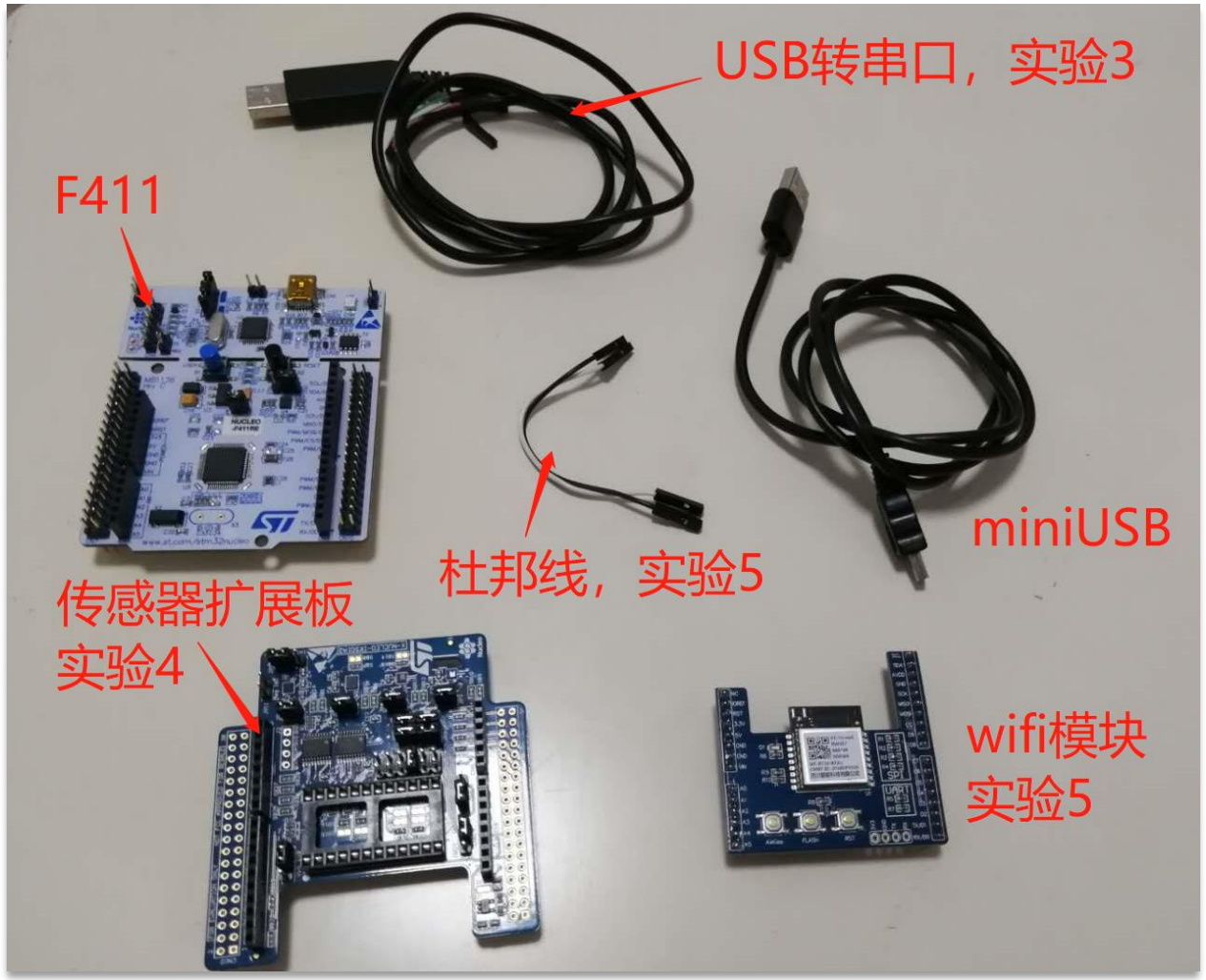
入门实战

目录

RT-Thread

器材介绍



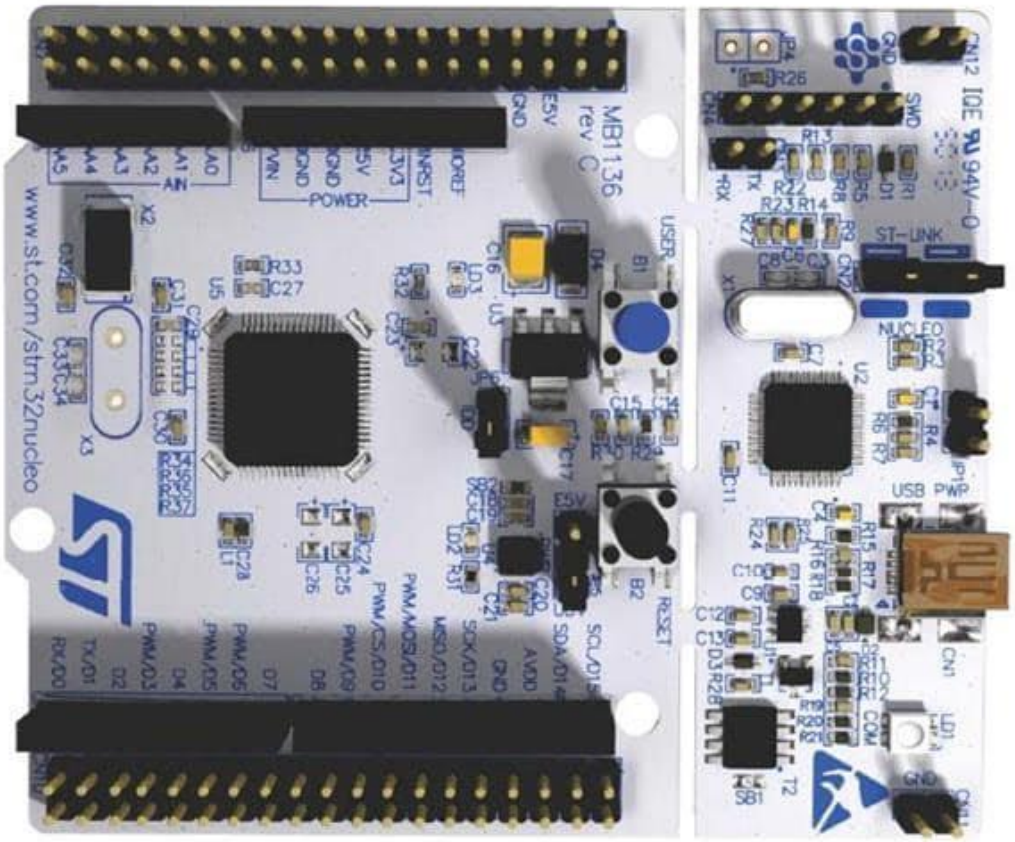
RT-Thread

实验一、运行RT-Thread

硬件-STM32F411 Nucleo-64

STM32F411RET6

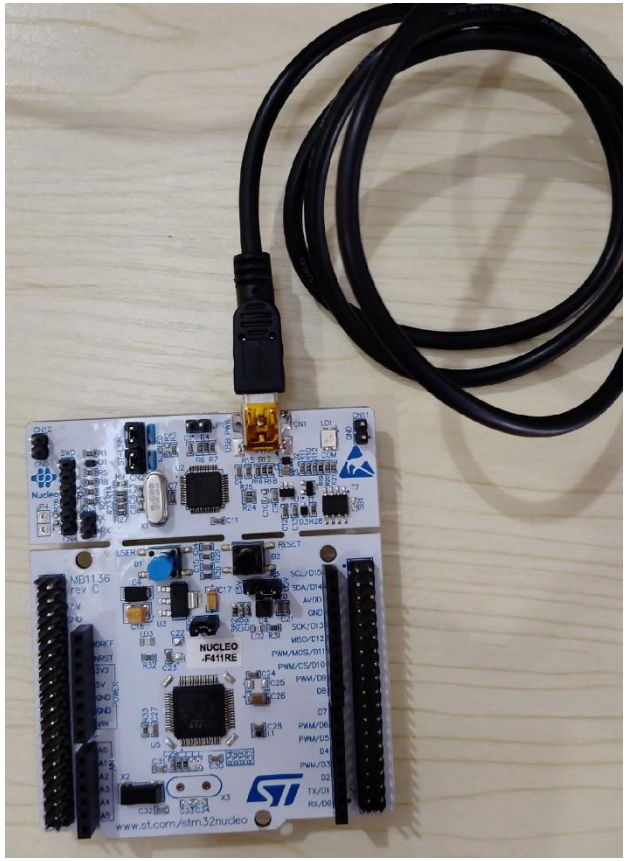
- ARM Cortex-M4 100NHZ
- Flash：512KB
- SRAM： 128KB
- ST-LINK V2.1



RT-Thread

硬件连接

- 使用 Type-A to Mini-B 线连接开发板和 PC



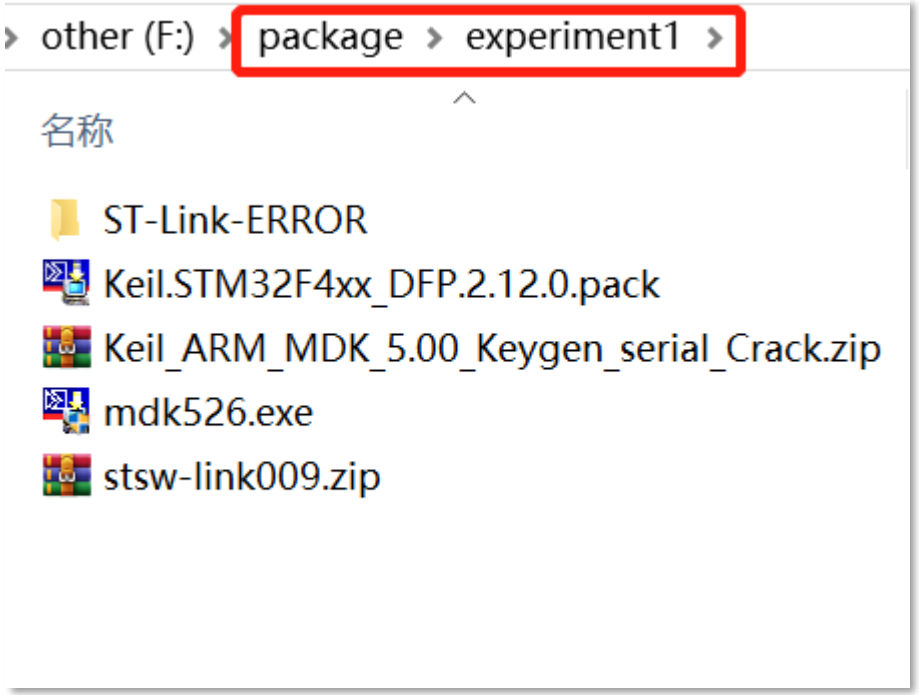
STM32 F4

RT-Thread

软件环境搭建

硬件需要的开发环境

- 安装MDK5
- 双击安装Keil.STM32F4xx….pack
- 解压安装ST-Link驱动（stsw-link009.zip）



解压RT-Thread源代码到“**非中文字符**”，“**不含空格**” 目录

RT-Thread

源码结构说明

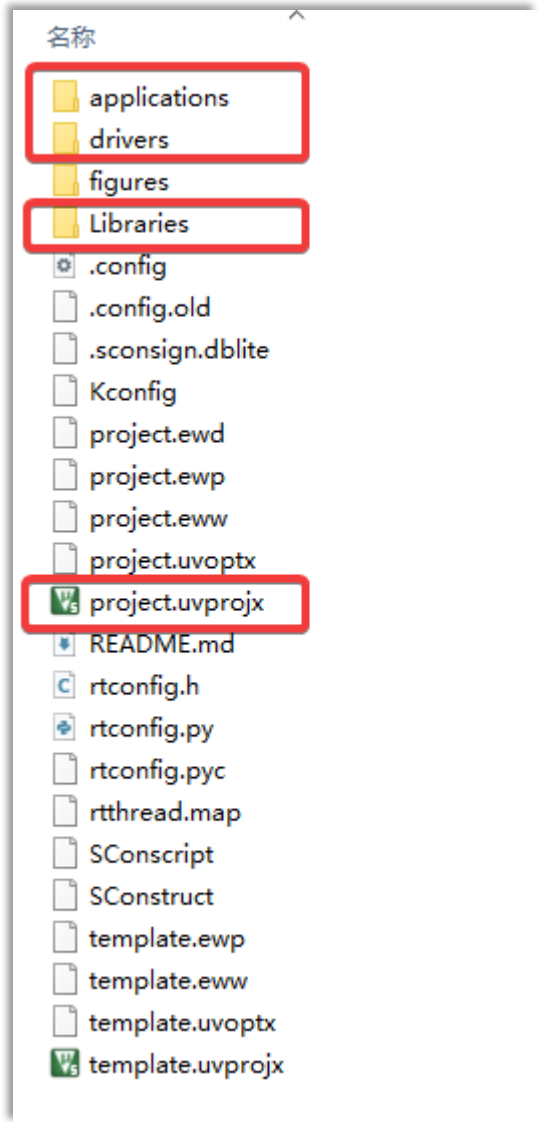
名称

- bsp
- components
- documentation
- examples
- include
- libcpu
- src
- tools
- .gitattributes
- .gitignore
- .travis.yml
- AUTHORS
- ChangeLog.md
- Kconfig
- LICENSE
- README.md
- README_zh.md

根目录名	描述
bsp	板级支持包
components	RT-Thread的各个组件代码，例如文件系统, shell
include	RT-Thread内核的头文件
libcpu	各类芯片的移植代码，此处包含了Cortex-M0的移植文件
src	RT-Thread内核的源文件

RT-Thread

BSP目录结构



bsp目录下	描述
applications	RT-Thread 的用户例程, 应用层代码就写到这里
drivers	RT-Thread 的驱动，不同平台的底层驱动具体实现
Libraries	芯片所用的固件库文件
Project.uvprojx	KEIL5 工程文件

RT-Thread

工程

F:\rt-thread-no-save\bsp\es32f0654\project.uvprojx - µVision

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

time_us

rt-thread_es32f065x

工程配置

下载

编译

Project

Project: project

rt-thread_es32f065x

- Kernel
- Applications
  - main.c
- Drivers
- Libraries
- cpu
- DeviceDrivers
- finsh

Project | Books | {} Func... | {}, Temp...

main.c

```
 * 2019-01-28      wangyq        the first version
 */

#include <rtthread.h>
#include <rtdevice.h>

#define LED_PIN 39

int main(void)
{
    int count = 1;
    /* set PC08 pin mode to output */
    rt_pin_mode(LED_PIN, PIN_MODE_OUTPUT);

    while (count++)
    {
        rt_pin_write(LED_PIN, PIN_HIGH);
        rt_thread_mdelay(500);
        rt_pin_write(LED_PIN, PIN_LOW);
        rt_thread_mdelay(500);
    }
    return RT_EOK;
}
```

用户程序入口

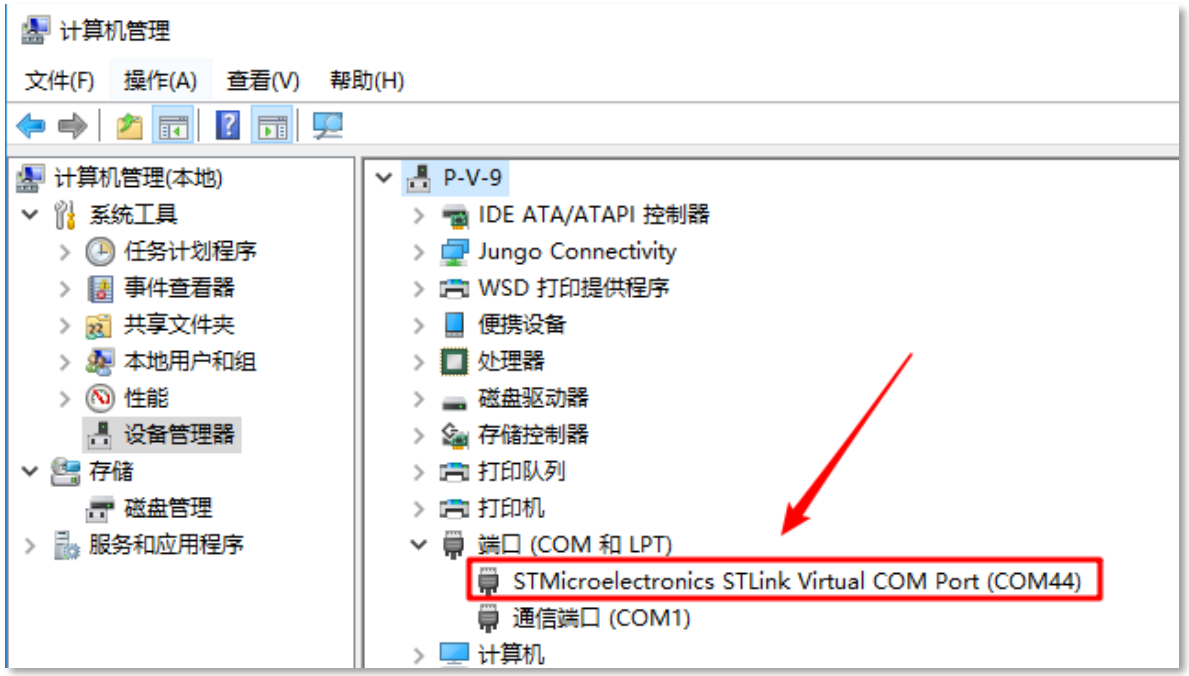
Build Output

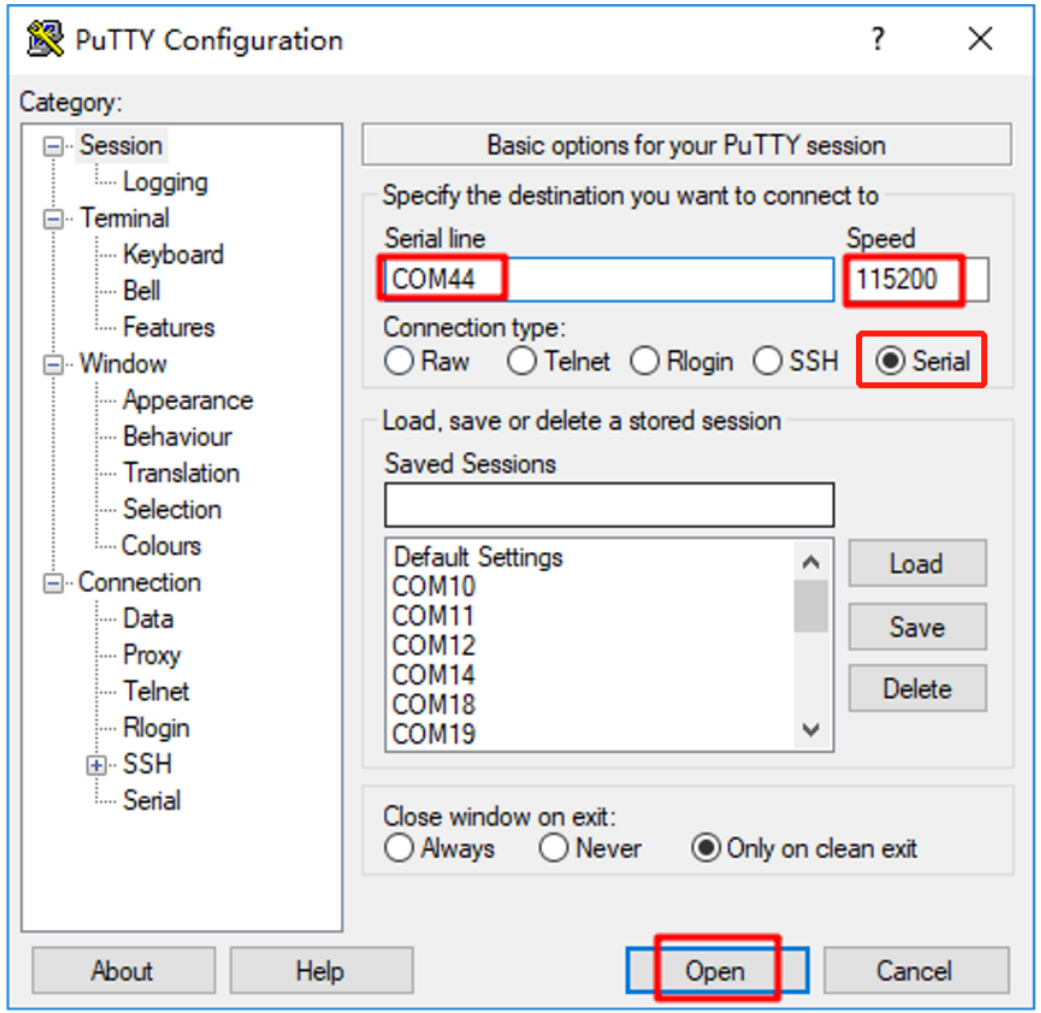
```
Load "F:\\rt-thread-no-save\\bsp\\es32f0654\\build\\keil\\rtthread.axf"
Erase Done.
Programming Done.
Verify OK.
Application running ...
Flash Load finished at 21:23:36
```

