

# ESP8266 EdgerOS SDDC 开发环境搭建

ESP8266 是乐鑫科技研发的一款面向物联网应用的高性价比、高度集成的 Wi-Fi MCU, 乐鑫科技对外提供了 ESP8266 的二次开发 SDK, 通过在 ESP8266 SDK 中集成 EdgerOS 的 SDDC 协议源代码, 实现 ESP8266 与搭载 EdgerOS 的智能边缘计算机 Spirit 通信。文本介绍如何搭建 ESP8266 EdgerOS SDDC 开发环境。

## 一、安装安信可 IDE

### 1.1 下载 IDE 安装包

<https://docs.ai-thinker.com/>

#### 热点板块

- 1.ESP8266开发指导（1.5版本 IDE资源及使用） [网盘链接](#)，提取码：shm3
- 2.ESP8266 AT固件开发使用指导（接线，透传）：[https://docs.ai-thinker.com/esp8266/examples/at\\_demo](https://docs.ai-thinker.com/esp8266/examples/at_demo)
- 3.技术博文分享总站：[链接](#)
- 4.新品发布资料（微信搜索公众号“安信可科技”）

下载 AiThinkerIDE\_V1.5.2.exe 即可。

### 1.2 安装 IDE

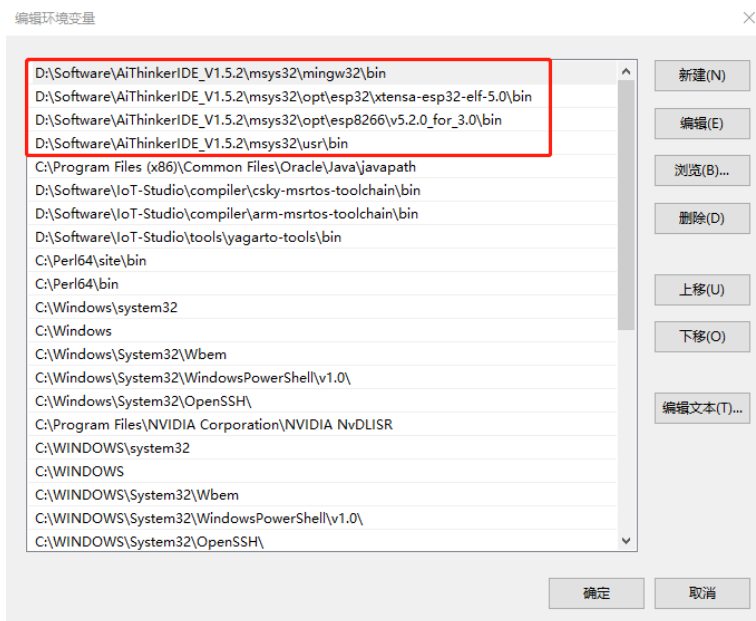
双击 AiThinkerIDE\_V1.5.2.exe，完成安装：

电脑 > Data (D:) > Installer > 安信可一体化IDE

名称	修改日期	类型	大小
esp8266 windows工具链	2021/3/17 11:30	文件夹	
AiThinkerIDE_V1.5.2.exe	2021/3/17 11:34	应用程序	609,651 KB
Git-2.26.0-64-bit.exe	2021/3/17 11:34	应用程序	45,576 KB
plugin_cygwin.zip	2021/3/17 11:30	WinRAR ZIP 压缩...	342,442 KB

### 1.3 调整 IDE PATH

将 AiThinkerIDE 相关的 4 个环境变量上移到最前面：



## 二、下载 ESP8266 SDK

在一个路径不带空格的目录下，执行以下命令下载：

```
git clone -recursive https://github.com/ms-rtos/AiThinkerProjectForESP.git
```

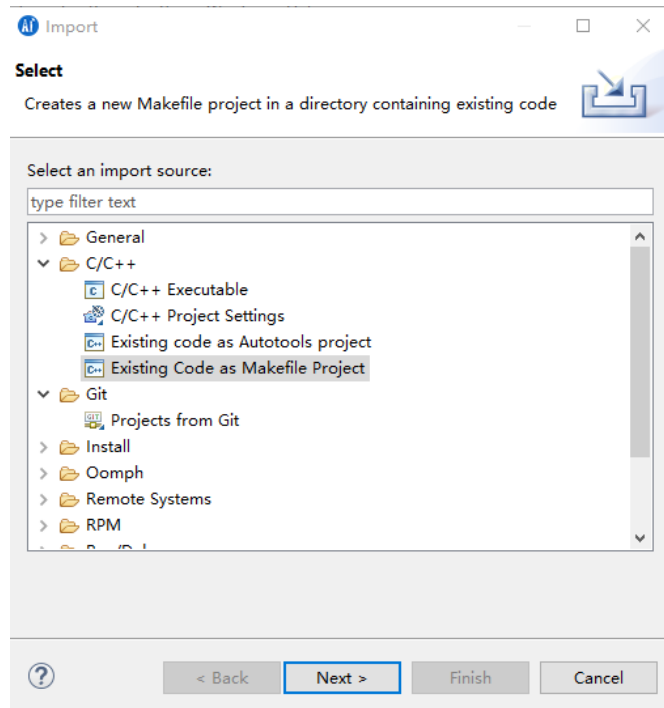
如果 github 下载速度慢，可以试试 gitee：

```
git clone -recursive https://gitee.com/ms-rtos/AiThinkerProjectForESP.git
```

