WT32-SC01

Getting Started Guide for AWS IoT Core

Table of Contents

1 Document information

1.1 Document revision history

Date	Modified by	Description
November 15, 2024	Vans	First release

1.2 Applicable operating systems for this guide

This guide is applicable to the ESP32 series chips with FreeRTOS system.

2 Overview

WT32-SC01 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 WT32-SC01 development board adopts esp32-wrover-b module, this module is a general-purpose WiFi + BT + BLE MCU module, 16MB SPI flash and 8MB PSRAM are configured in. WT32-SC01 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.

3 Hardware description

3.1 Datasheet

The link to the product datasheet: *https://en.wireless-tag.com/product-item-25.html*

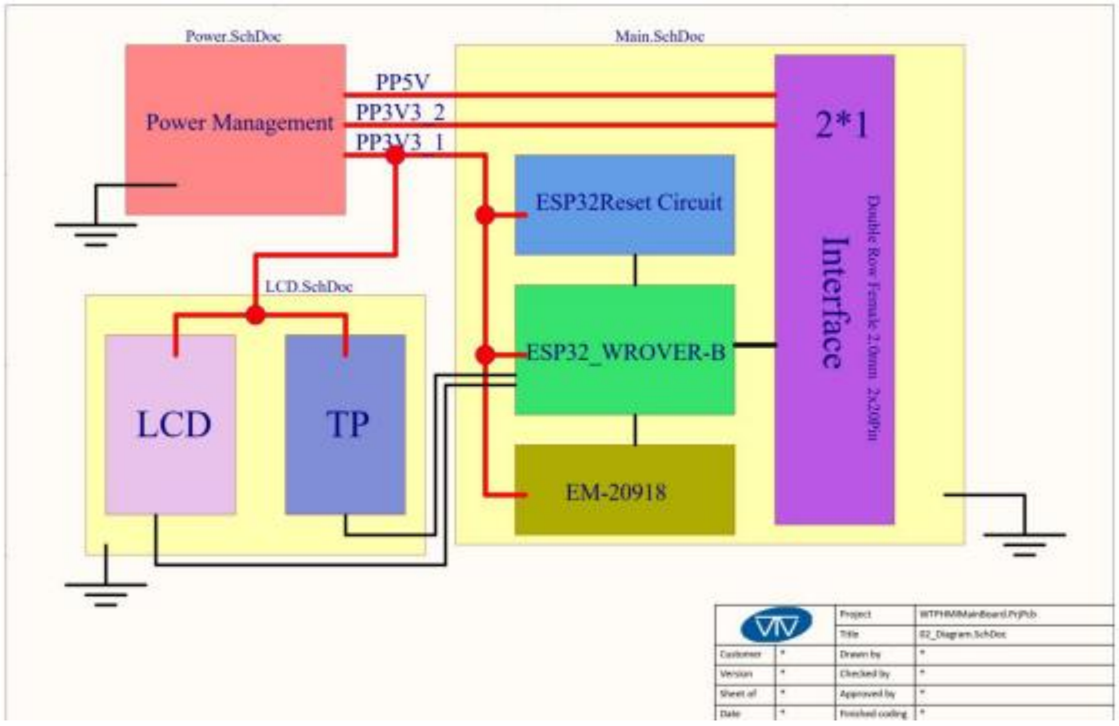
3.2 Standard kit contents

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

- Module: ESP32-WROVER-B, with 32Mbit flash onboard
- Memory: 8MB PSRAM, 448KB ROM, 520KB SRAM
- Resolution: 320*480
- LCM Display interface: 3.5-inch, SPI interface, 0.5mm spacing, 24pin
- Type-C Interface: to power the development board, UART communication and firmware download

Please links to the page on our company website for more detail:
https://en.wireless-tag.com/product-item-25.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

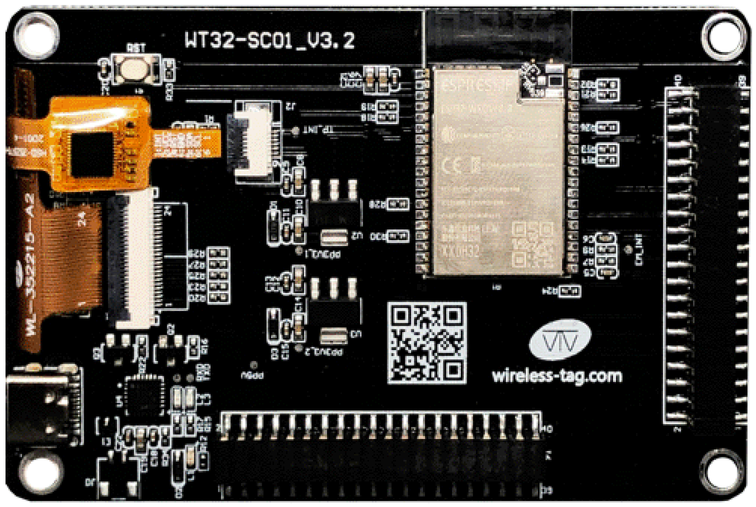
Esp-idf , A tutorial on setting up the sdk environment can be found in the Getting Started guide.
ESP-IDF is Espressif's official IoT Development Framework for the ESP32, ESP32-S, ESP32-C and ESP32-H series of SoCs.
It provides a self-sufficient SDK for any generic application development on these platforms, using programming languages such as C and C++.

4.2 Additional software references

ESP32 Chip Manual: Provides specifications for the ESP32 chip.
ESP-IDF Programming Guide for ESP32: Extensive documentation for the ESP-IDF development framework.