CMSIS-DAP Debugger

RT-Thread

连接终端工具





注意：如 STLink虚拟串口 有黄色感叹号，请参考资料包ST-Link-ERROR中解决方法。

RT-Thread

运行结果

按下复位键，RT-Thread Logo正常打印

```
 \ | /
- RT -     Thread Operating System
 / | \     4.0.1 build Apr 10 2019
 2006 - 2019 Copyright by rt-thread team
msh >
```

RT-Thread

运行结果

按下 ‘TAB’ 键查看内置的命令

```
msh >
RT-Thread shell commands:
reboot            - Reboot System
version           - show RT-Thread version information
list_thread       - list thread
list_sem          - list semaphore in system
list_event        - list event in system
list_mutex        - list mutex in system
list_mailbox      - list mail box in system
list_msgqueue     - list message queue in system
list_mempool      - list memory pool in system
list_timer        - list timer in system
list_device       - list device in system
help              - RT-Thread shell help.
ps                - List threads in the system.
time              - Execute command with time.
free              - Show the memory usage in the system.
```

RT-Thread

运行结果

输入 “ps” 命令查看线程的状态

```
msh >ps
thread   pri  status      sp     stack size max used left tick  error
-------- ---  ------- ---------- ----------  ------  ---------- ---
tshell    20  running 0x00000084 0x00001000     12%   0x00000008 000
tidle0    31  ready   0x00000044 0x00000100     32%   0x00000004 000
main      10  suspend 0x0000008c 0x00000800     10%   0x00000013 000
msh >
```

RT-Thread

运行结果

输入 “list_device” 命令查看注册到系统的设备

```
list_event        - list event in system
list_mutex        - list mutex in system
list_mailbox      - list mail box in system
list_msgqueue     - list message queue in system
list_mempool      - list memory pool in system
list_timer        - list timer in system
list_device       - list device in system
help              - RT-Thread shell help.
ps                - List threads in the system.
time              - Execute command with time.
free              - Show the memory usage in the system.

msh >list device
device           type                ref count
-------- -------------------- ----------
uart2    Character Device     2
pin      Miscellaneous Device 0
msh >
```

RT-Thread

运行结果

输入 “free” 命令查看系统内存占用

```
list_mailbox      - list mail box in system
list_msgqueue     - list message queue in system
list_mempool      - list memory pool in system
list_timer        - list timer in system
list_device       - list device in system
help              - RT-Thread shell help.
ps                - List threads in the system.
time              - Execute command with time.
free              - Show the memory usage in the system.

msh >free
total memory: 127688
used memory : 7072
maximum allocated memory: 7072
msh >
```

RT-Thread

实验二、快速搭建RT-Thread项目框架

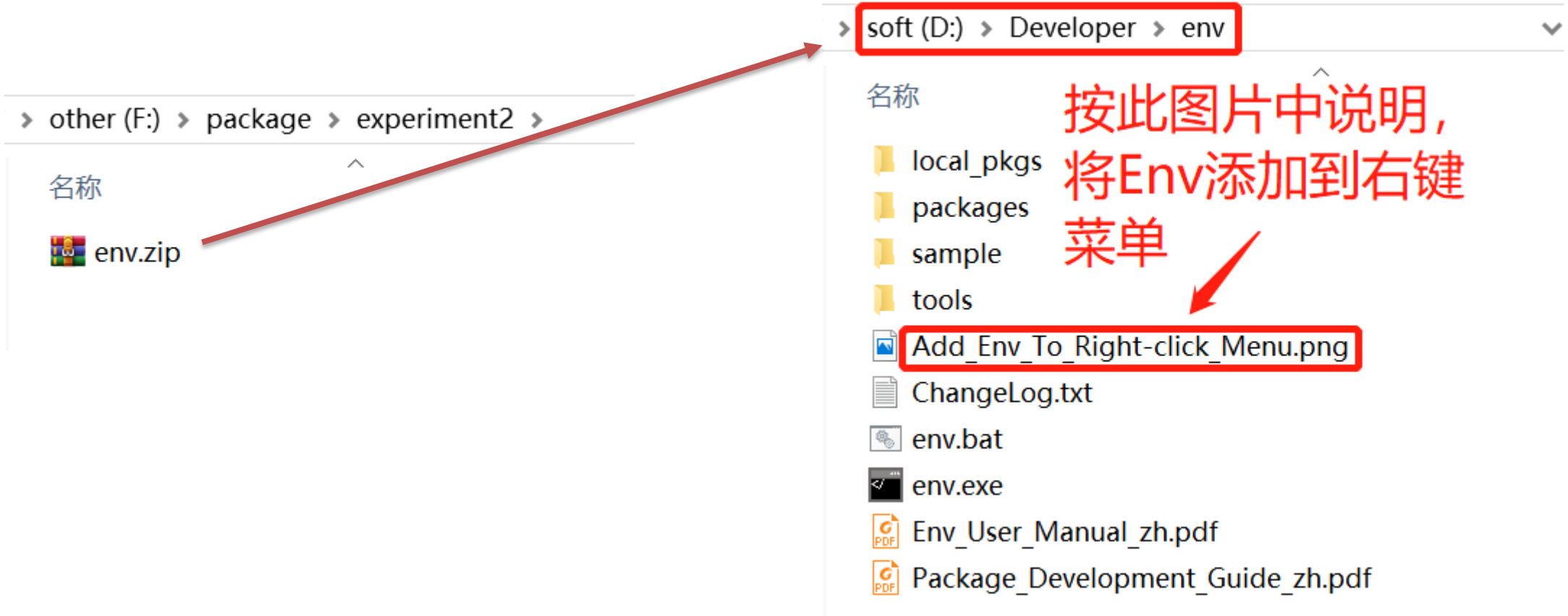
需求分析

人机调试接口	FinSH组件
使用 IIC 控制传感器	i2c设备
需要使用 SPI Flash	spi 设备

RT-Thread

环境准备

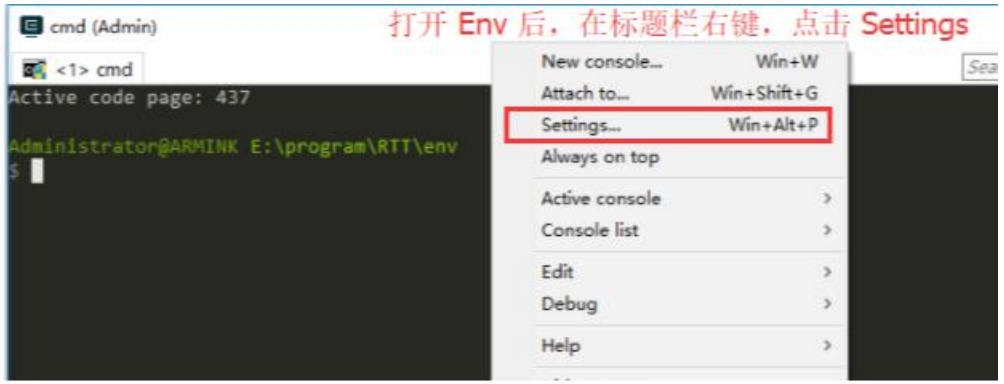
解压 env.zip 到“**非中文字符**”，“**不含空格**”的目录



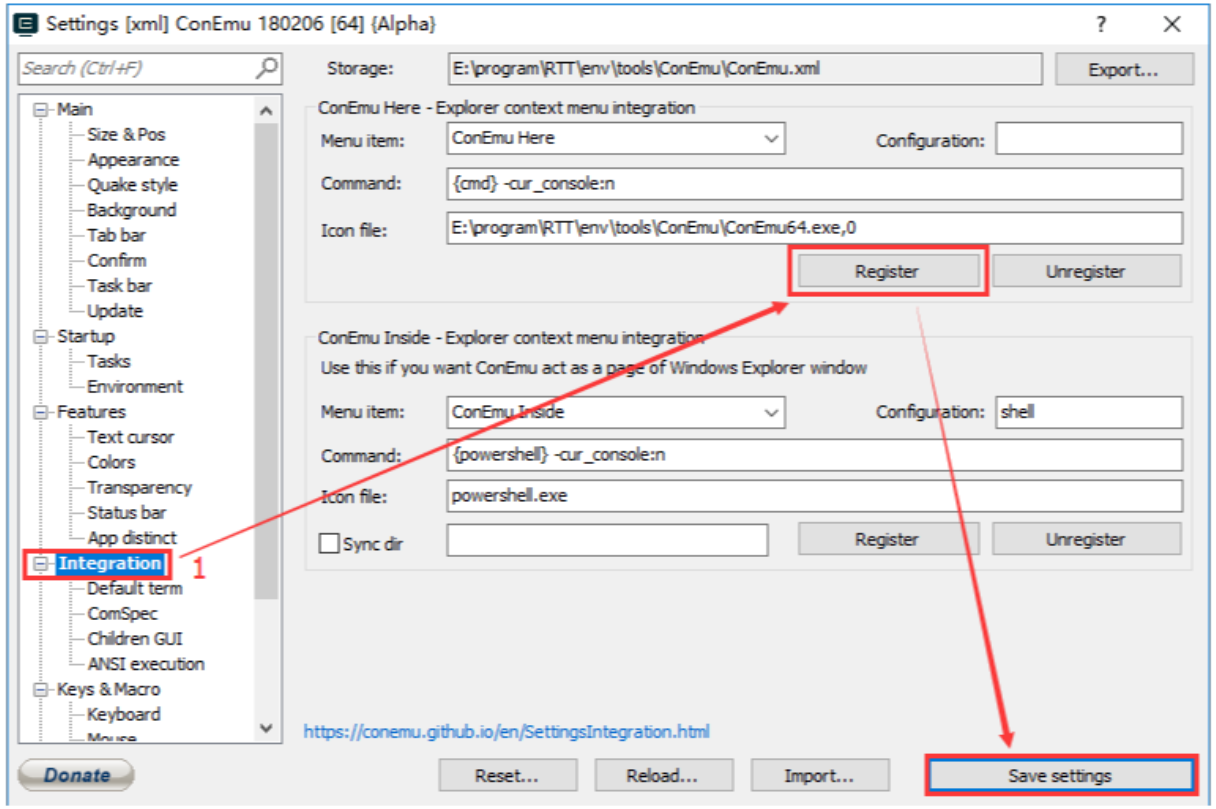
RT-Thread

Env注册到右键快捷菜单

①

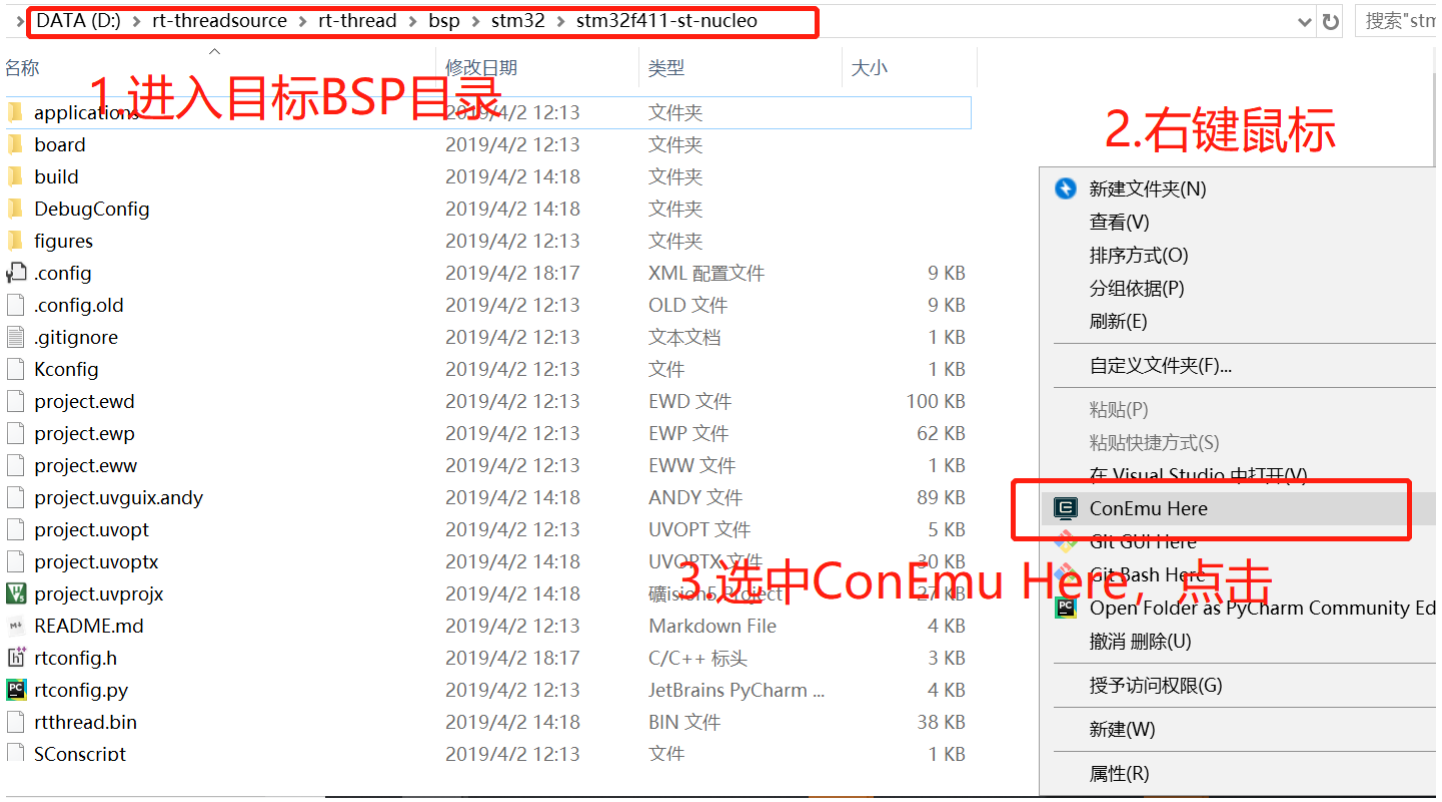


②



RT-Thread

Env在具体的BSP下使用



RT-Thread

Env在具体的BSP下使用

输入 **menuconfig 回车**

cmd

<1> cmd

```
欢迎使用 RT-Thread env (V1.1.1) 工具
 \ | /
- RT -     Thread Operating System
 / | \
2006 - 2019 Copyright by rt-thread team
Online help documents : https://www.rt-thread.org/document/site

andy@LAPTOP-0G7BUCLF D:\rt-threadsource\rt-thread\bsp\stm32\stm32f411-st-nucleo
> menuconfig
```

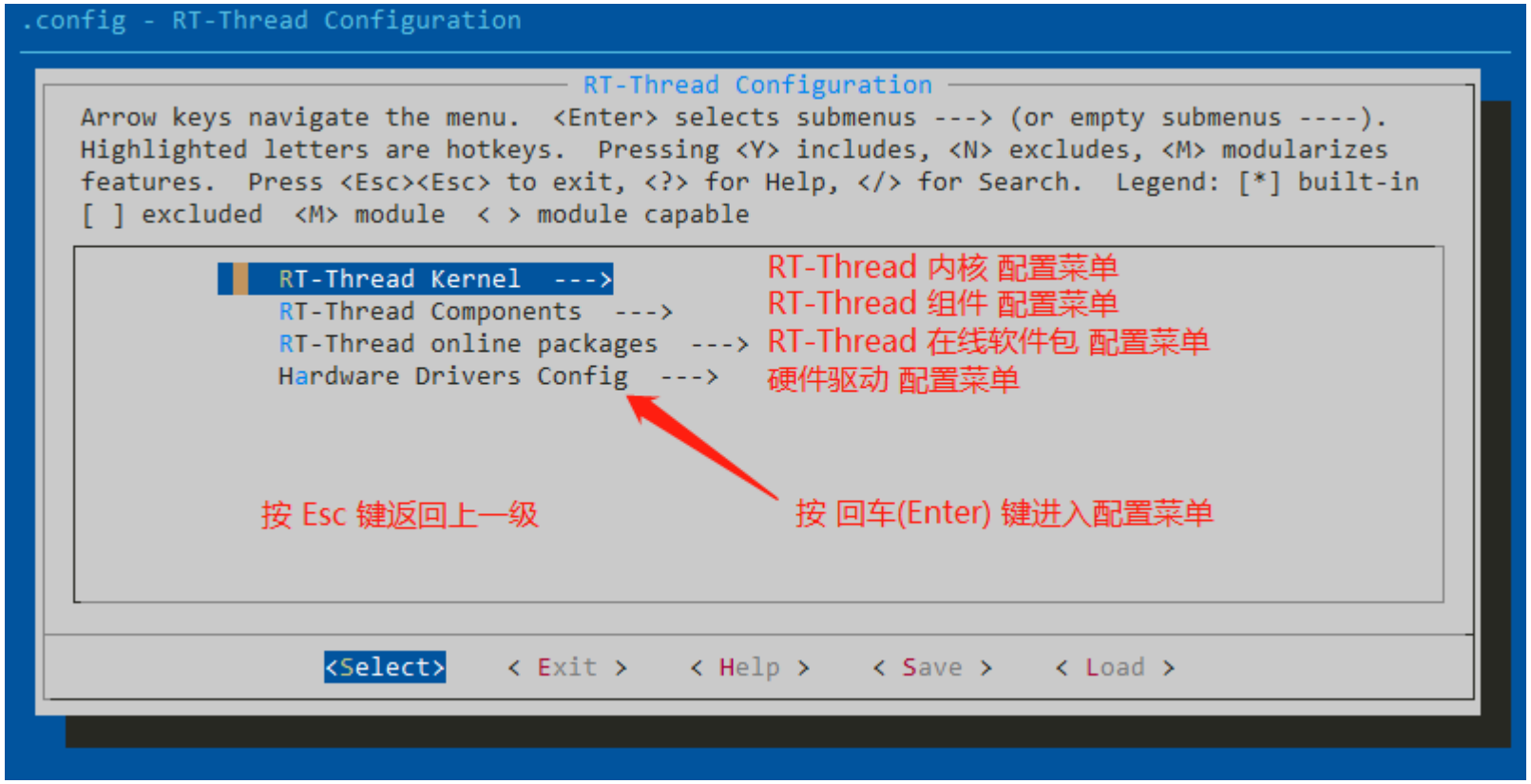
具体的BSP下执行menuconfig

RT-Thread

Env在具体的BSP下使用

menuconfig 基本操作：

1. 方向 UP 键向上
2. 方向 DOWN 键向下
3. 空格键切换选中状态
4. Enter 键进入下级菜单或修改选项的值
5. ESC 键后退



Env在具体的BSP下使用

开启 i2c、spi 驱动，其对应的设备框架，会被自动选中。

```
→ Hardware Drivers Config → On-chip Peripheral Drivers
                    On-chip Peripheral Drivers
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or emp
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> exclude
  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*]
  <M> module  < > module capable

            [*] Enable GPIO
            [*] Enable UART  --->
            [*] Enable I2C1 BUS (software simulation)  --->
            [*] Enable SPI BUS  --->
            [ ] Enable PWM  ----
            [ ] Enable RTC  ----
            [ ] Enable on-chip FLASH
```