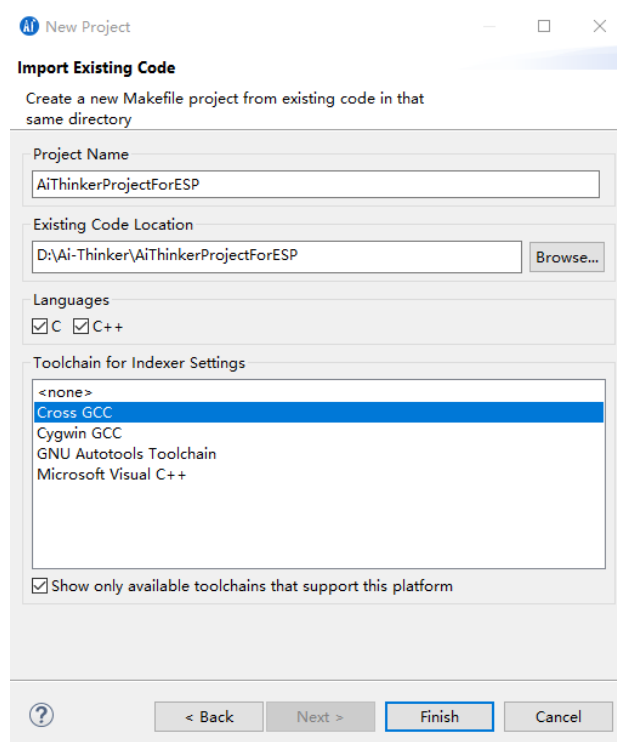
该 ESP8266 SDK 为翼辉信息工程师在安信可科技开源项目 AiThinkerProjectForESP 的基础上，二次开发而来，特此鸣谢安信可科技。

## 三、导入 ESP8266 SDK

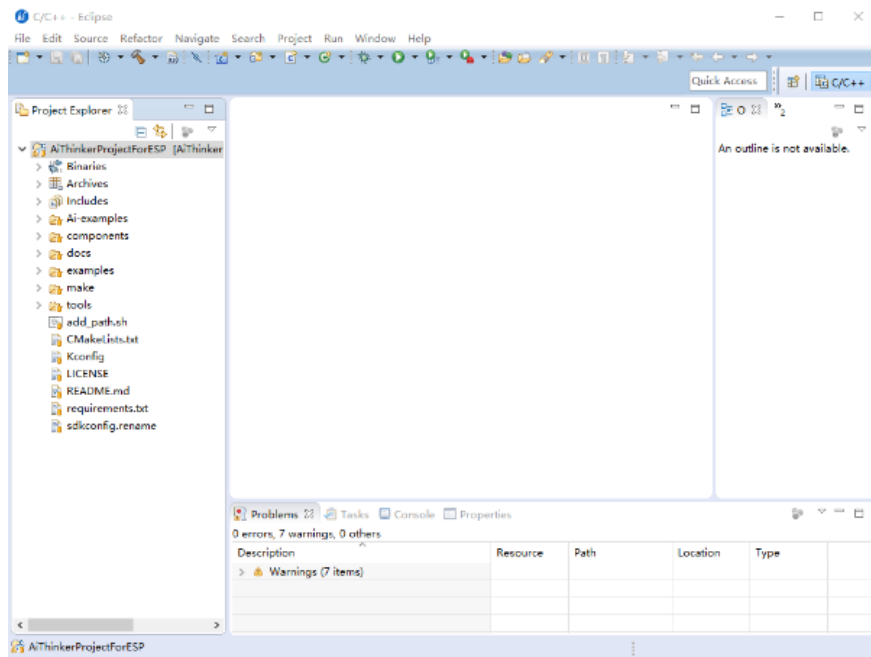
双击桌面的 AiThinkerIDE\_V1.5.2 图标，打开 AiThinkerIDE，点击“File”->“Import”菜单，将弹出“Import”向导：