5 Set up device hardware

5.1 Product Images



5.2 Pin Description

Pin	Name	Description
1	EN	Module enable signal.Active high.
2	3V3	3V3 Power supply
3	GND	Ground
4	SENSOR_VP	GPIO36,ADC1_CH0
5	SENSOR_VN	GPIO39,ADC1_CH3
7	IO34	GPIO34,ADC1_CH6
8	IO35	GPIO35,ADC1_CH7,I2S_ASDOUT
9	IO32	GPIO32,XTAL_32K_P,ADC1_CH4
10	IO33	GPIO33,XTAL_32K_N,ADC1_CH5
11	IO25	GPIO25,ADC2_CH8,I2S_LRCK
12	IO26	GPIO26,ADC2_CH9,EMAC_RXD1,I2S_DSDIN

13	IO27	GPIO27,ADC2_CH7
14	IO14	GPIO14,ADC2_CH6
15	IO12	GPIO12,ADC2_CH5
16	IO5	GPIO5,I2S_SCLK
17	IO0	GPIO0,ADC2_CH1,I2S_MCLK
18	IO13	GPIO13,ADC2_CH4
19	IO15	GPIO15,ADC2_CH3
20	IO21	GPIO21
21	IO22	GPIO22
22	IO23	GPIO23
23	IO18	GPIO18,I2C_SDA
24	IO19	GPIO19,I2C_SCL
25	IO1	GPIO1,U0TXD
26	IO3	GPIO3,U0RXD

Table 1 : IO Descriptions

5.3 Power Supply

This development board supports USB Type-C 5V power supply and reserved external power input interface. It is recommended that the input voltage 5V support current is not less than 1A when other expansion boards are not inserted, and the input voltage 5V support current is not less than 2A when other expansion boards are inserted (refer to the actual power consumption of the expansion board for details). Note: When the external power supply is provided through the reserved power interface, the power supply voltage input range is 5V-9V, and the load current is recommended to be I> 2A.

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	Pull-up	1
GPIO2	Pull-down	0

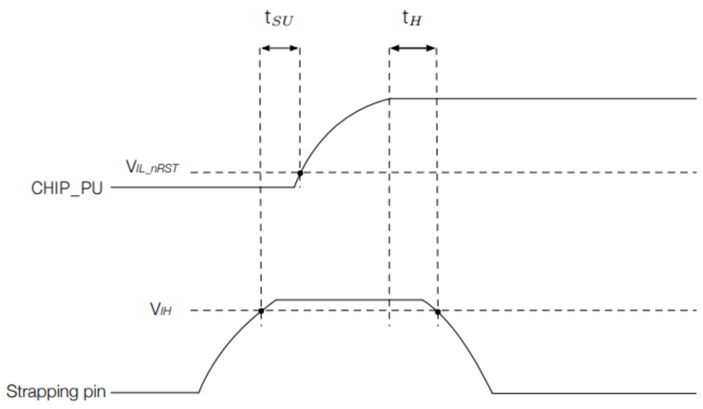
Default Configuration of Strapping Pins

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	GPIO2
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

- LED1: The power indicator (red light) lights up when you plug in the USB cable;
- LED2: The TXD indicator and RXD indicator in the UART will flash when there is data flow.

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

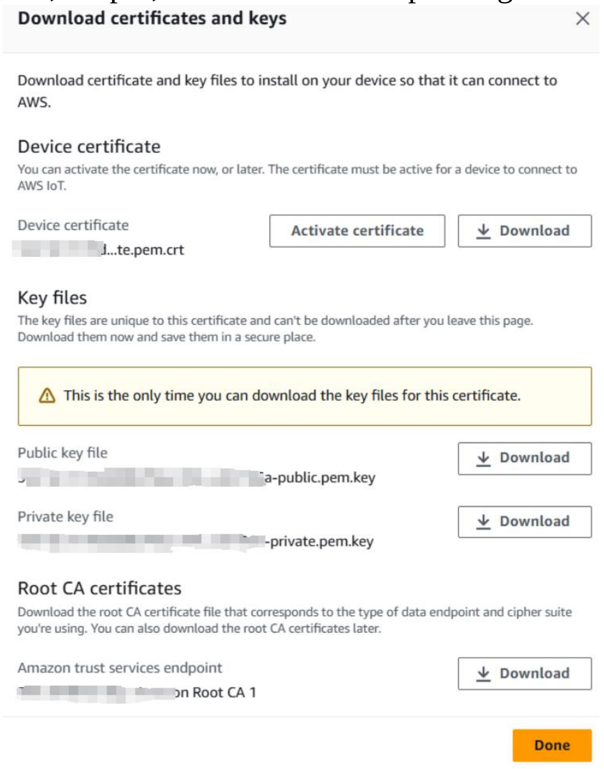
Refer to the online AWS documentation at Create AWS IoT Resources. Follow the steps outlined in these sections to provision resources for your device:

- [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.



Download certificates and keys

Download certificate and key files to install on your device so that it can connect to AWS.

Device certificate

You can activate the certificate now, or later. The certificate must be active for a device to connect to AWS IoT.

Device certificate
...te.pem.crt

Key files

The key files are unique to this certificate and can't be downloaded after you leave this page. Download them now and save them in a secure place.

This is the only time you can download the key files for this certificate.

Public key file
-public.pem.key

Private key file
-private.pem.key

Root CA certificates

Download the root CA certificate file that corresponds to the type of data endpoint and cipher suite you're using. You can also download the root CA certificates later.

Amazon trust services endpoint
n Root CA 1

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 ESP32 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.