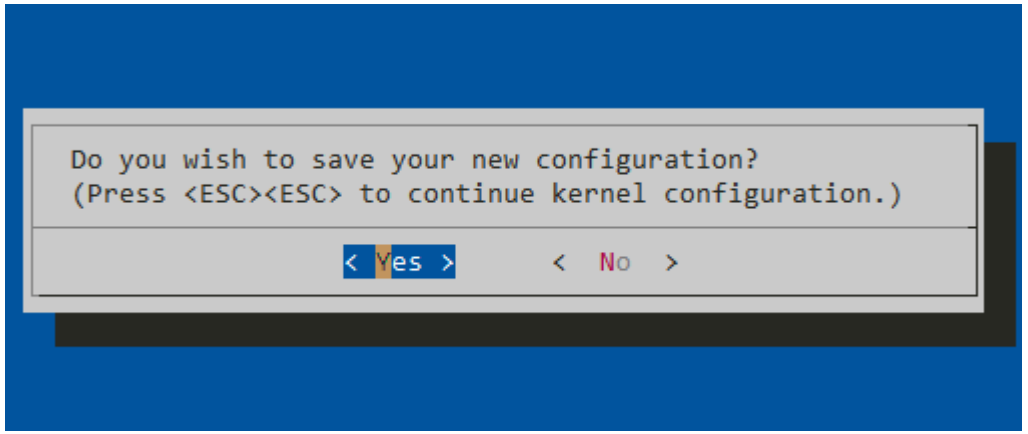
```
-*- Enable I2C1 BUS (software simulation)
      *** Notice: PB8 --> 24; PB9 --> 25 ***
(24)  i2c1 scl pin number (NEW)
(25)  I2C1 sda pin number (NEW)
```

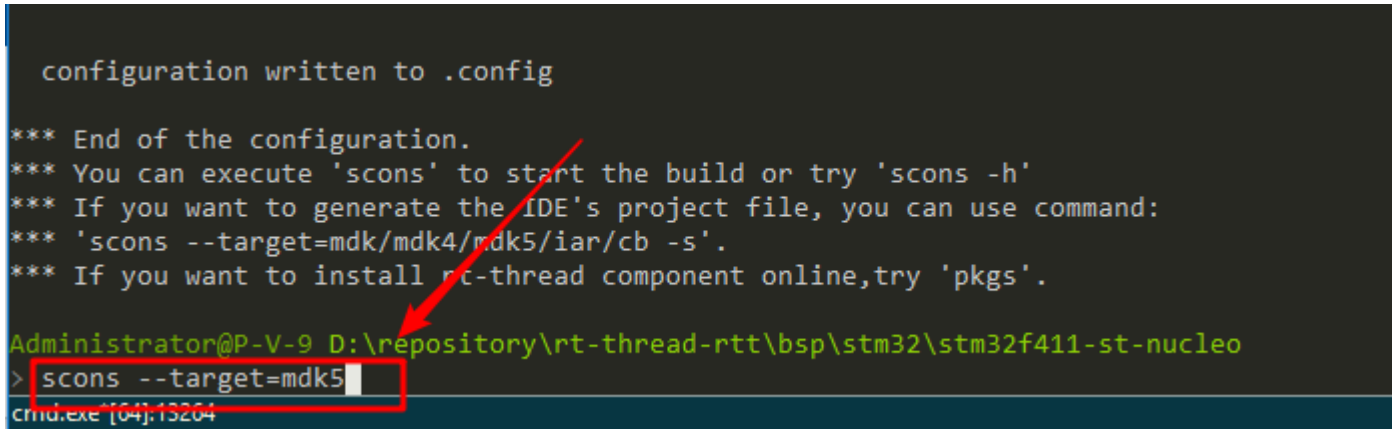
```
-*- Enable SPI BUS
[*]   Enable SPI1 BUS
[ ]     Enable SPI1 TX DMA (NEW)
[ ]     Enable SPI1 RX DMA (NEW)
```

RT-Thread

Env在具体的BSP下使用

按 ESC 退出，保存设置；输入命令 scons --target=mdk5 重新生成工程

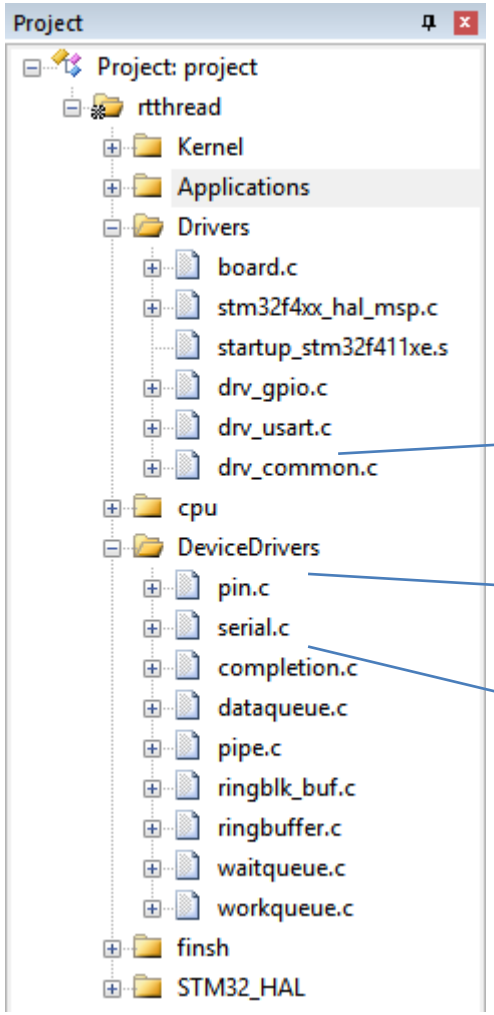




RT-Thread

Env在具体的BSP下使用

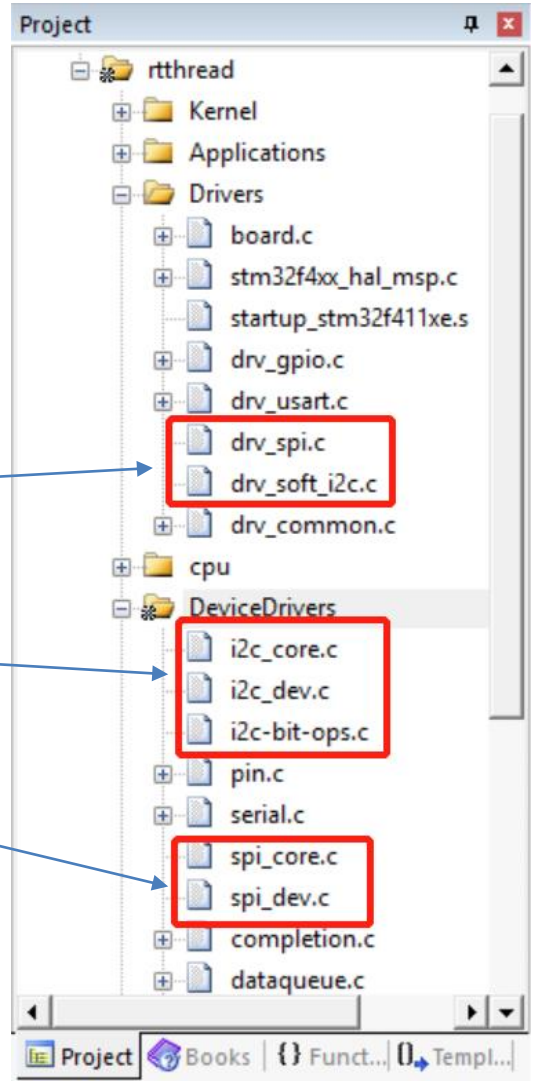
配置工程前后工程文件对比



spi/i2c驱动

i2c框架

spi框架



RT-Thread

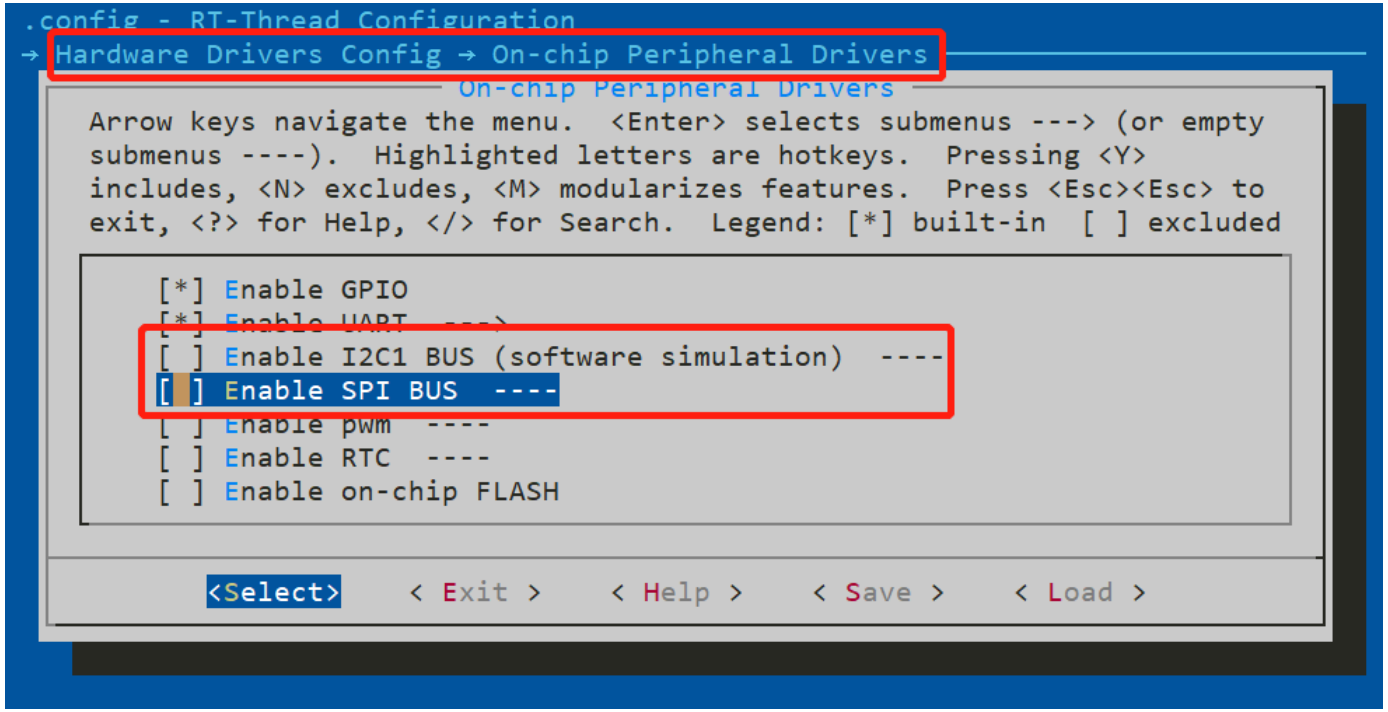
Env在具体的BSP下使用

输入 “list_device” 命令看看

```
 \ | /
- RT -     Thread Operating System
 / | \     4.0.1 build Apr 10 2019
 2006 - 2019 Copyright by rt-thread team
msh >list_device
device           type                ref count
-------- -------------------- ----------
i2c1     I2C Bus              0
spi1     SPI Bus              0
uart2    Character Device     2
pin      Miscellaneous Device 0
msh >
```