选择"C/C++"下的"Existing code as Makefile Project"类型，然后点击"Next"按钮：



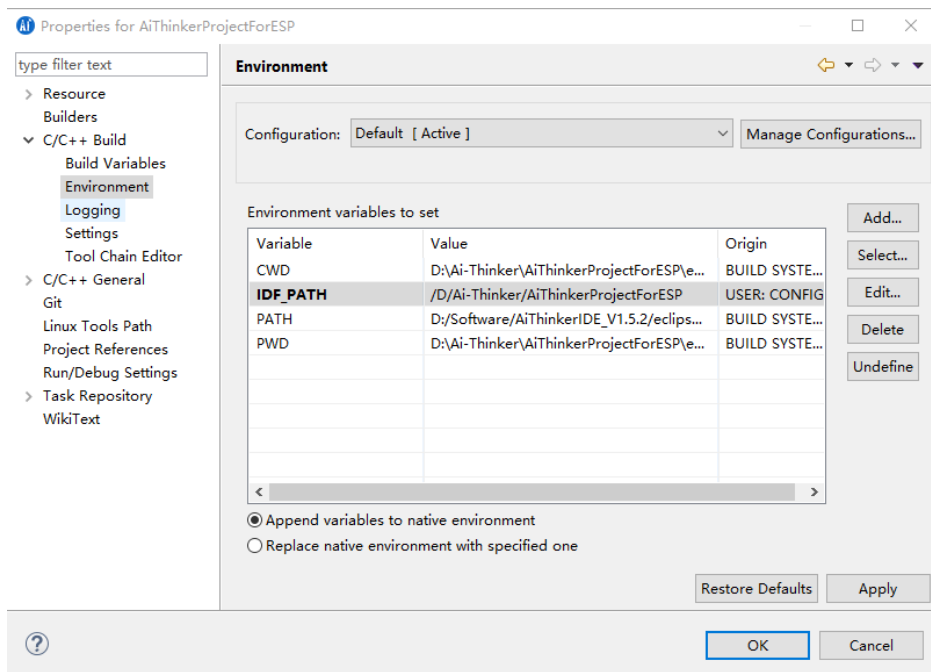
点击"Browse"按钮选择刚刚下载的 ESP8266 SDK 的路径，"Toolchain for indexer Settings" 选择"Cross GCC"，最后点击"Finish"按钮完成导入：

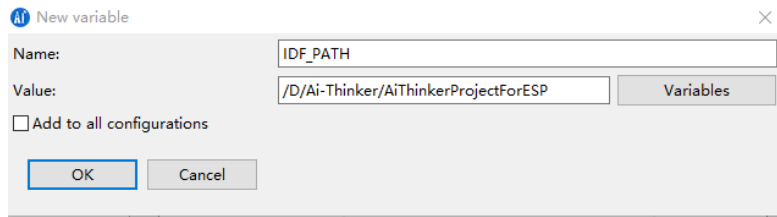


## 四、配置项目

### 4.1 配置 IDF\_PATH 变量

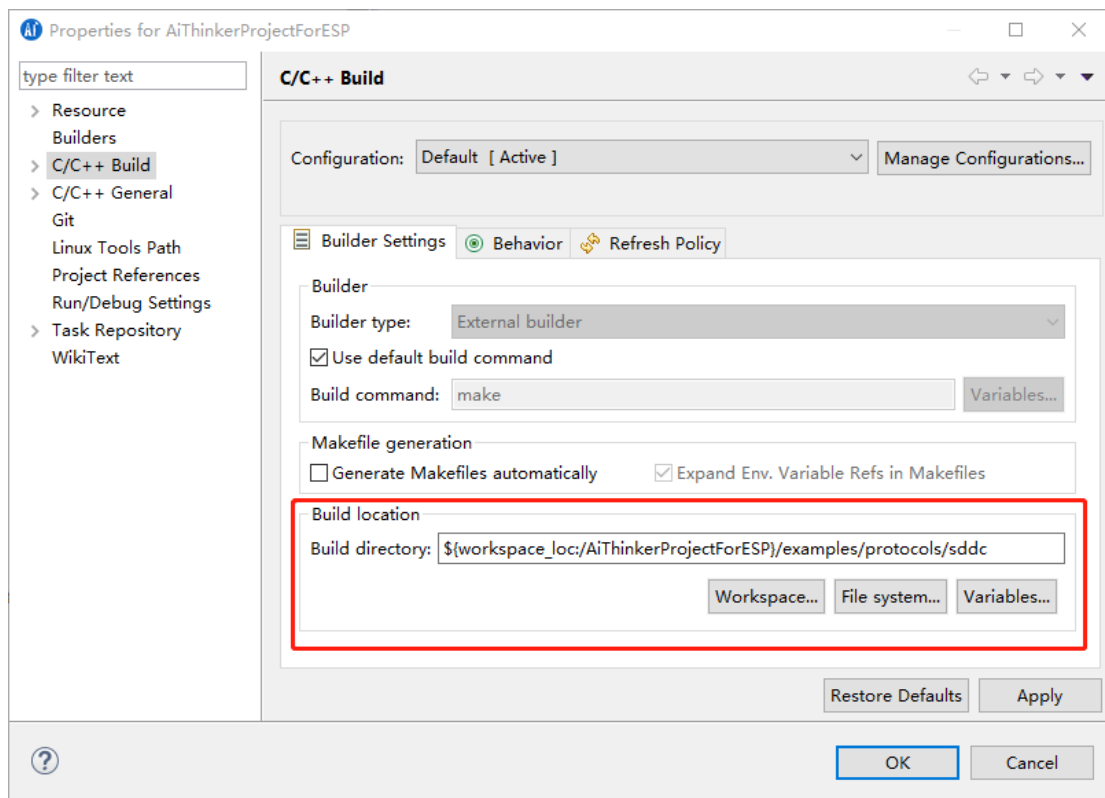
鼠标选中项目名称点击右键，在右键菜单中选择“Properties”，选择“C/C++ Build”->“Environment”，点击“Add”按钮添加一个“IDF\_PATH”变量，值为 ESP8266 SDK 的路径，注意需要将路径修改为 unix 风格，最后点击“OK”按钮：





