touch interrupt
TP_SDA	GPIO6	Touch IIC data
TP_SCL	GPIO5	Touch IIC clock
TP_RST	GPIO4	Touch reset, multiplexed with LCD reset

table 5 LCD Interface

Pin	Description	Remark
1	RS485-A	RS485 bus
2	RS485-B	
3	GND	Ground
4	+5V	Power supply or output voltage

table 6 RS485 Interface

5.3 Electrical Parameters

Parameter	Test conditions	Min	Typical	Max	Unit
Operating Voltage	-	4.7	5.0	5.5	V
Operating current	USB provides 5V power, with maximum backlight brightness	170	175	190	mA

5.4 Boot Configurations

The chip can be configured with the following startup parameters via the Strapping pin at power-up or hardware reset.

Strapping pin: GPIO0 and GPIO2

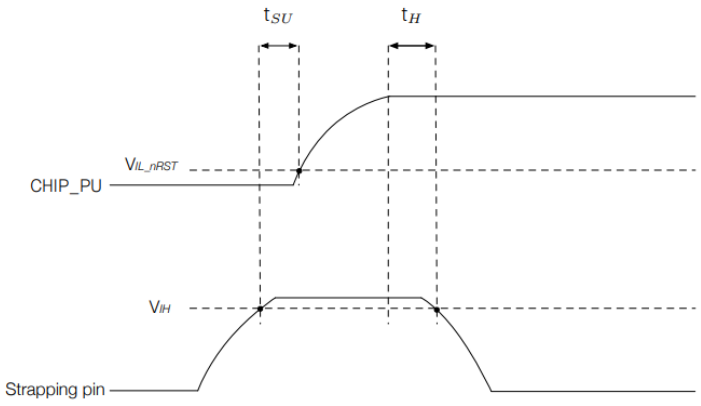
Strapping Pin	Default Configuration	Bit Value
GPIO0	Weak Pull-up	1
GPIO46	Weak Pull-down	0

Default Configuration of Strapping Pins

The timing of signals connected to the strapping pins should adhere to the setup time and hold time specifications in Table and Figure.

Parameter	Description	Min (ms)
t_{SU}	Setup time is the time reserved for the power rails to stabilize before the CHIP_PU pin is pulled high to activate the chip	0
t_H	Hold time is the time reserved for the chip to read the strapping pin values after CHIP_PU is already high and before these pins start operating as regular IO pins.	3

Description of Timing Parameters for the Strapping Pins



Visualization of Timing Parameters for the Strapping Pins

Boot Mode	GPIO0	GPIO46
SPI Boot Mode	**1**	**Any value**
Joint Download Boot Mode	0	0

Chip Boot Mode Control

Bold marks the default value and configuration.

5.5 LED Instructions

- Green light next to Type-C port: power indicator, power on normal, the light is on;
- RGB light: press and hold the BOOT button to power on, the RGB light shows green to indicate that it enters boot mode.

6 Setup your AWS account and permissions

If you do not have an existing AWS account and user, refer to the online AWS documentation at Set up your AWS Account. To get started, follow the steps outlined in the sections below:

- Sign up for an AWS account
- Create an administrative user
- Open the AWS IoT console

Pay special attention to the Notes.

7 Create resources in AWS IoT

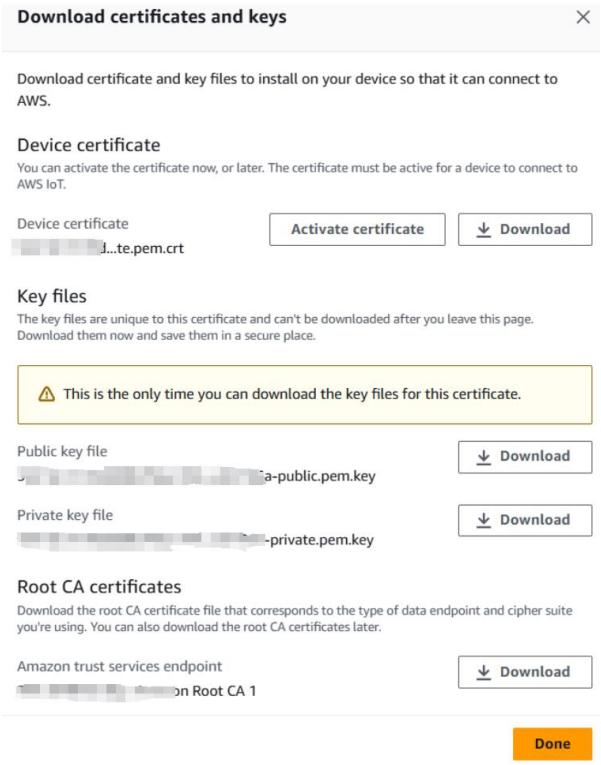
Refer to the online AWS documentation at Create AWS IoT Resources. Follow the steps outlined in these sections to provision resources for your device:

- Create an AWS IoT Policy
- Create a thing object

Pay special attention to the Notes.

8 Provision the device with credentials

During “Create a thing object”, you will encounter the requirement to download the certificate in the last step, as shown below, keep it, which is the corresponding server certificate of the device.



9 Build the demo

This tutorial uses the esp-idf/example/protocol/mqtt/ssl_mutual_auth example to test the device's connection to AWS-IoT-Core.

9.1 Engineering Configuration

1. After entering the project, you need to replace the three certificates in the main directory. The certificates to be replaced are stored in the connection toolkit you downloaded earlier. The replacement corresponds to the following:
   - The .client.crt file is the client certificate; use the .pem.crt file instead.
   - The .client.key file is the client key; use the .private.pem.key file instead.
   - The .mosquitto.org.crt file is the server-side secret key; use the CA1.pem file instead.
2. Replace the link to the mqtt server accessed by the project and add the client_id configuration entry.The link is replaced with the link used when connecting to the device, and the client_id used “basicPubSub”.
   Note: The link needs to be prefixed with mqtt://
3. Activate the IDF environment, configure the chip as ESP32S3 and modify the WiFi configuration information of the project via menuconfig.
   Note: Configuration path for WiFi configuration: Connection Configuration Example ->WiFi SSID / WiFi Password

10 Run the demo

Take the project introduced in the previous chapter, burn it into your device, and open the serial port debugging assistant.

11 Verify messages in AWS IoT Core

Open “MQTT test cilent”, set the Subscribe topic to “sdk/test/python” and click “Subscribe”. button.

12 Troubleshooting

If the connection fails when connecting to mqtt, it is recommended to check whether the above steps have been completed, whether the client_id and uri have been correctly modified, and whether the certificate has been correctly replaced.
If there is something you don't understand, you can also refer here.