如何关闭 i2c/spi 设备

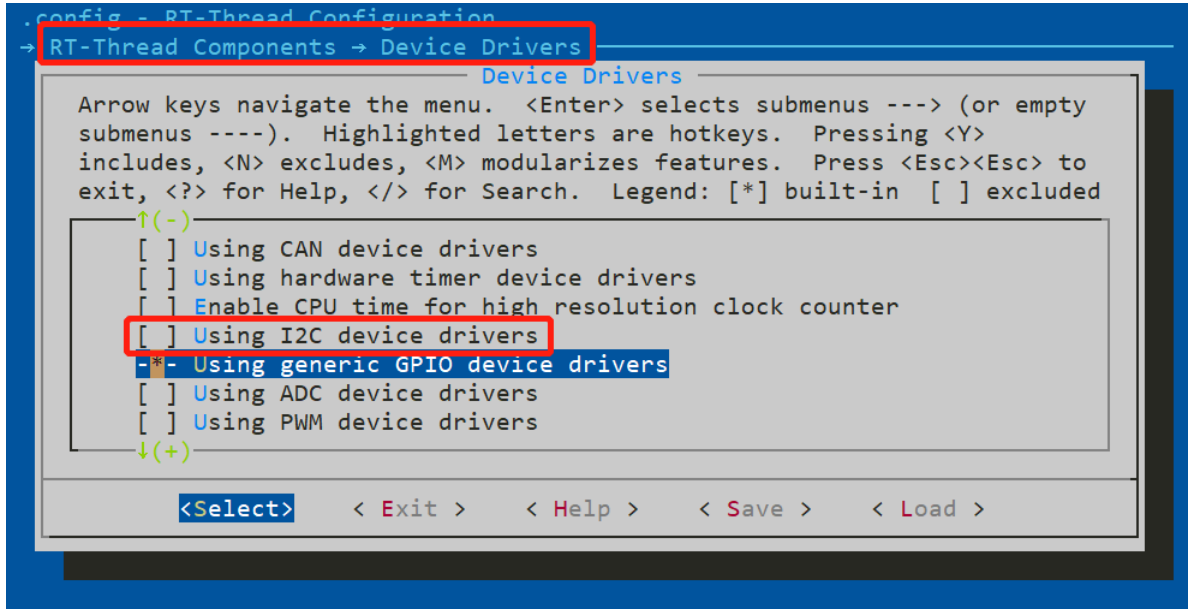
1. 关闭i2c/spi设备驱动

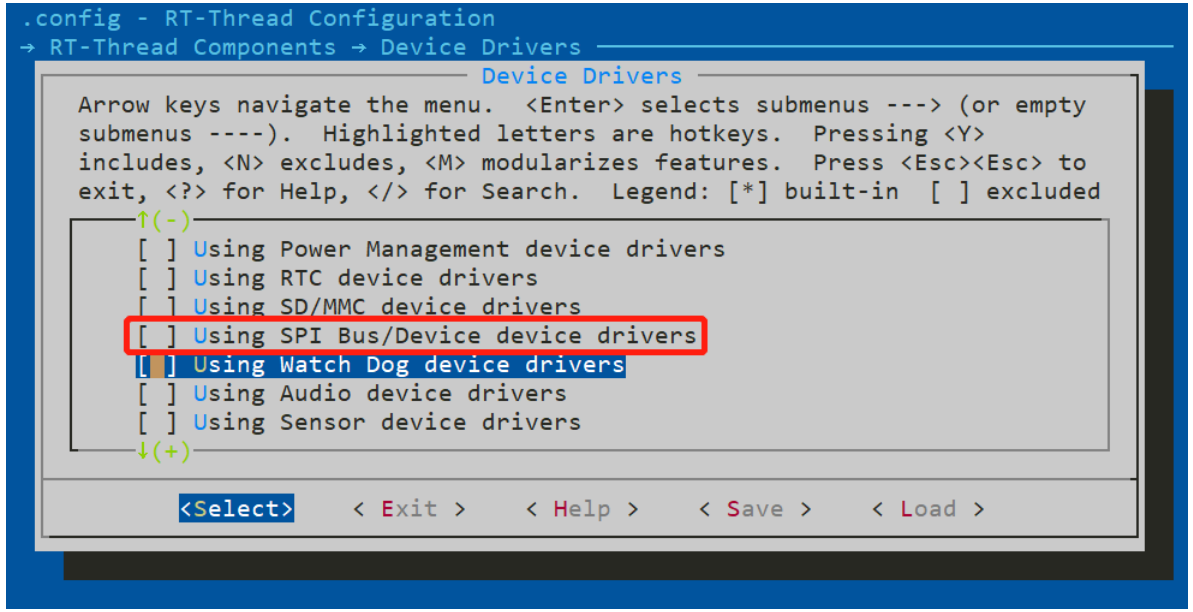


RT-Thread

如何关闭 i2c/spi 设备

2. 关闭i2c/spi设备框架

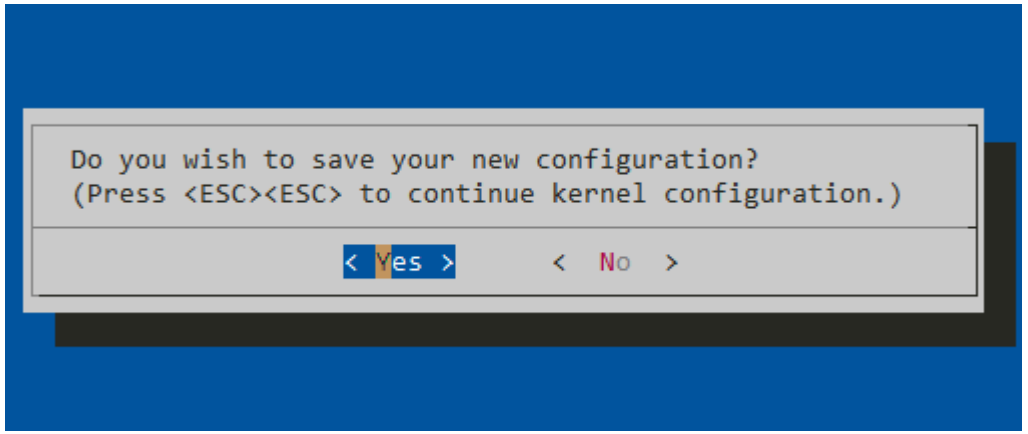


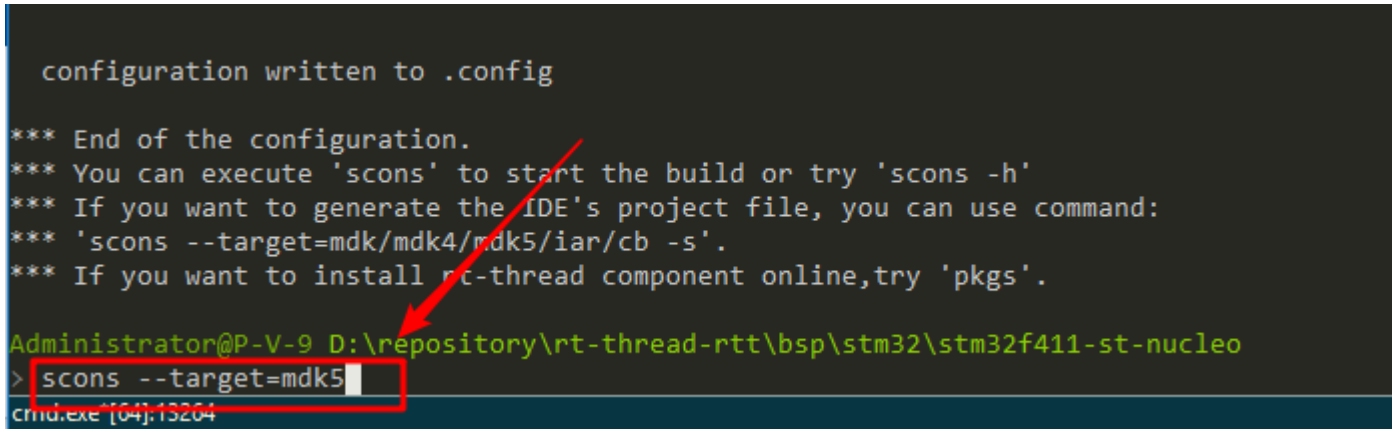


RT-Thread

如何关闭 i2c/spi 设备

按 ESC 退出，保存设置；输入命令 scons --target=mdk5 重新生成工程





RT-Thread

编译运行

- 编译下载，观察现象。

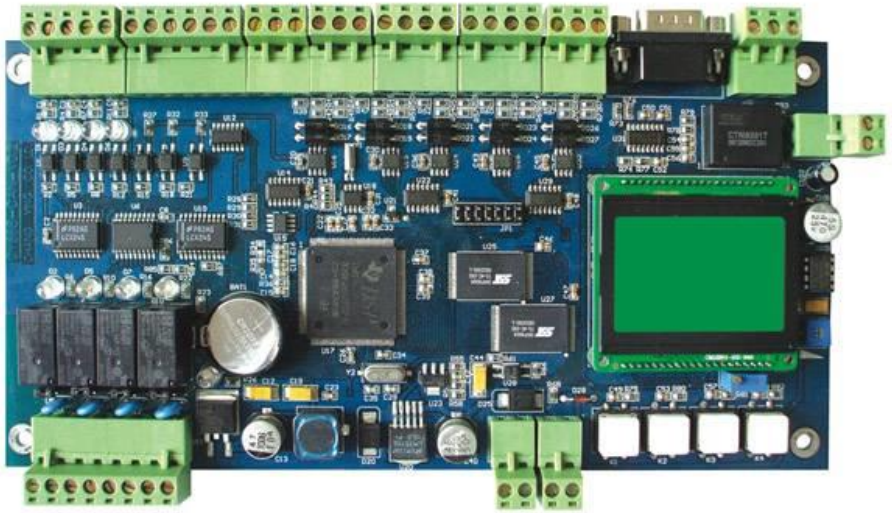
```
list_event        - list event in system
list_mutex        - list mutex in system
list_mailbox      - list mail box in system
list_msgqueue     - list message queue in system
list_mempool      - list memory pool in system
list_timer        - list timer in system
list_device       - list device in system
help              - RT-Thread shell help.
ps                - List threads in the system.
time              - Execute command with time.
free              - Show the memory usage in the system.

msh >list device
device           type                ref count
-------- -------------------- ----------
uart2    Character Device     2
pin      Miscellaneous Device 0
msh >
```

RT-Thread

实验三、适配项目开发板

适配项目开发板

- 在已有的 BSP 中找到与项目开发板相同芯片的开发板



- 配置系统时钟
- 配置 FinSH 使用的串口。

RT-Thread

适配项目开发板

- 配置系统时钟
- 配置 FinSH 使用的串口
- 硬件连接
- 编译运行

RT-Thread

配置系统时钟

Project

- Project: project
  - rtthread
    - Kernel
    - Applications
      - main.c
    - Drivers
      - board.c
      - stm32f4xx_hal_msp.c
      - startup_stm32f411xe.s
      - drv_gpio.c
      - drv_usart.c
      - drv_common.c
    - cpu
    - DeviceDrivers
    - finsh
    - STM32_HAL

main.c | board.c

```c
 *
 * Change Logs:
 * Date           Author       Notes
 * 2018-11-06     SummerGift   first version
 */

#include "board.h"

void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /**Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1),
  /**Initializes the CPU, AHB and APB busses clocks
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_BYPASS;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 4;
  RCC_OscInitStruct.PLL.PLLN = 84;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /**Initializes the CPU, AHB and APB busses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE
```