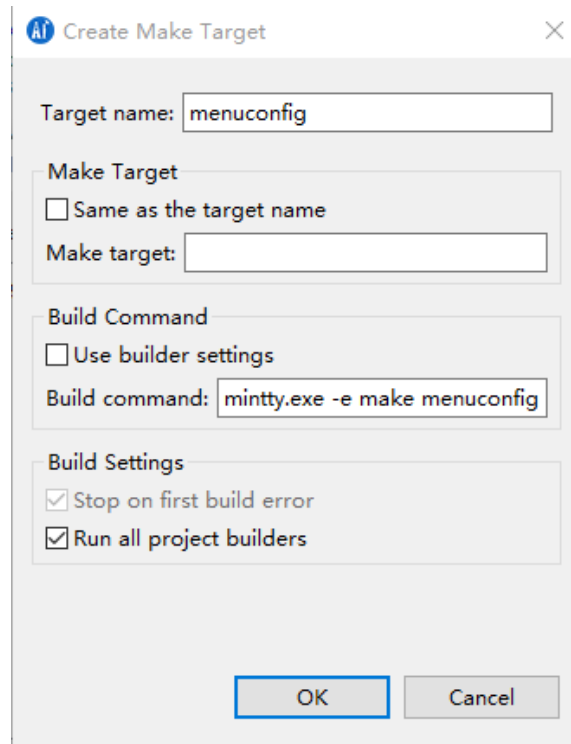
## 4.2 配置构建路径

鼠标选中项目名称点击右键，在右键菜单中选择"Properties"，选择"C/C++ Build"，修改"Build directory"为需要编译的例程路径，如"\${workspace\_loc:/AiThinkerProjectForESP}/examples/protocols/sddc"，最后点击"OK"按钮：



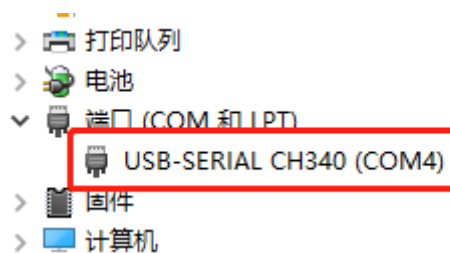
## 4.3 配置 menuconfig 目标

鼠标选中项目名称点击右键，在右键菜单中选择"Make Targets"->"Create",在弹出的对话框中取消勾选"Same as the target name" 与 "User builder settings" 这 2 个选项，将"Target name" 修改为 "menuconfig"， 并且在 "Build command" 中输入 "mintty.exe -e make menuconfig"，最后点击"OK"按钮：