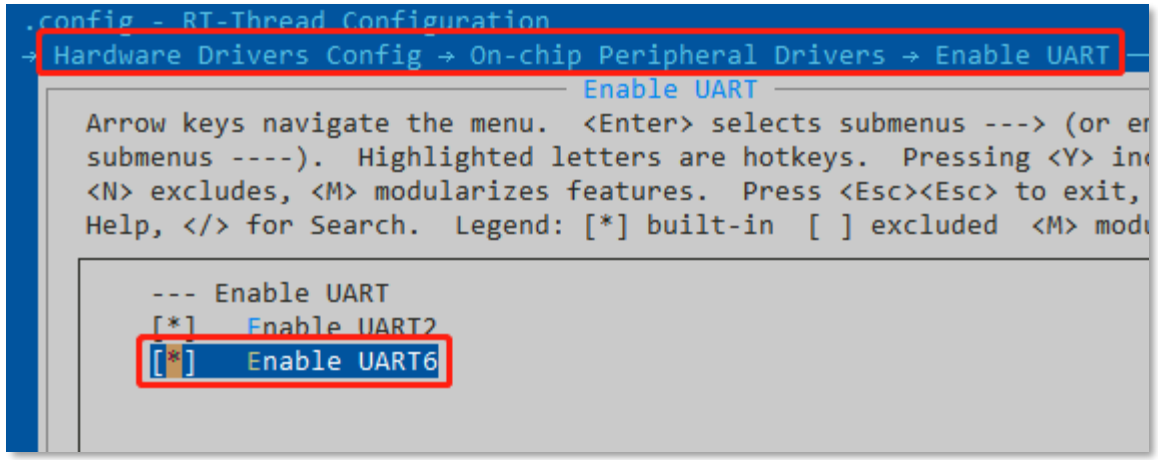
RT-Thread

适配项目开发板

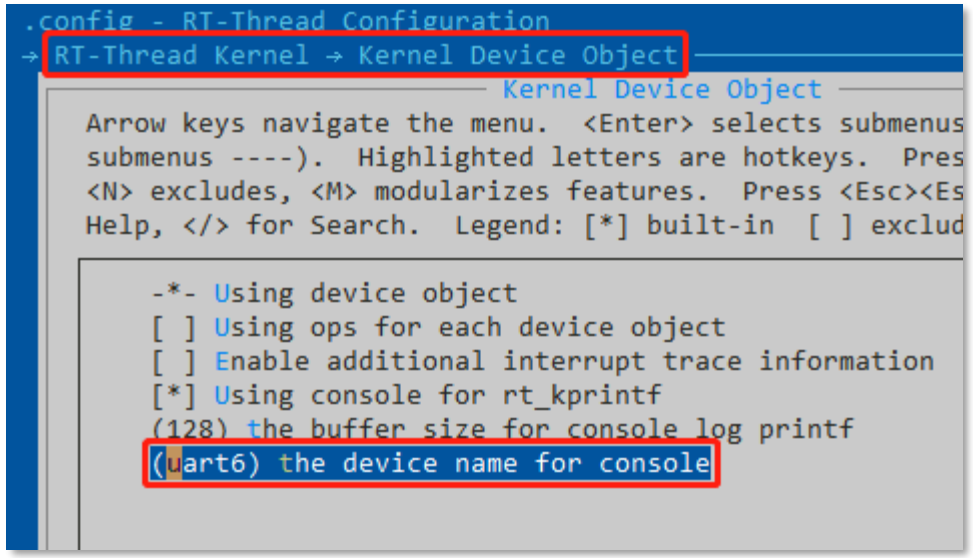
- 配置系统时钟
- 配置 FinSH 使用的串口
- 硬件连接
- 编译运行

RT-Thread

配置 FinSH 使用的串口



1. 开启对应串口

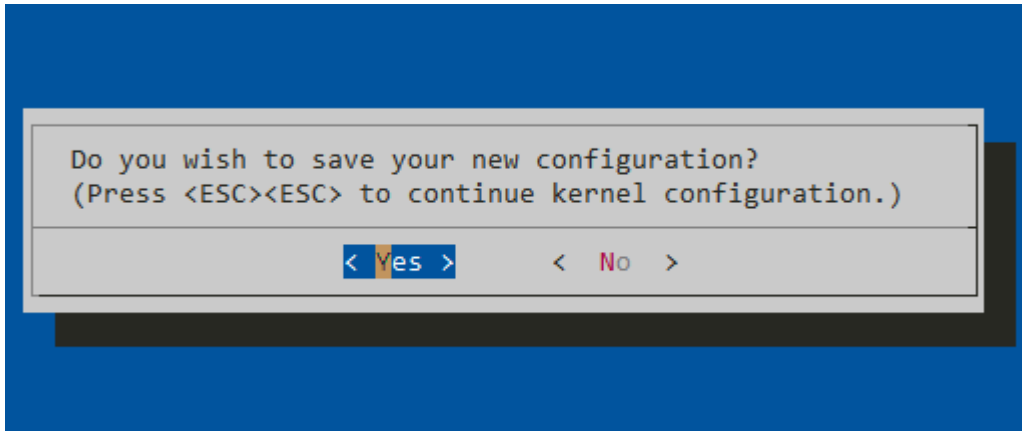


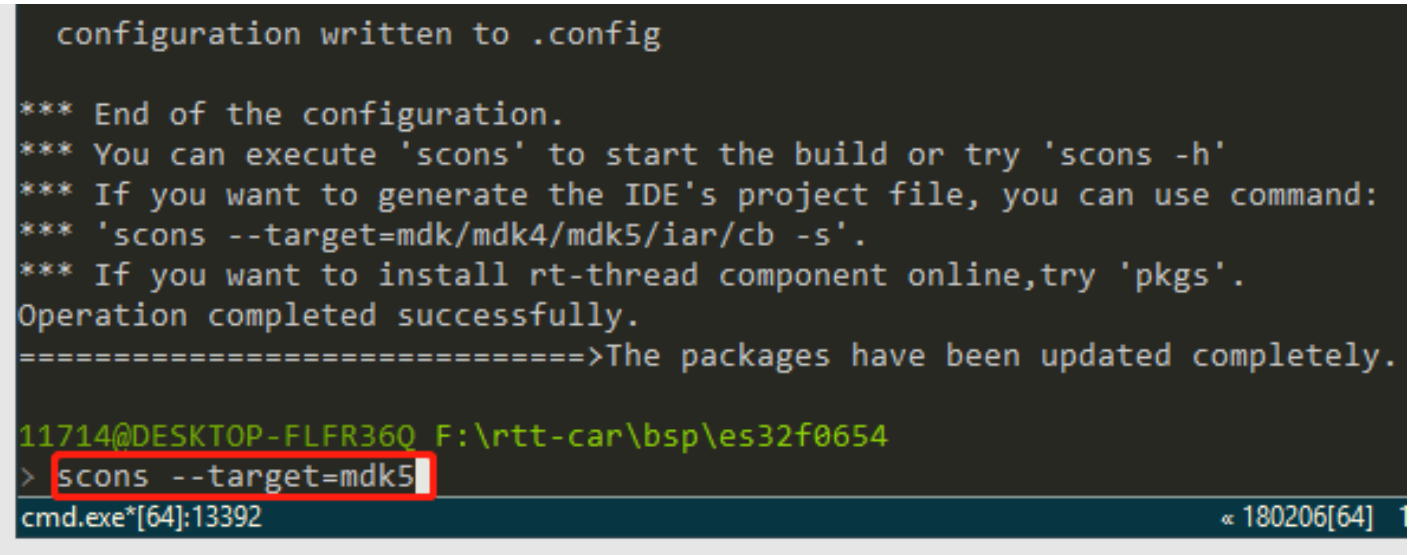
2. 配置console对应串口

RT-Thread

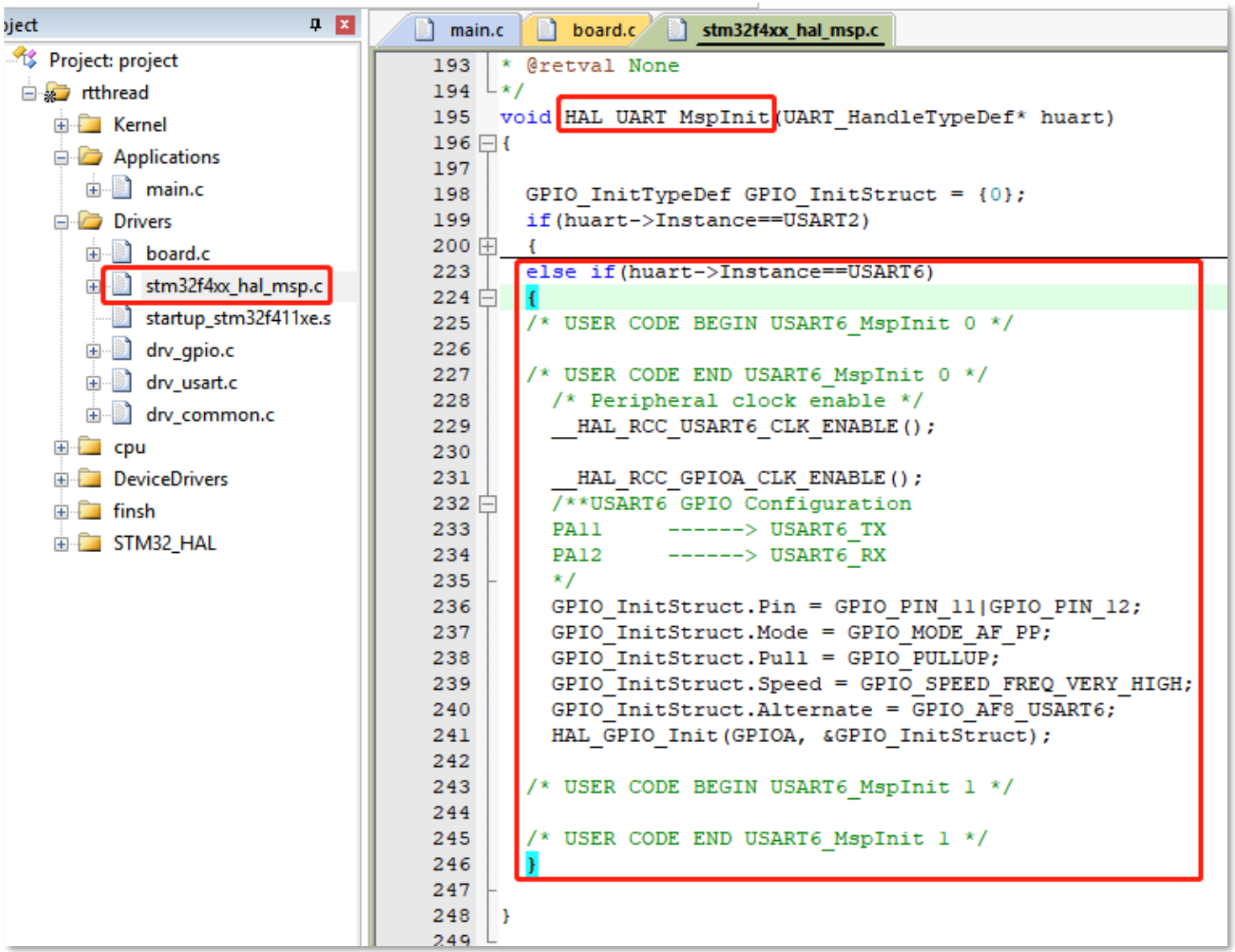
env配置

- 退出，保存设置；输入命令 scons --target=mdk5 重新生成工程





确认串口引脚



RT-Thread

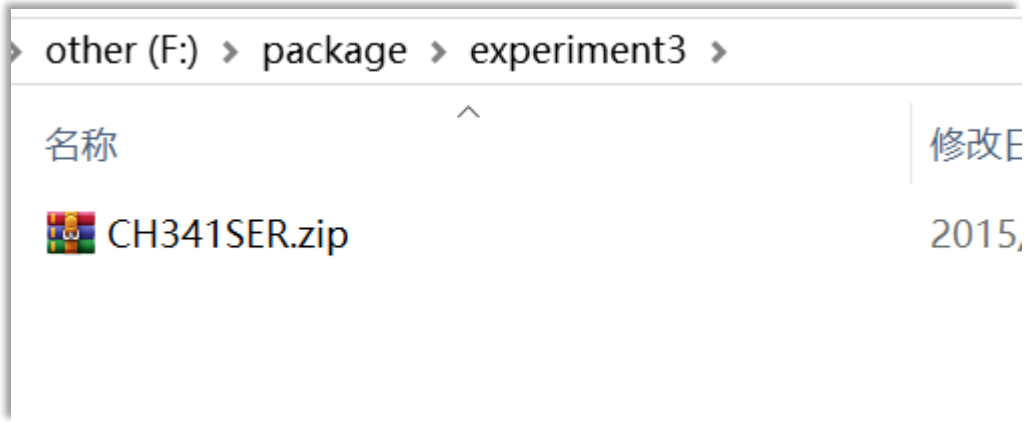
适配项目开发板

- 配置系统时钟
- 配置 FinSH 使用的串口
- 硬件连接
- 编译运行

RT-Thread

驱动安装

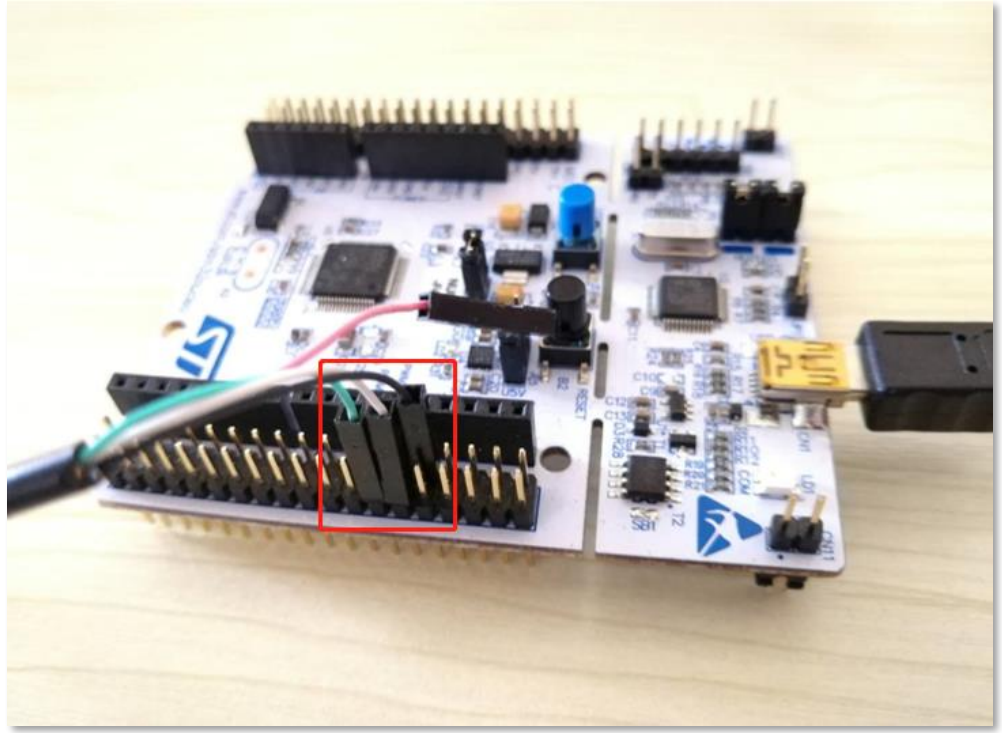
- 安装 CH340 USB转串口模块的驱动



硬件连接

- 使用USB转串口工具连接对应引脚到PC

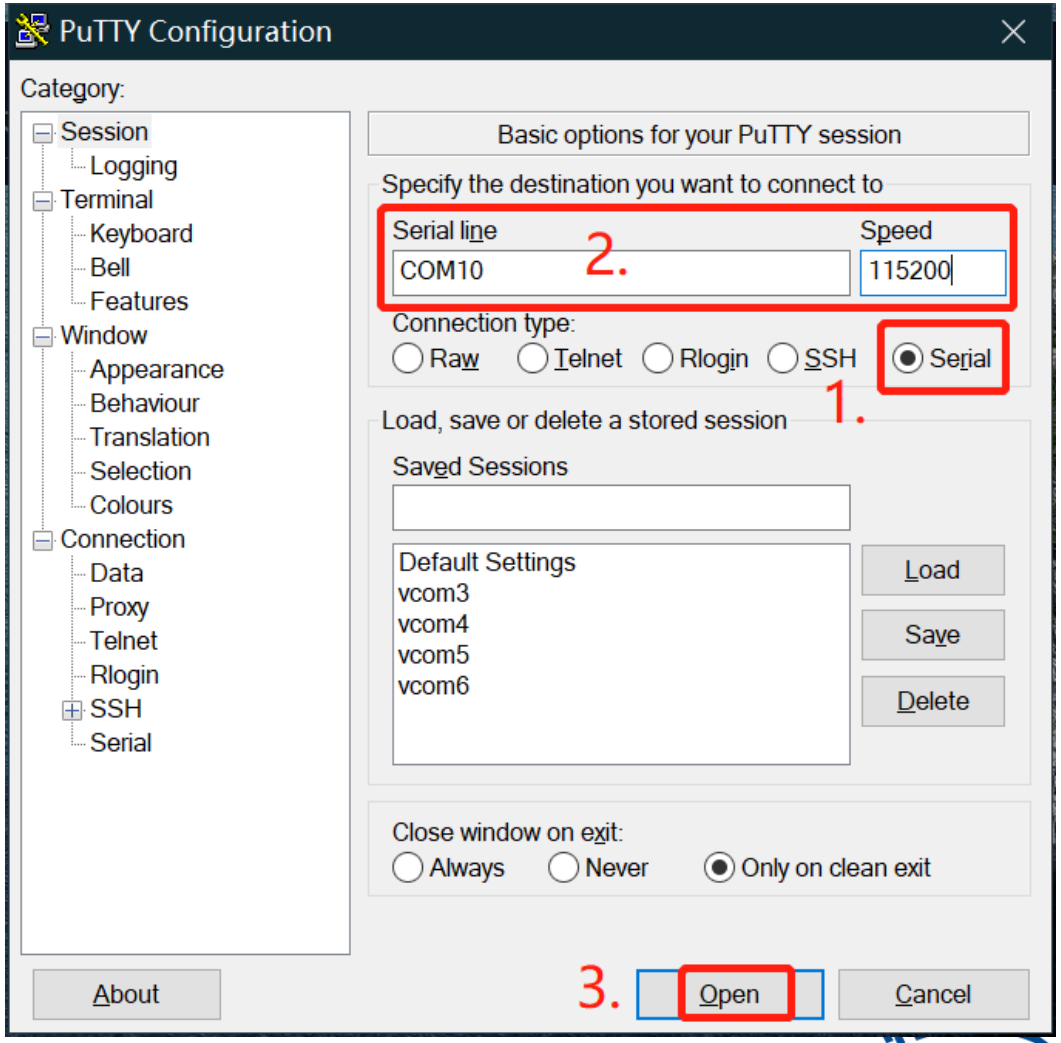
绿色接PA12
白色接PA11



注意：引脚定义在开发板的包装上

RT-Thread

连接终端工具



RT-Thread

适配项目开发板

- 配置系统时钟
- 配置 FinSH 使用的串口
- 硬件连接
- 编译运行

RT-Thread

编译运行

- 编译下载，观察现象。

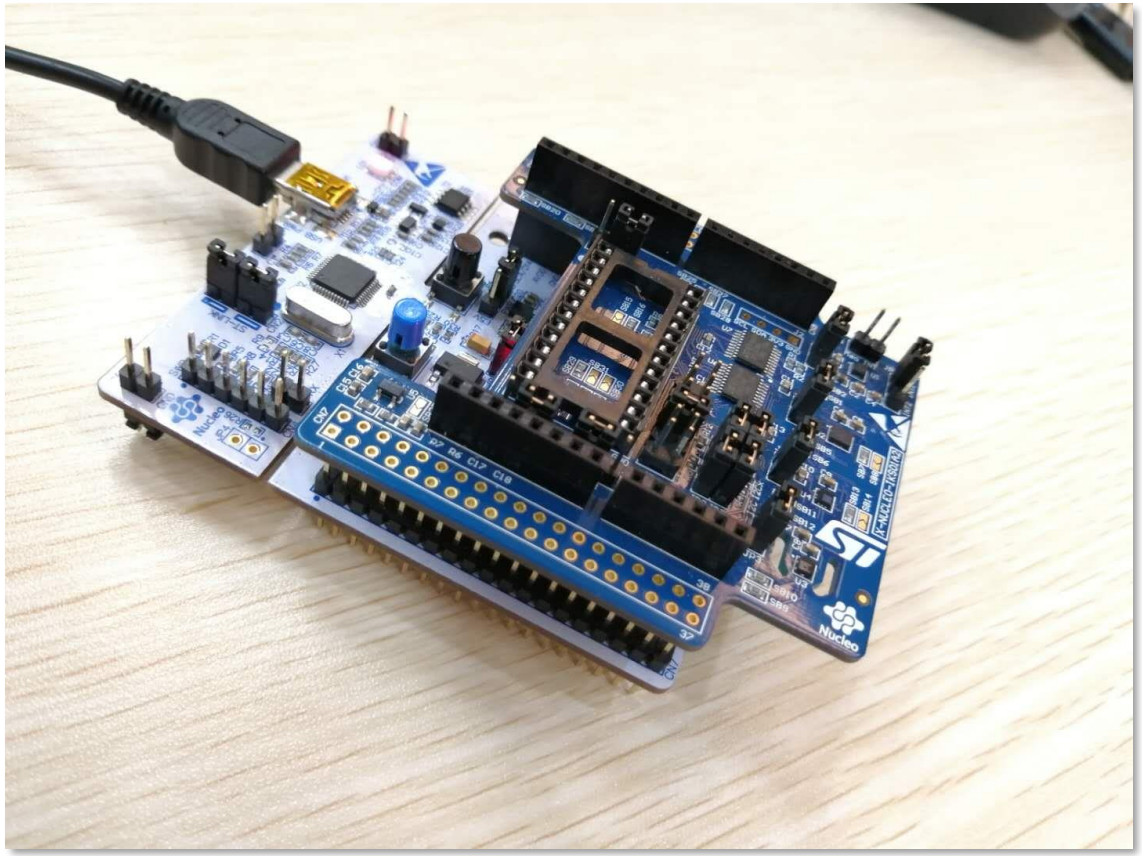
```
 \ | /
- RT -     Thread Operating System
 / | \     4.0.1 build Apr 10 2019
 2006 - 2019 Copyright by rt-thread team
msh >
```

RT-Thread

实验四、读取传感器数据

硬件连接

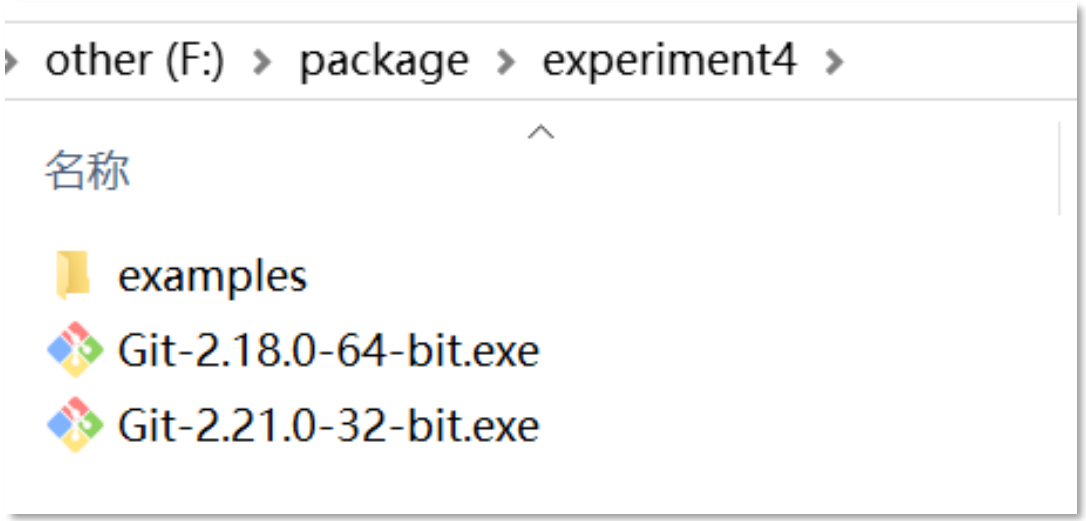
- 传感器插入开发板上



RT-Thread

软件安装

- 64位电脑双击 Git-2.18.0-64-bit.exe 安装 git
- 32位电脑双击 Git-2.21.0-32-bit.exe 安装 git



RT-Thread

获取软件包

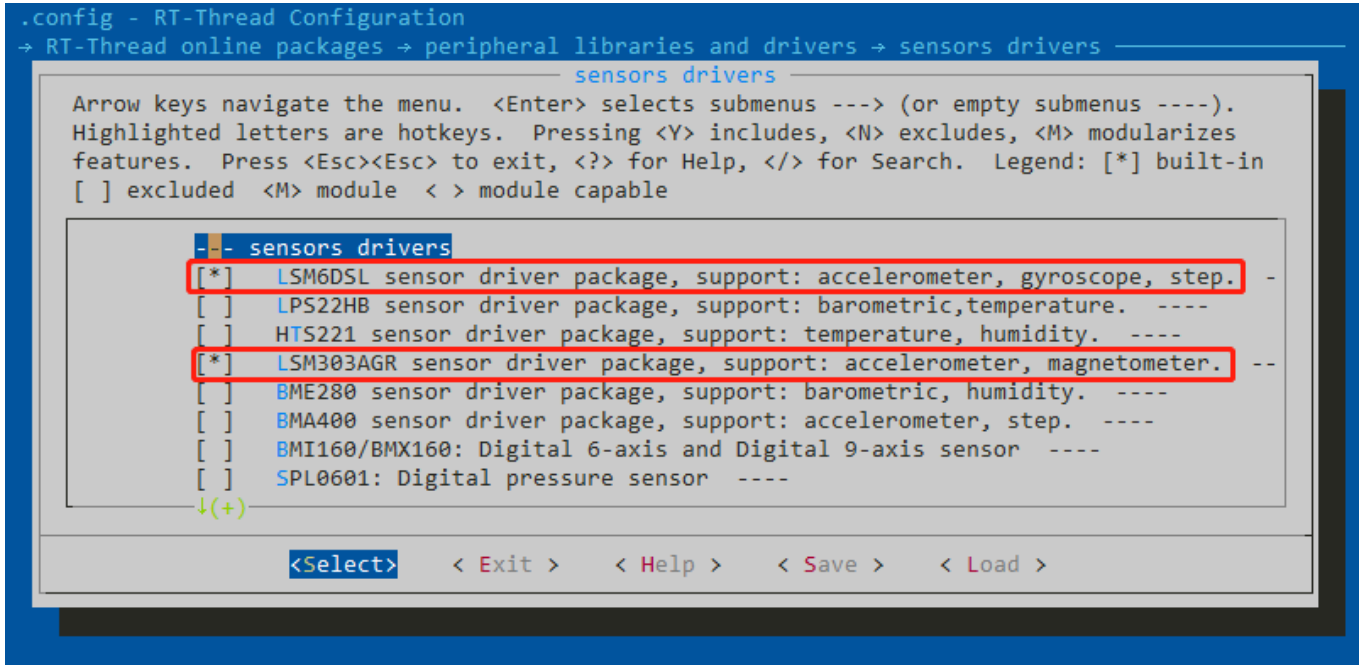
- 关闭并重新打开Env，执行命令 ‘pkgs --upgrade’ 更新软件包索引

```
blue@DESKTOP-LG7HM9K F:\rt-thread\bsp\stm32\stm32f411-st-nucleo
> pkgs --upgrade
Begin to upgrade env packages.
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 8 (delta 5), reused 0 (delta 0)
Unpacking objects: 100% (8/8), done.
From https://gitee.com/RT-Thread-Mirror/packages
 * branch            HEAD       -> FETCH_HEAD
================================>  Env packages upgrade done

Begin to upgrade env scripts.
From https://gitee.com/RT-Thread-Mirror/env
 * branch            HEAD       -> FETCH_HEAD
================================>  Env scripts upgrade done
```

获取软件包

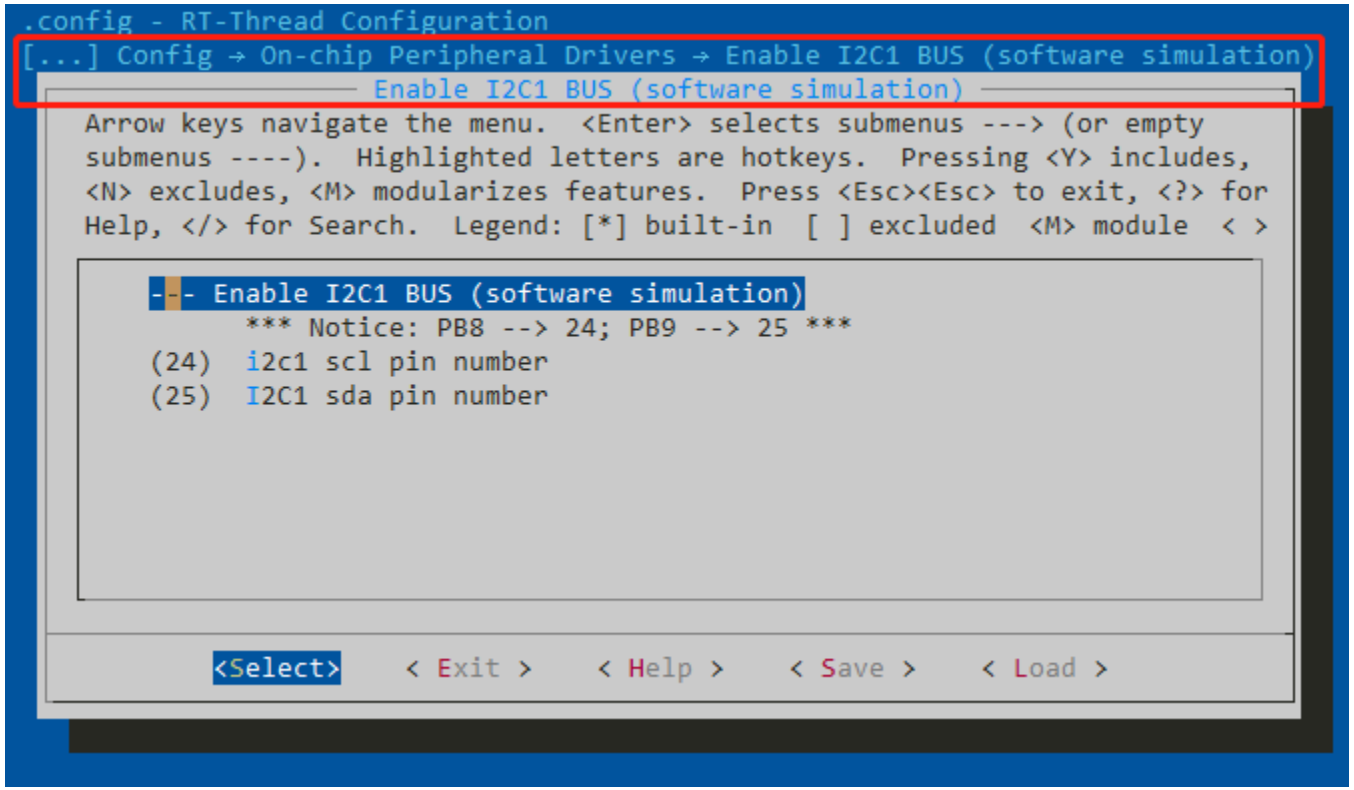
- 在menuconfig配置工具中，开启对应的sensor设备驱动

RT-Thread online packages --->
peripheral libraries and drivers --->
[*] sensors drivers --->