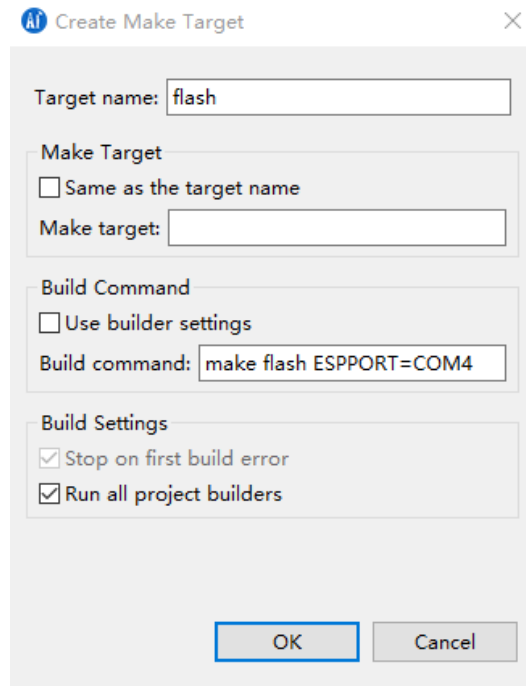


#### 4.4 配置 FLASH 编程目标

将 NodeMCU 开发板接入电脑, 打开设备管理, 查看操作系统为 NodeMCU 分配的 COM 号。



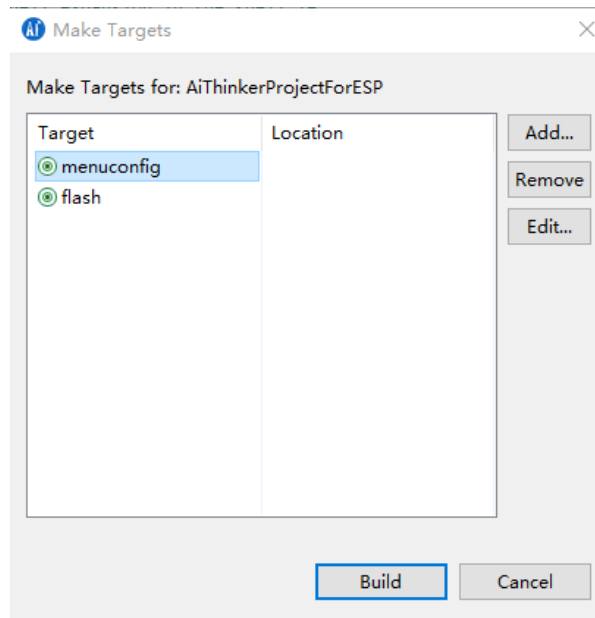
鼠标选中项目名称点击右键, 在右键菜单中选择"Make Targets" -> "Create", 在弹出的对话框中取消勾选"Same as the target name" 与 "User builder settings" 这 2 个选项, 将"Target name"修改为"flash", 并且在"Build command" 中输入"make flash ESPPORT=COM4", 最后点击"OK"按钮, 注意 COM 号为 NodeMCU 插入电脑后分配出来的 COM 号:

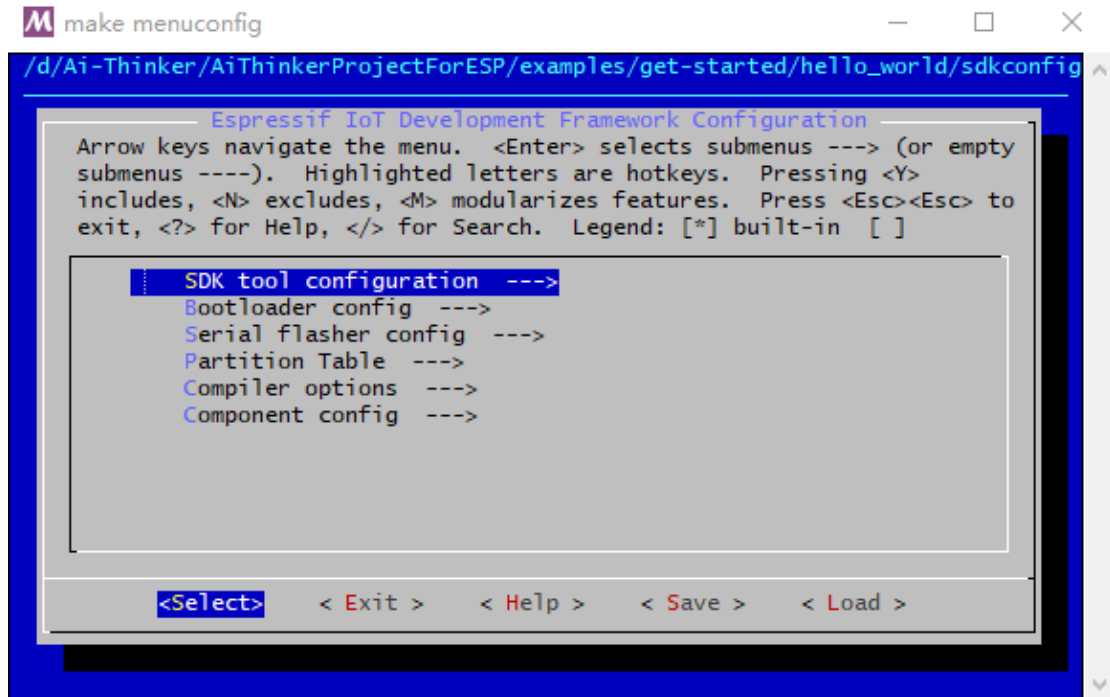


## 五、编译 SDK 和例程

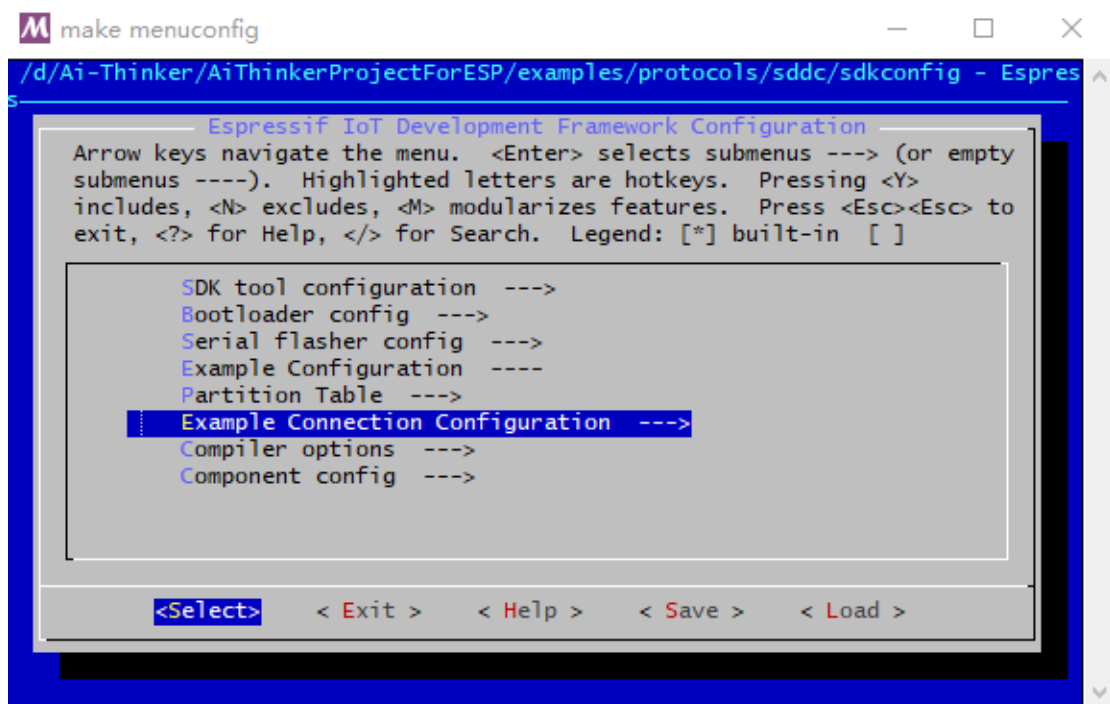
### 5.1 配置 SDK 和例程

鼠标选中项目名称点击右键，在右键菜单中选择”Make Targets”-> “Build”,在弹出的对话框中选中”menuconfig”项，然后点击”Build”按钮：