开启传感器需要的IIC设备

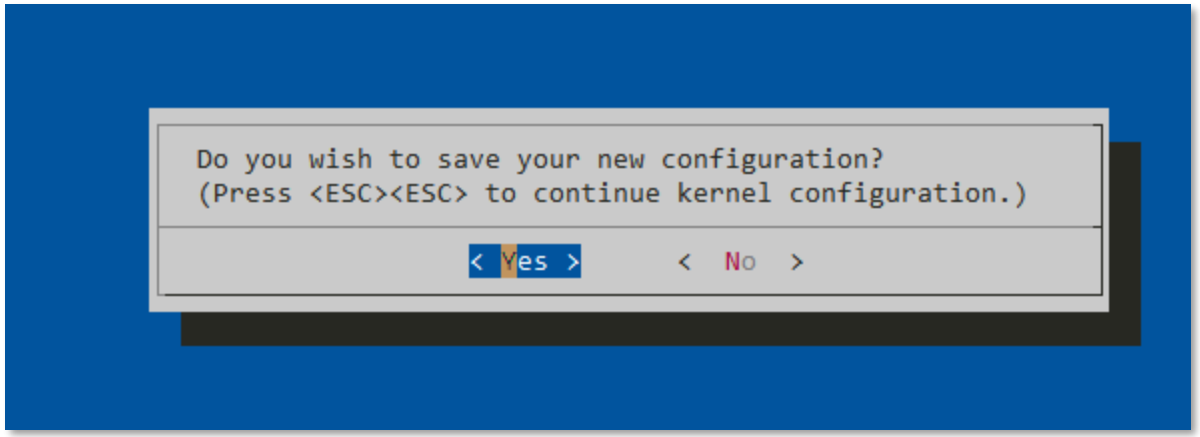
- 开启片上外设的I2C1



RT-Thread

退出并保存配置

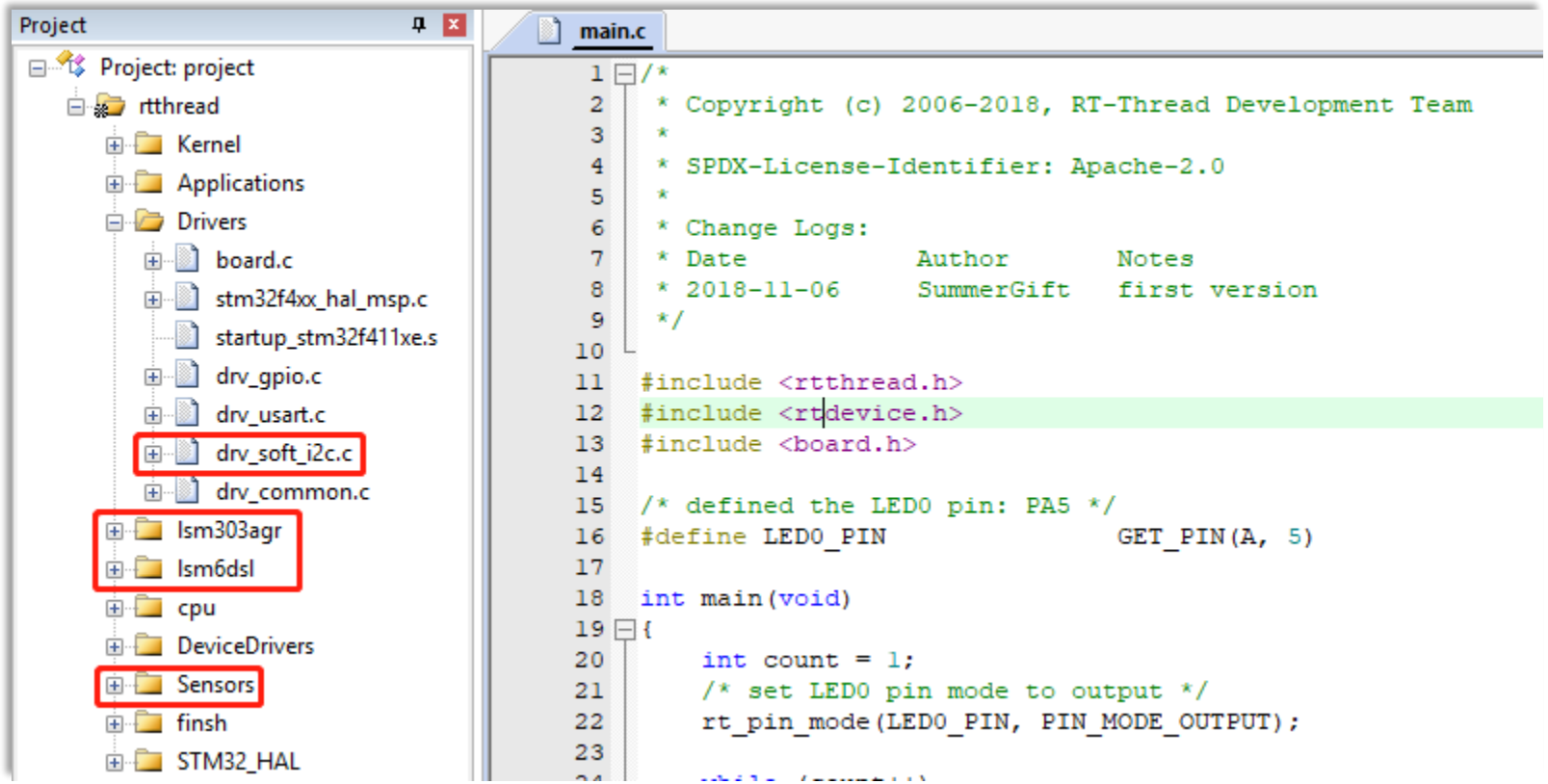
狂按 ESC 键退出，最后选中 YES 保存配置。



RT-Thread

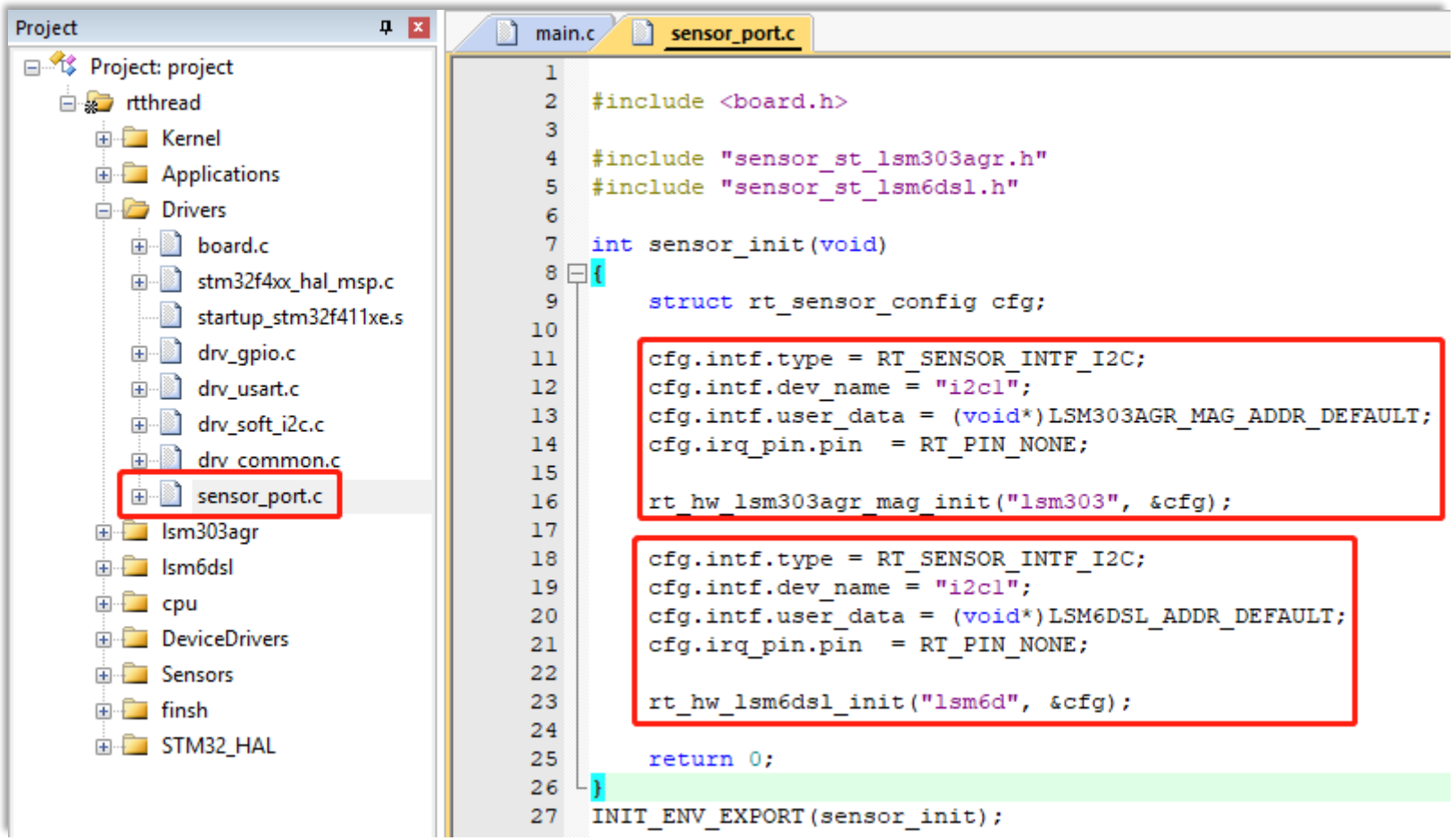
生成工程

- 然后 pkgs –update 更新软件包，scons –target=mdk5 重新生成工程。



添加port文件

- 将 sensor_port.c 添加到工程



RT-Thread

编译下载，测试传感器

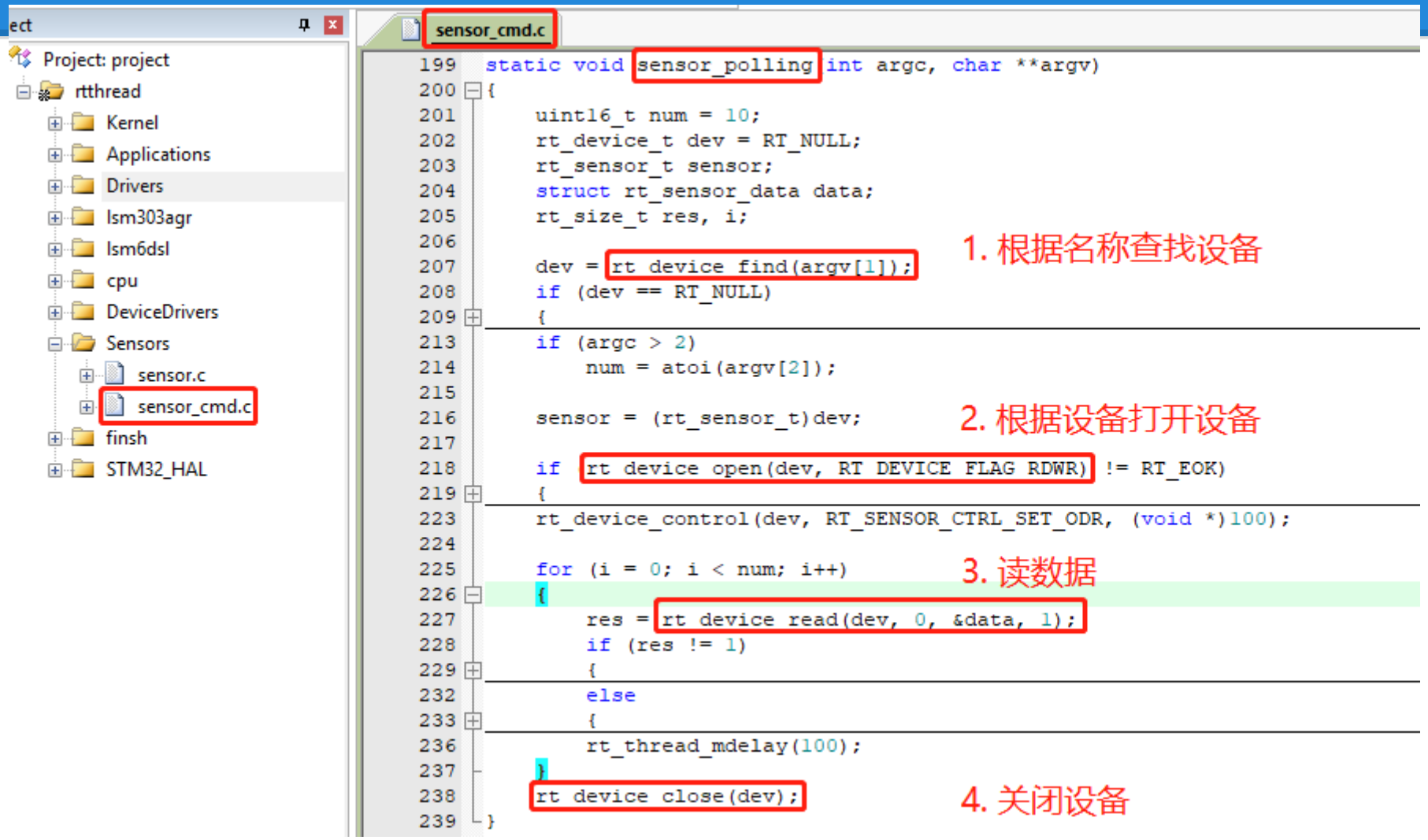
```
msh >list_device
device           type                 ref count
-------- -------------------- ----------
step_lsm Sensor Device        0
gyro_lsm Sensor Device        0
acce_lsm Sensor Device        0
mag_lsm3 Sensor Device        0
i2c1     I2C Bus              0
uart2    Character Device     2
pin      Miscellaneous Device 0
msh >sensor polling mag lsm3
[D/sensor.st.lsm303agr.mag] set power normal
[D/sensor.st.lsm303agr.mag] set odr 100
[I/sensor.cmd] num:  0, x: -177, y:  340, z: -684, timestamp:31220
[I/sensor.cmd] num:  1, x: -228, y:  403, z: -553, timestamp:31327
[I/sensor.cmd] num:  2, x: -220, y:  399, z: -549, timestamp:31435
[I/sensor.cmd] num:  3, x: -222, y:  397, z: -553, timestamp:31543
[I/sensor.cmd] num:  4, x: -234, y:  403, z: -550, timestamp:31651
[I/sensor.cmd] num:  5, x: -223, y:  396, z: -558, timestamp:31759
[I/sensor.cmd] num:  6, x: -231, y:  397, z: -553, timestamp:31867
[I/sensor.cmd] num:  7, x: -234, y:  405, z: -556, timestamp:31975
[I/sensor.cmd] num:  8, x: -231, y:  393, z: -558, timestamp:32083
[I/sensor.cmd] num:  9, x: -225, y:  405, z: -555, timestamp:32191
[D/sensor.st.lsm303agr.mag] set power down
msh >
```

注册好的传感器设备

测试传感器

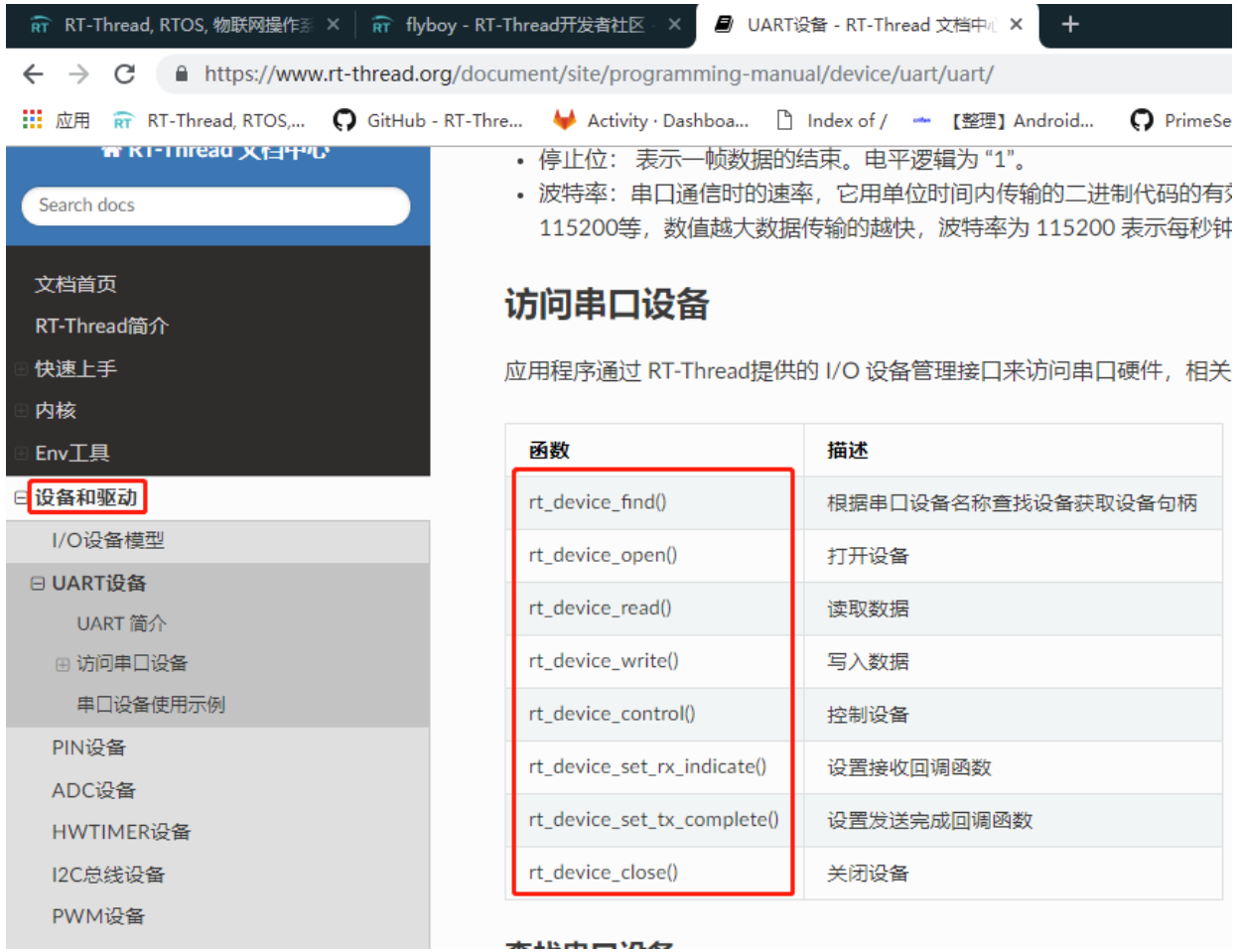
RT-Thread

API讲解

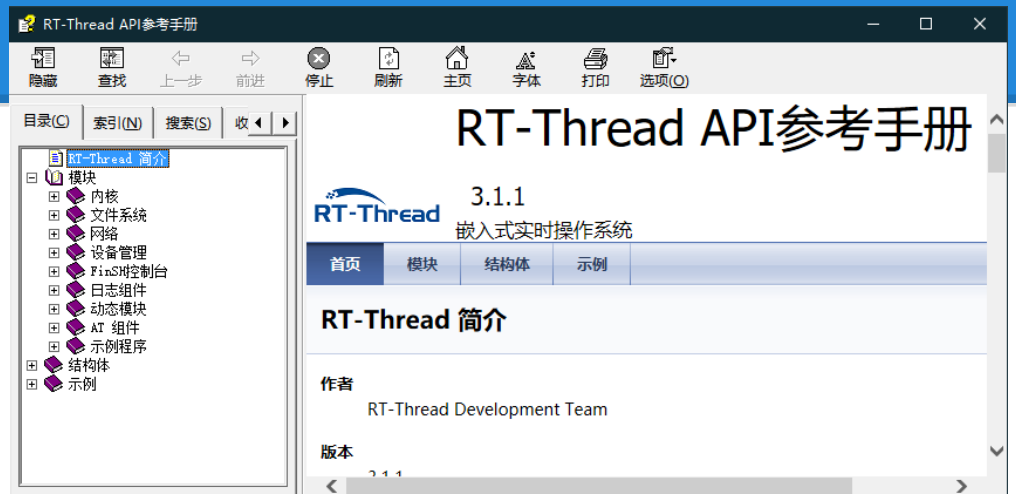


RT-Thread

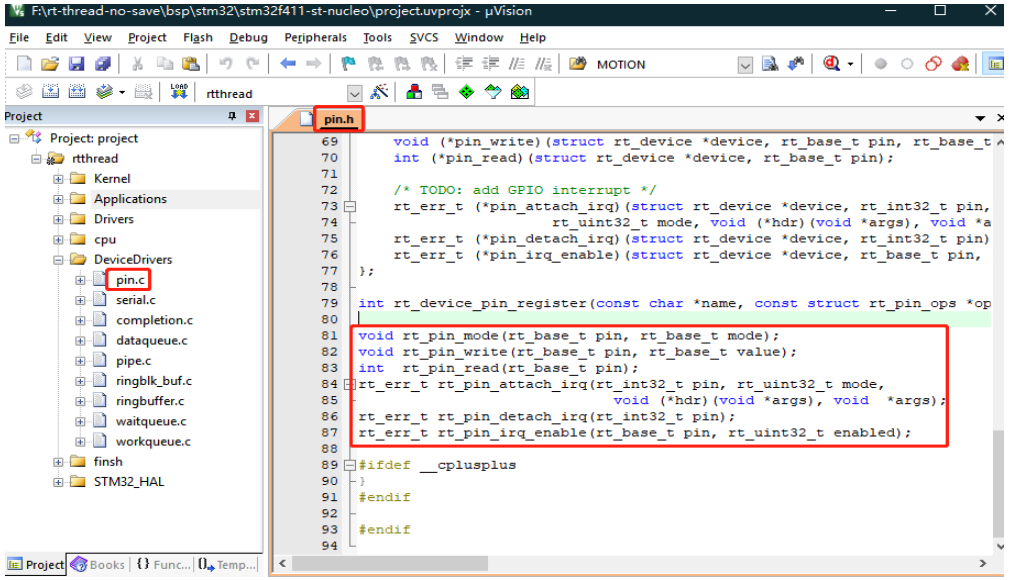
如何查找API



1. 文档中心



2. API手册



3. 源代码

RT-Thread

实验五、将传感器数据推送到本地服务器

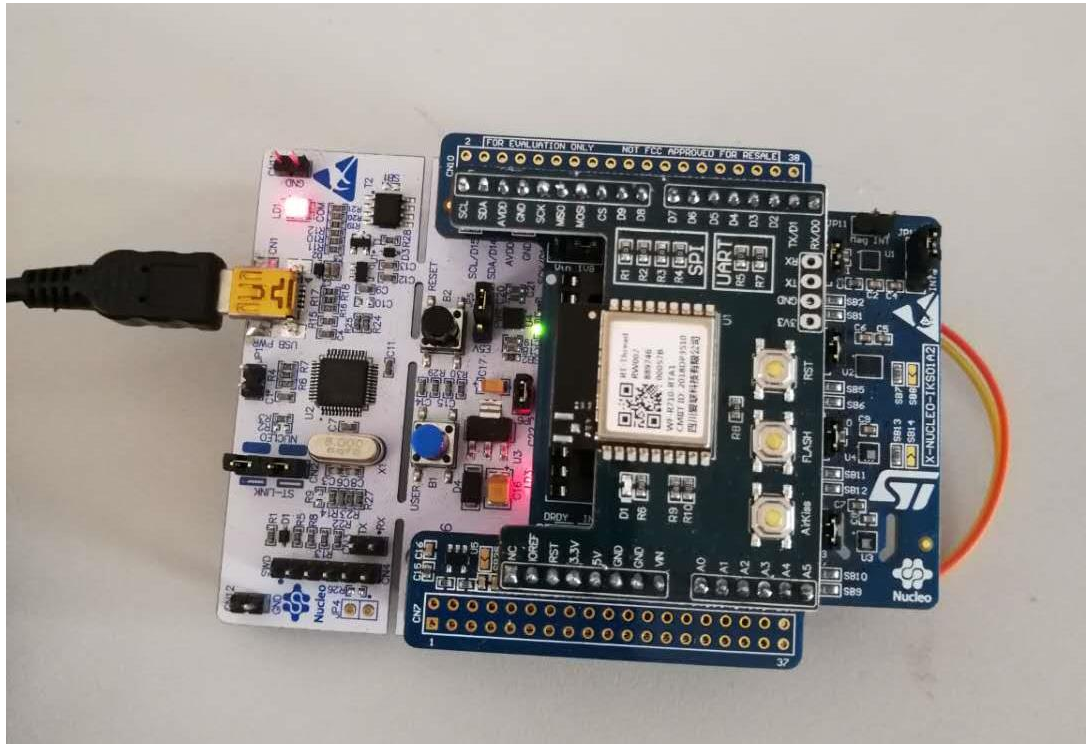
将传感器数据推送到本地服务器

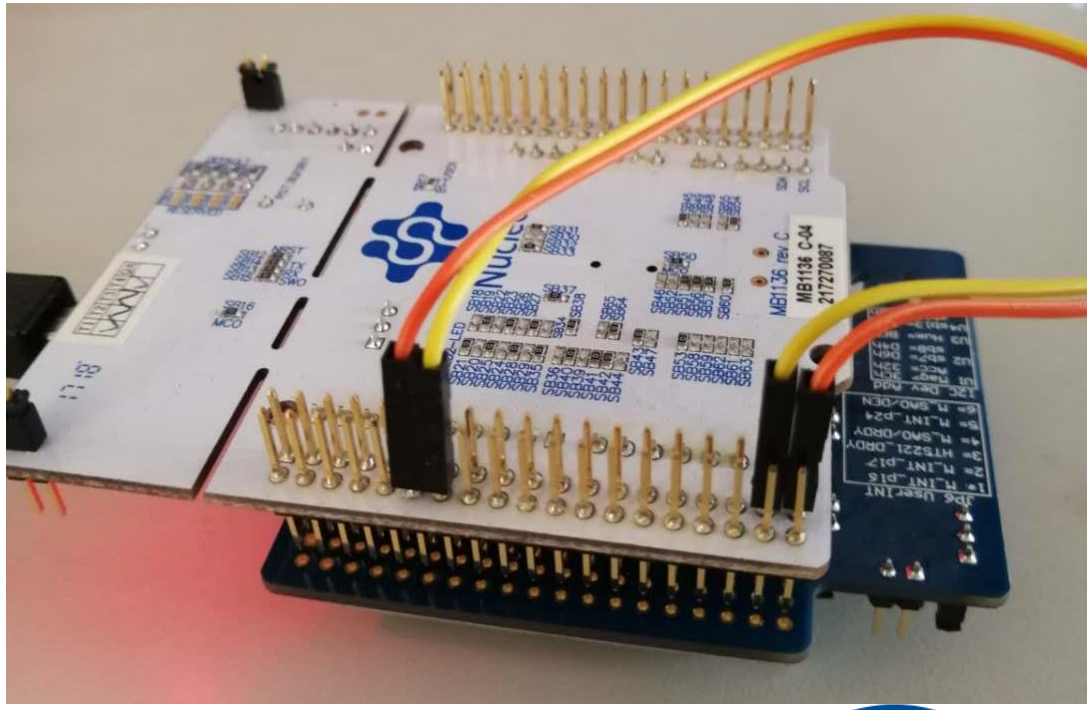
- 硬件连接
- 配置工程
- 添加示例代码
- 编译运行

RT-Thread

硬件连接

- RW007插入传感器扩展板上
- 用杜邦线分别连接PA2、PA3到A11、A12





RT-Thread

将传感器数据推送到本地服务器

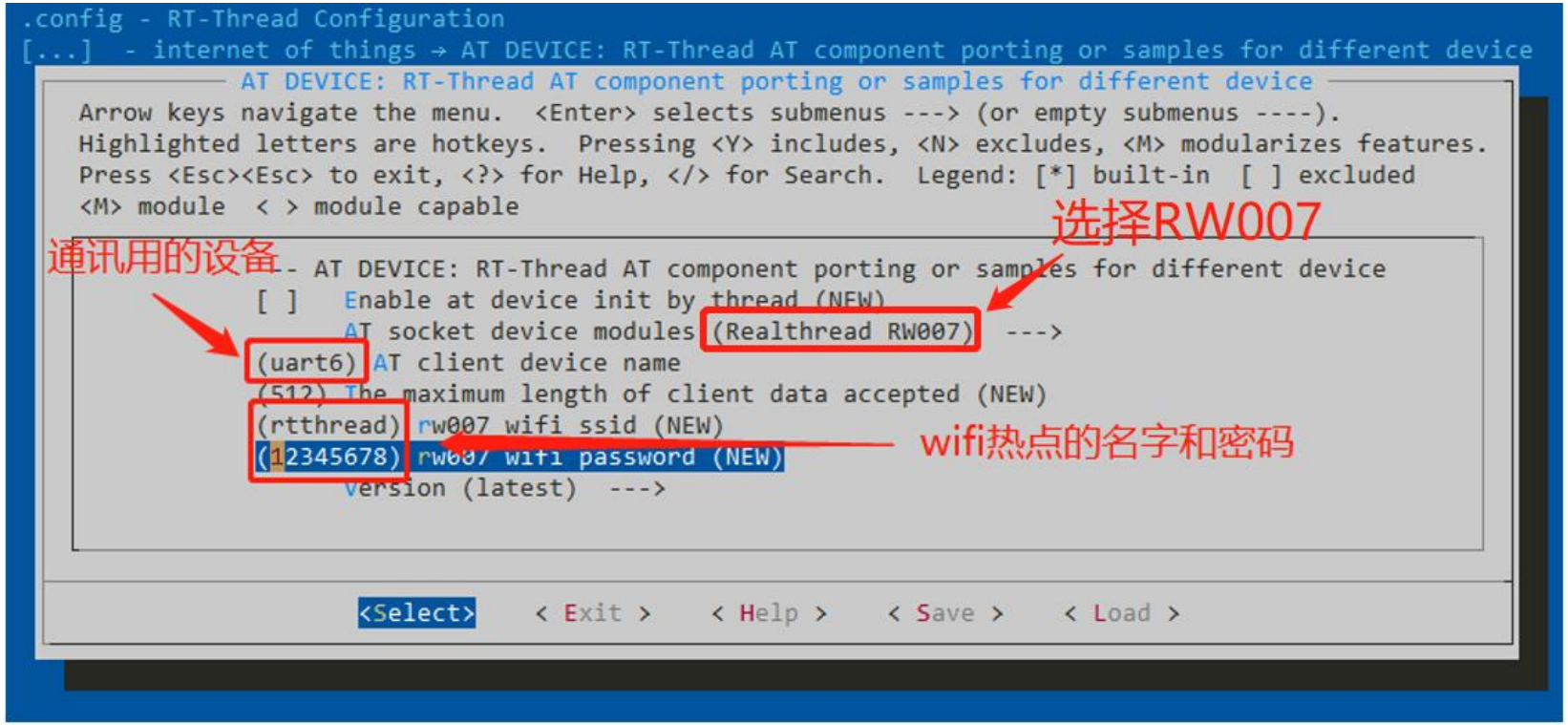
- 硬件连接
- 配置工程
- 添加示例代码
- 编译运行

RT-Thread

配置工程

在IoT软件包里找到 AT device 菜单，进入配置。

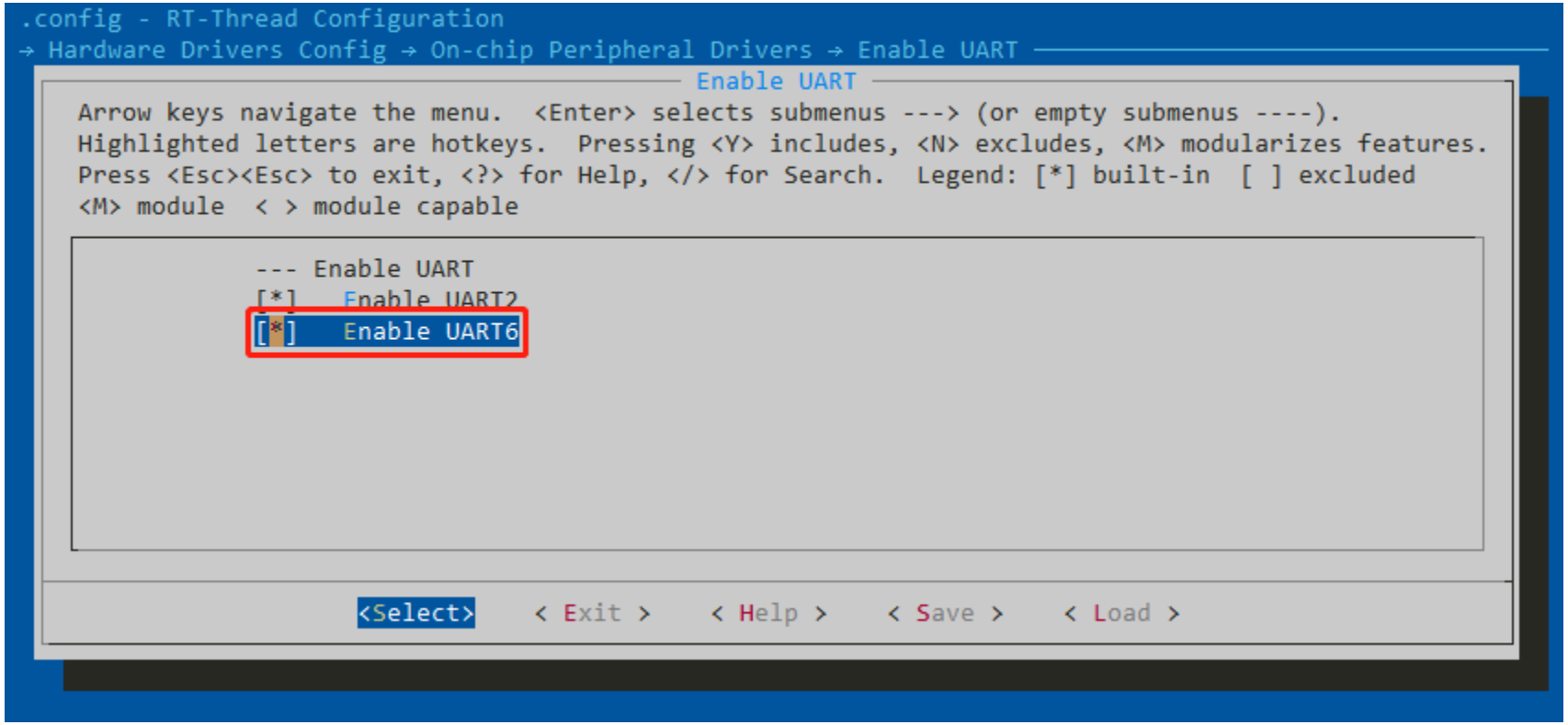
RT-Thread online packages --->
IoT - internet of things --->
[*] AT DEVICE: RT-Thread AT component porting or samples for differ