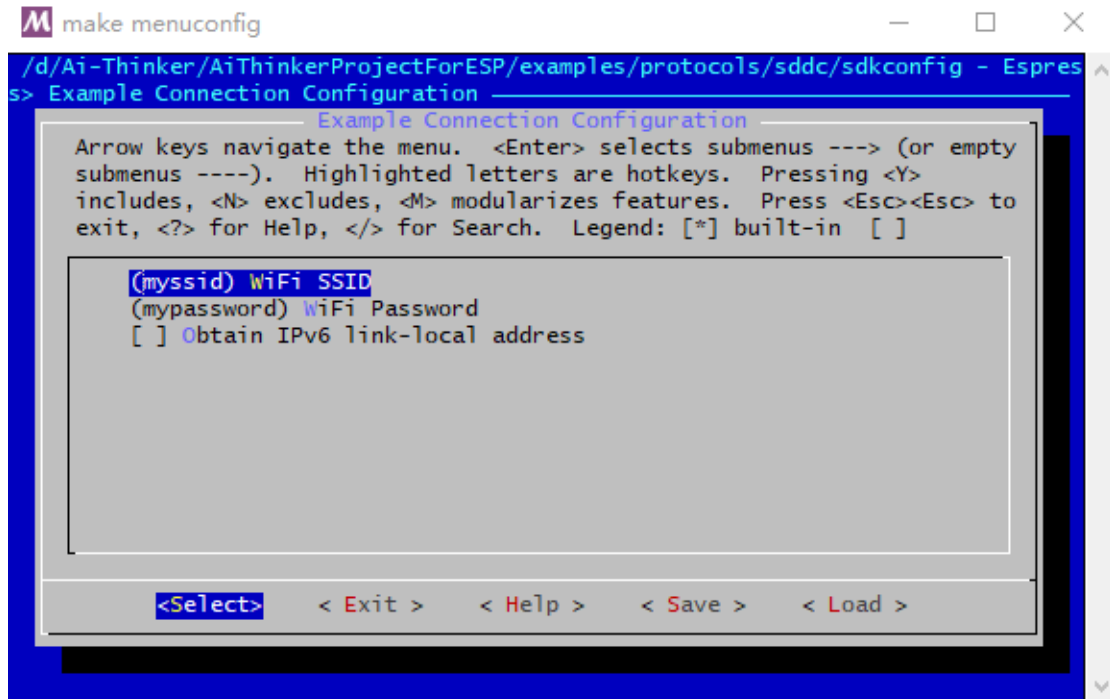


需要修改连接 WiFi AP 的配置：



将 WiFi SSID 和 Password 修改为智能边缘计算机 Spirit1 的 WiFi SSID 和 Password：





使用键盘操作，最后选中 Exit 回车后退出定制，等待配置完成。

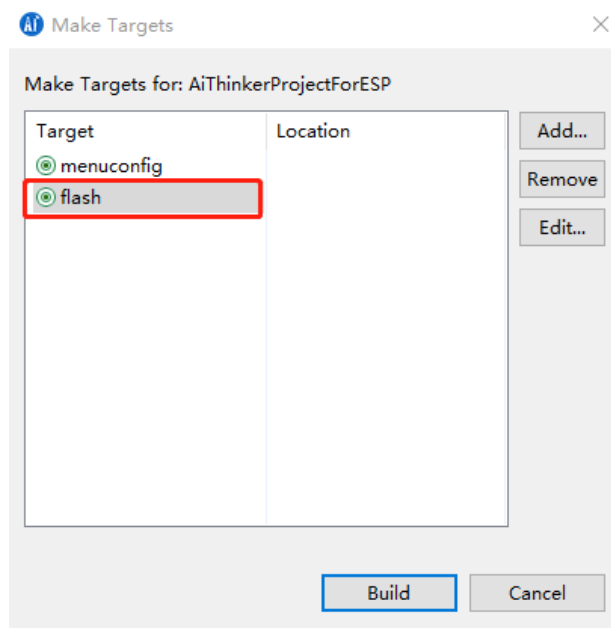
## 5.2 编译 SDK 和例程

鼠标选中项目名称点击右键，在右键菜单中选择“Build project”，等待编译完成。

# 六、烧写镜像和验证

## 6.1 烧写镜像

鼠标选中项目名称点击右键，在右键菜单中选择“Make Targets”->“Build”，在弹出的对话框中选中“flash”项，然后点击“Build”按钮：



```
Problems Tasks Console Properties
CDT Build Console [AiThinkerProjectForESP]
13:48:30 **** Build of configuration Default for project AiThinkerProjectForESP ****
make flash ESPPORT=COM4
-----
----- Welcome To AiThinker IDE V1.5 -----
----- Git Commit: a6fb921b38d6a2da582b2a1dd54acdcac66095f7 -----
----- SDK Version: v3.2-442-ga6fb921b-dirty -----
-----
-----
Toolchain version: crosstool-ng-1.22.0-100-ge567ec7b
Compiler version: 5.2.0
Python requirements from D:/Ai-Thinker/AiThinkerProjectForESP\requirements.txt are satisfied.
App "hello-world" version: e5ec3b46-dirty
Flashing binaries to serial port COM4 (app at offset 0x10000)...
esptool.py v2.4.0
Connecting...
Chip is ESP8266EX
Features: WiFi
MAC: 
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Compressed 10768 bytes to 7135...

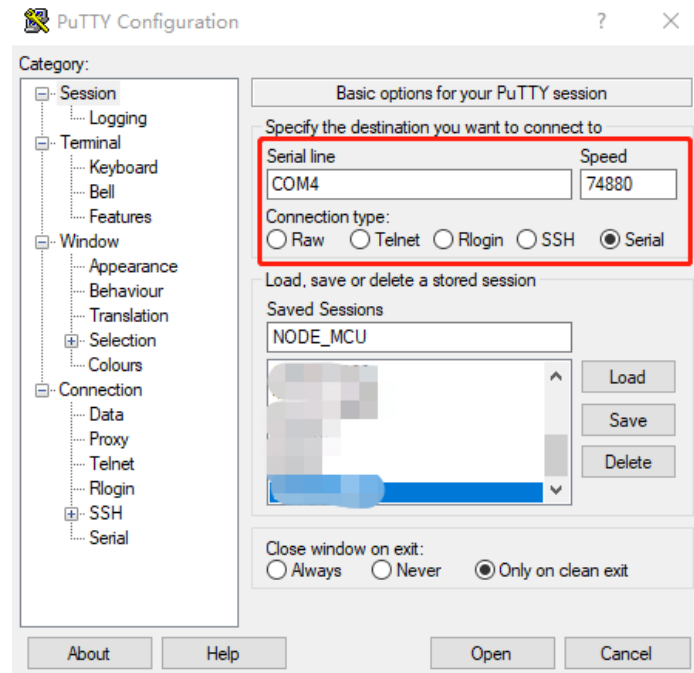
Writing at 0x00000000... (100 %)
Wrote 10768 bytes (7135 compressed) at 0x00000000 in 0.6 seconds (effective 136.3 kbit/s)...
Hash of data verified.
Compressed 195968 bytes to 133017...

Writing at 0x00010000... (11 %)
Writing at 0x00014000... (22 %)
Writing at 0x00018000... (33 %)
Writing at 0x0001c000... (44 %)
Writing at 0x00020000... (55 %)
Writing at 0x00024000... (66 %)
Writing at 0x00028000... (77 %)
Writing at 0x0002c000... (88 %)
Writing at 0x00030000... (100 %)
Wrote 195968 bytes (133017 compressed) at 0x00010000 in 11.7 seconds (effective 133.7 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 83...

Writing at 0x00008000... (100 %)
Wrote 3072 bytes (83 compressed) at 0x00008000 in 0.0 seconds (effective 1895.6 kbit/s)...
Hash of data verified.
```