注意：请使用自己手机开启热点。

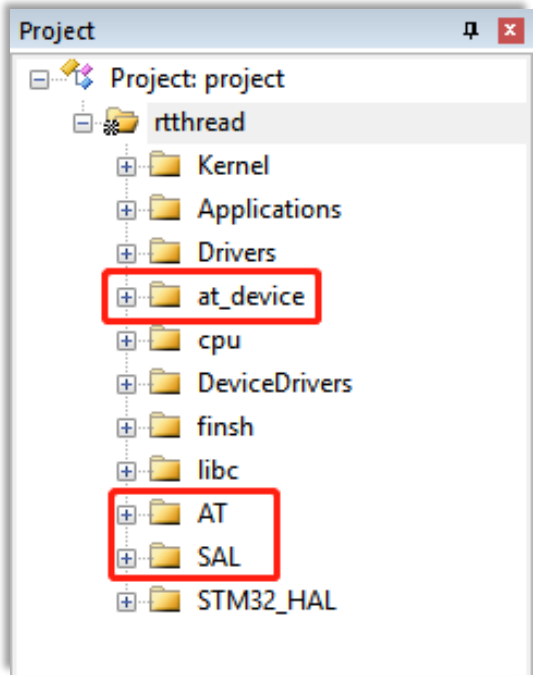
配置工程

- 开启通讯用的设备 uart6



生成工程

- pkgs --update 更新软件包，scons --target=mdk5 重新生成工程。



RT-Thread

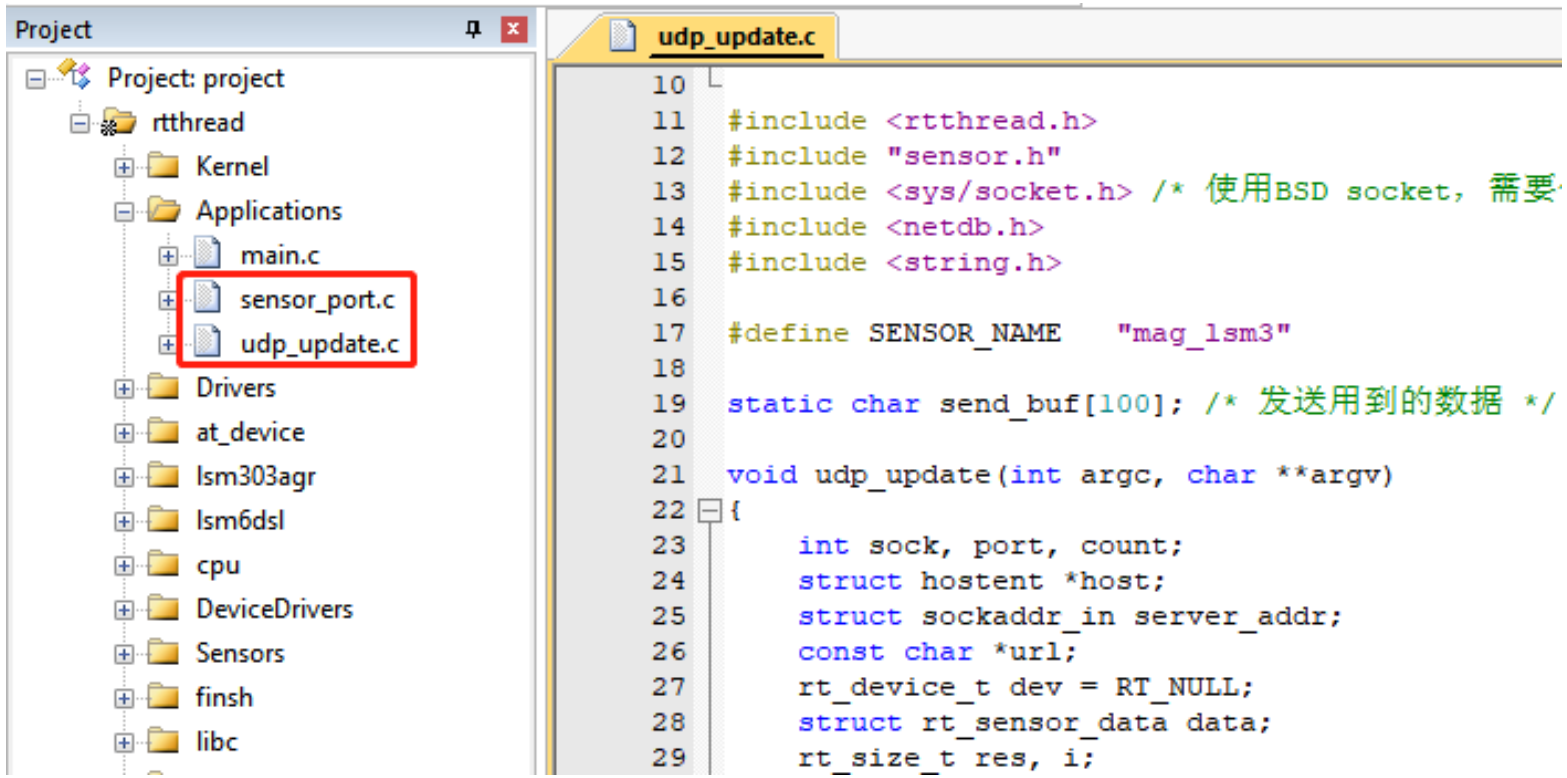
将传感器数据推送到本地服务器

- 硬件连接
- 配置工程
- 添加示例代码
- 编译运行

RT-Thread

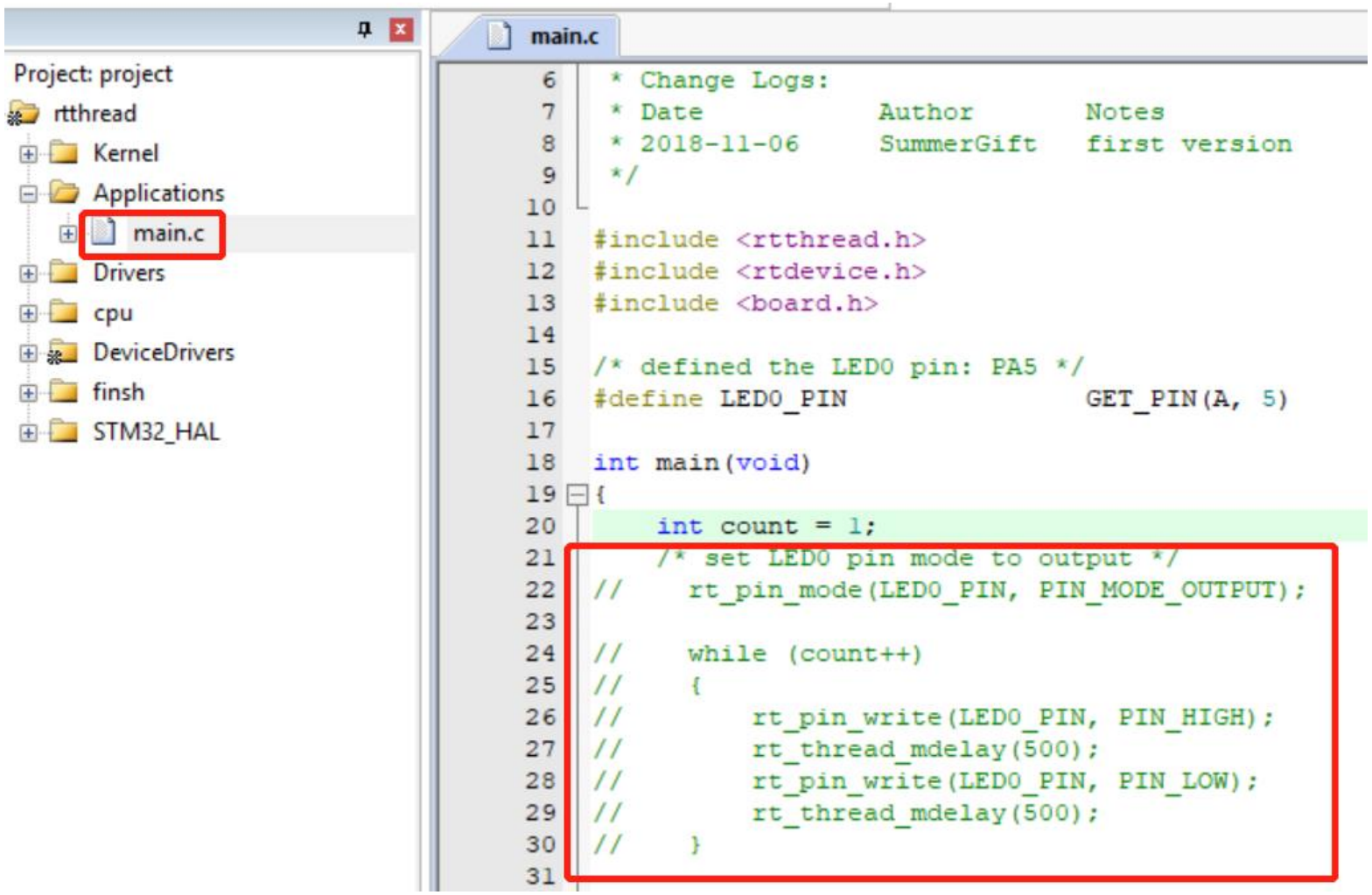
添加示例代码

- 添加sensor_port.c 和udp_update.c两个文件到工程。



RT-Thread

屏蔽mian.c中闪灯程序



RT-Thread

将传感器数据推送到本地服务器

- 硬件连接
- 配置工程
- 添加示例代码
- 编译运行

RT-Thread

编译运行

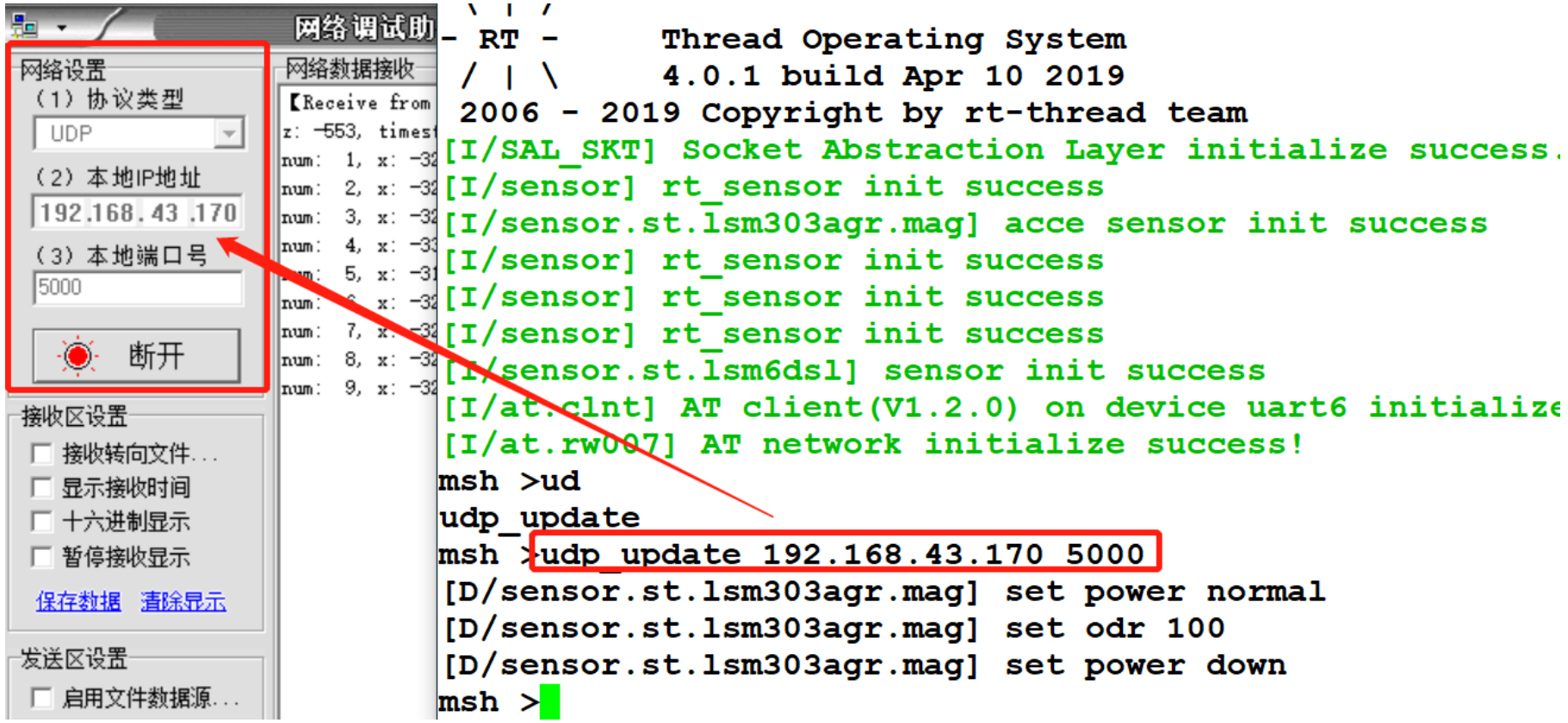
- 编译下载程序。
- 先复位RW007，再复位开发板。
- 笔记本电脑也连接手机热点，让电脑和开发板处于同一网络下。

```
 \ | /
- RT -     Thread Operating System
 / | \     4.0.1 build Apr 10 2019
 2006 - 2019 Copyright by rt-thread team
[I/SAL_SKT] Socket Abstraction Layer initialize success.
[I/sensor] rt_sensor init success
[I/sensor.st.lsm303agr.mag] acce sensor init success
[I/sensor] rt_sensor init success
[I/sensor] rt_sensor init success
[I/sensor] rt_sensor init success
[I/sensor.st.lsm6dsl] sensor init success
[I/at.clnt] AT client(V1.2.0) on device uart6 initialize success.
[I/at.rw007] AT network initialize success!
msh >
```

RT-Thread

编译运行

打开网络调试助手，开启UDP服务器，然后执行示例程序。



演示：上传数据到OneNet云

演示：上传数据到OneNet云

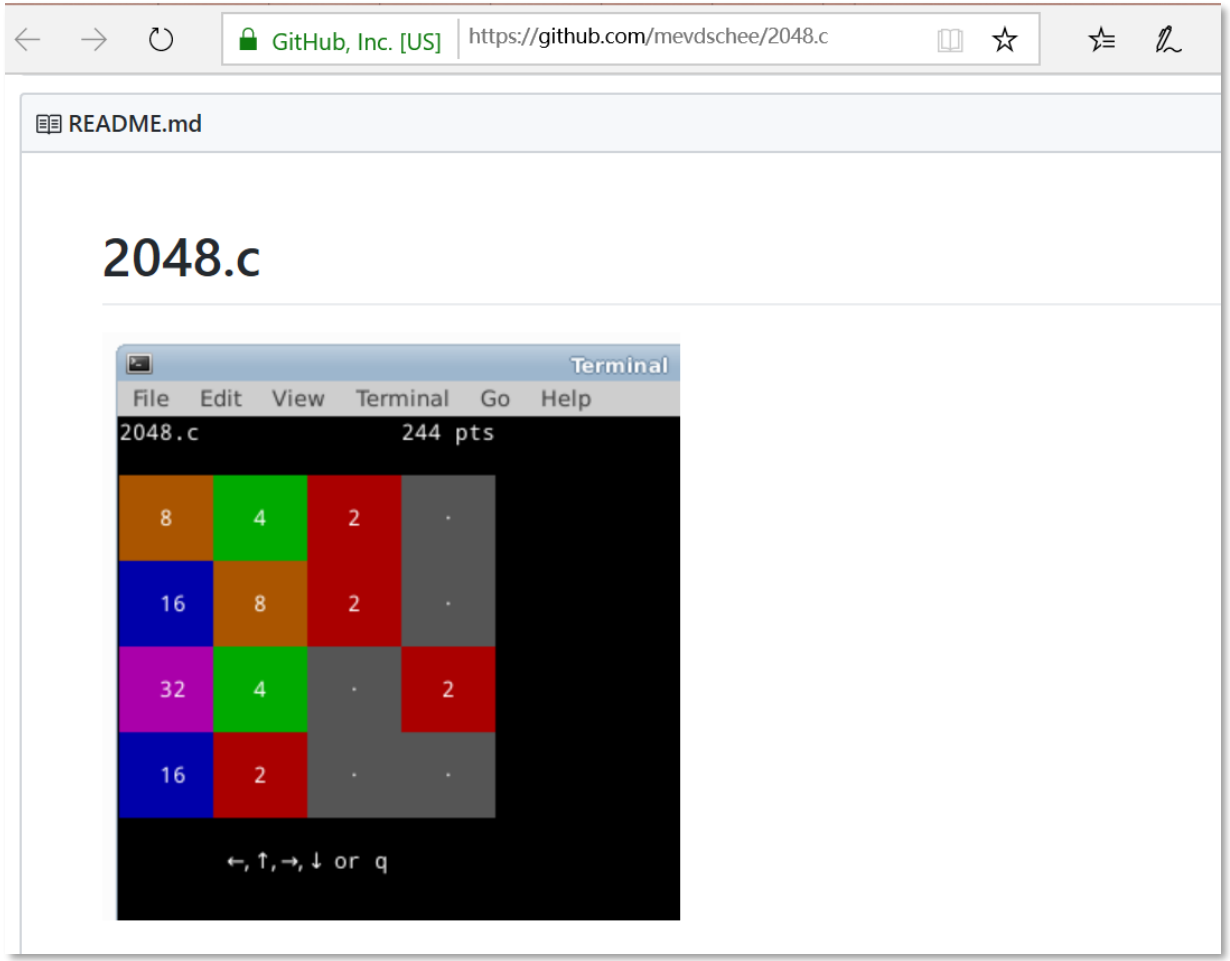
- 参考连接OneNet云的教程： https://www.rt-thread.org/document/site/tutorial/qemu-network/onenet/onenet/

RT-Thread

演示：移植游戏2048

演示：移植游戏2048

- 仓库地址：https://github.com/mevdschee/2048.c



RT-Thread

开启虚拟文件系统

```
.config - RT-Thread Configuration
→ RT-Thread Components → Device virtual file system
                    Device virtual file system
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for
  Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

      [*] Using device virtual file system
      [*]   Using working directory
      (2)   The maximal number of mounted file system
      (2)   The maximal number of file system type
      (16)  The maximal number of opened files
      [ ]   Using mount table for file system
      [ ]   Enable elm-chan fatfs
      -*-   Using devfs for device objects
      [ ]   Enable ReadOnly file system on flash
      [ ]   Enable RAM file system
      ↓(+)

            <Select>    < Exit >    < Help >    < Save >    < Load >
```

RT-Thread

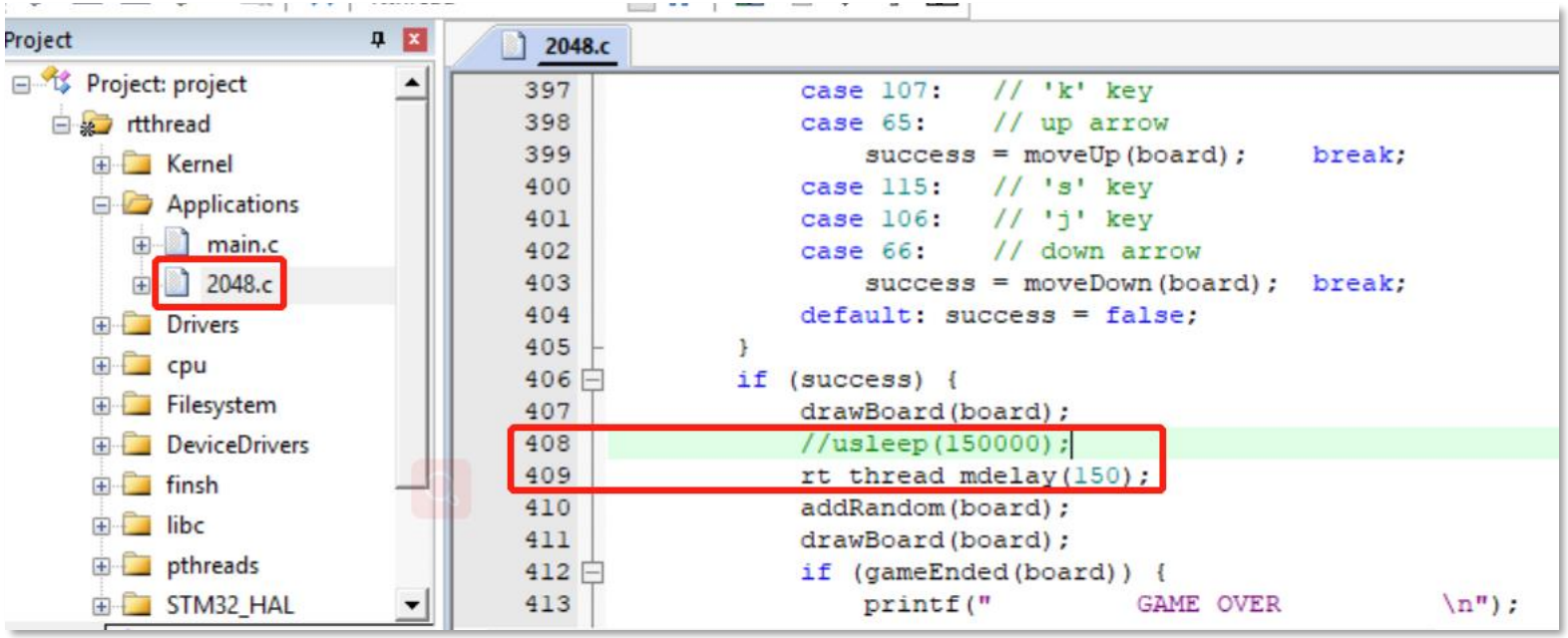
打开POSIX接口

```
.config - RT-Thread Configuration
→ RT-Thread Components → POSIX layer and C standard library
                    POSIX layer and C standard library
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for
  Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

      [*] Enable libc APIs from toolchain
      [*] Enable pthreads APIs
      [*] Enable POSIX layer for poll/select, stdin etc
      [ ]   Enable mmap() api
      [*]   Enable termios feature
      [ ]   Enable AIO
      [ ] Enable dynamic module with dlopen/dlsym/dlclose feature

            <Select>    < Exit >    < Help >    < Save >    < Load >
```

RT-Thread

添加代码

- 添加2048.c到工程
- 将usleep(150000)替换为rt_thread_mdelay(150)



添加代码

- 将mian函数替换为start_2048

```
//int main(int argc, char *argv[]) {
int start_2048(int argc, char *argv[]) {
    uint8_t board[SIZE][SIZE];
    char c;
    bool success;
```

- 导出函数start_2048到控制台

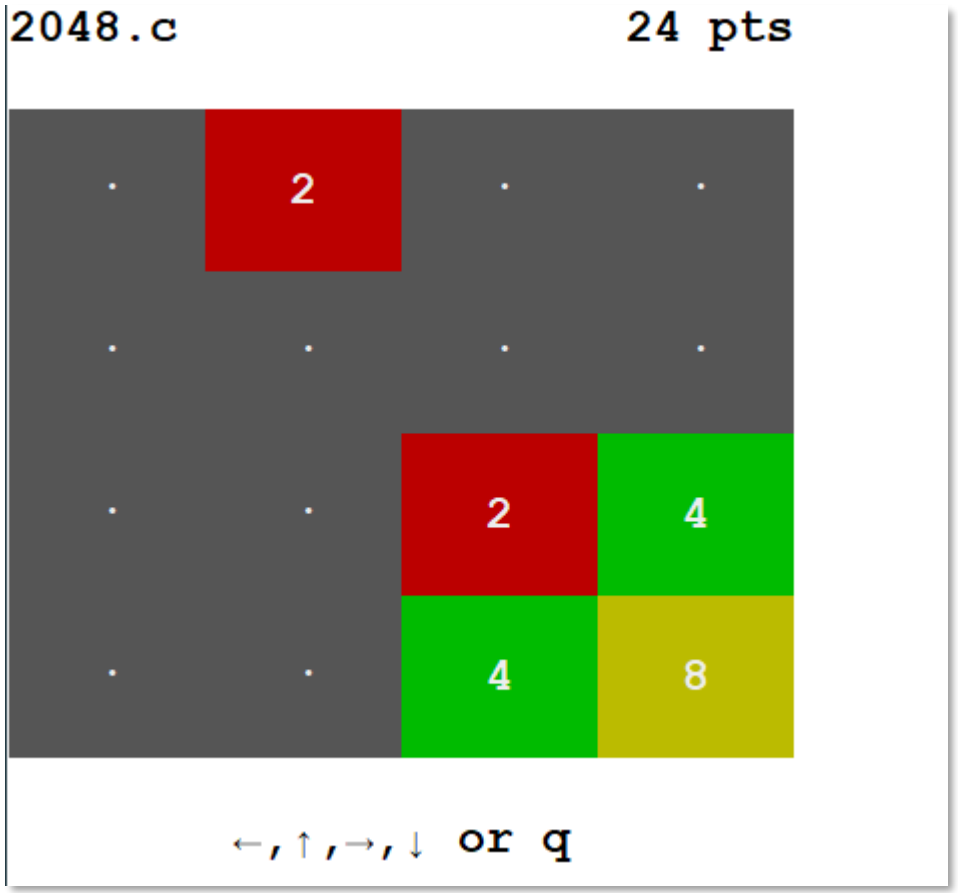
```
}
MSH_CMD_EXPORT(start_2048, start 2048)
```

RT-Thread

编译下载

运行时，输入start_2048启动游戏。

```
mkdir                    - Create the DIRECTORY
mkfs                     - format disk with file
df                       - disk free
echo                     - echo string to file
ps                       - List threads in the s
time                     - Execute command with
free                     - Show the memory usage

msh />st
start_2048
msh />start 2048
```



RT-Thread

RT-Thread