## 6.2 验证功能

使用 PuTTY 作为串口调试工具，新建一个 Serial 类型的 Session，按如下图所示配置串口，注意 Speed 为 74880，点击“Save”按钮保存 Session 方便以后直接打开，最后点击“Open”按钮：



按下 NodeMCU 的 RESET 按钮，将看到 SDDC 程序运行起来，并成功连接 EdgerOS：

COM4 - PuTTY

```
load 0x3ffe8408, len 24, room 8
tail 0
checksum 0x7c
load 0x3ffe8420, len 3528, room 8
tail 0
checksum 0x0b
I (45) boot: ESP-IDF e5ec3b46-dirty 2nd stage bootloader
I (46) boot: compile time 14:38:19
I (46) qio_mode: Enabling default flash chip QIO
I (53) boot: SPI Speed      : 40MHz
I (60) boot: SPI Mode      : QIO
I (66) boot: SPI Flash Size : 2MB
I (72) boot: Partition Table:
I (77) boot: ## Label      Usage          Type ST Offset   Length
I (89) boot:  0 nvs         WiFi data    01 02 00009000 00006000
I (100) boot:  1 phy_init    RF data     01 01 0000f000 00001000
I (112) boot:  2 factory     factory app  00 00 00010000 000f0000
I (123) boot: End of partition table
I (130) esp_image: segment 0: paddr=0x00010010 vaddr=0x40210010 size=0x4b778 (309112) map
I (247) esp_image: segment 1: paddr=0x0005b790 vaddr=0x4025b788 size=0x0d88c ( 55436) map
I (266) esp_image: segment 2: paddr=0x00069024 vaddr=0x3ffe8000 size=0x00698 ( 1688) load
I (267) esp_image: segment 3: paddr=0x000696c4 vaddr=0x40100000 size=0x00a50 ( 2640) load
I (278) esp_image: segment 4: paddr=0x0006a11c vaddr=0x40100a50 size=0x0584c ( 22604) load
I (298) boot: Loaded app from partition at offset 0x10000
I (321) system_api: Base MAC address is not set, read default base MAC address from EFUSE
I (326) system_api: Base MAC address is not set, read default base MAC address from EFUSE
phy_version: 1159.0, 85b471e, Apr 21 2020, 17:03:08, RTOS new
I (386) phy_init: phy ver: 1159_0
I (389) reset_reason: RTC reset 2 wakeup 0 store 0, reason is 2
I (409) example_connect: Connecting to EOS-00000F...
I (2080) wifi: state: 0 -> 2 (b0)
I (2169) wifi: state: 2 -> 3 (0)
I (2175) wifi: state: 3 -> 5 (10)
I (2178) wifi: pm start, type: 1
I (3070) tcpip_adapter: sta ip: 192.168.128.102, mask: 255.255.255.0, gw: 192.168.128.1
I (3076) example_connect: Connected to EOS-00000F
I (3080) example_connect: IPv4 address: 192.168.128.102
REPORT DATA: {
    "report": {
        "name": "IoT Pi",
        "type": "device",
        "excl": false,
        "desc": "IoT Pi",
        "model": "1",
        "vendor": "ACCOINFO"
    }
}
INVITE DATA: {
    "report": {
        "name": "IoT Pi",
        "type": "device",
        "excl": false,
        "desc": "IoT Pi",
        "model": "1",
        "vendor": "ACCOINFO"
    }
}
IP addr: 192.168.128.102
SDDC running...
Receive ping from: 192.168.128.1.
Receive discover from: 192.168.128.1.
Receive discover from: 192.168.128.1.
```