

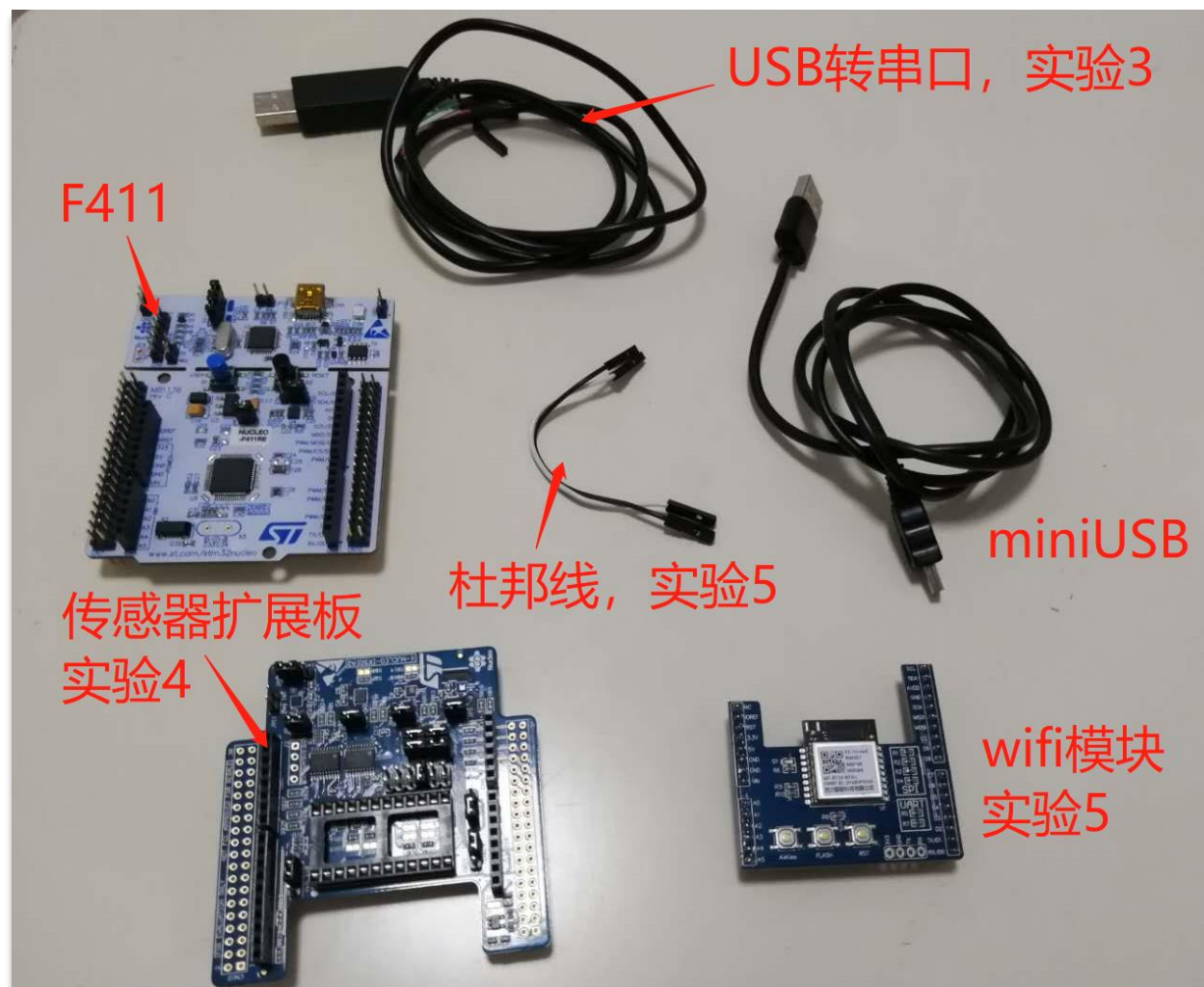


入门实战

目录

- 实验一、运行RT-Thread
- 实验二、快速搭建RT-Thread项目框架
- 实验三、适配项目开发板
- 实验四、读取传感器数据
- 实验五、将传感器数据发送到本地服务器
- 演示：上传传感器数据到OneNet云

器材介绍



The logo for RT-Thread, featuring a blue stylized wave icon above the text "RT-Thread" in a bold, blue, sans-serif font.

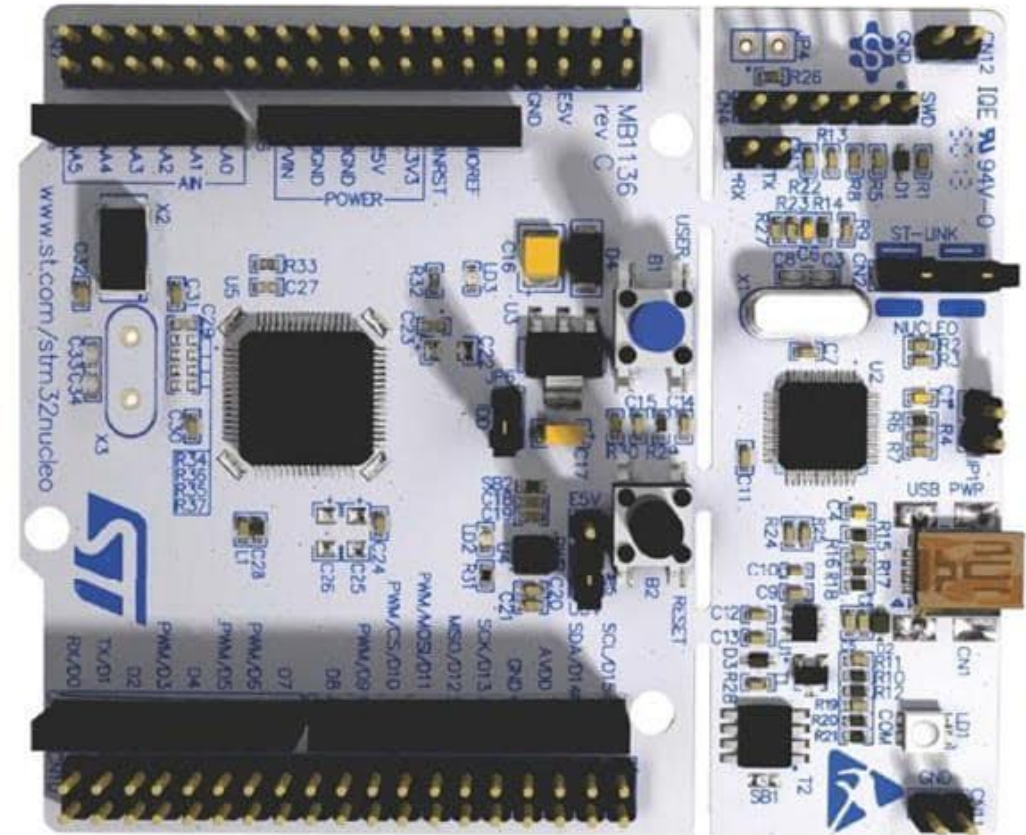
RT-Thread

实验一、运行RT-Thread

硬件-STM32F411 Nucleo-64

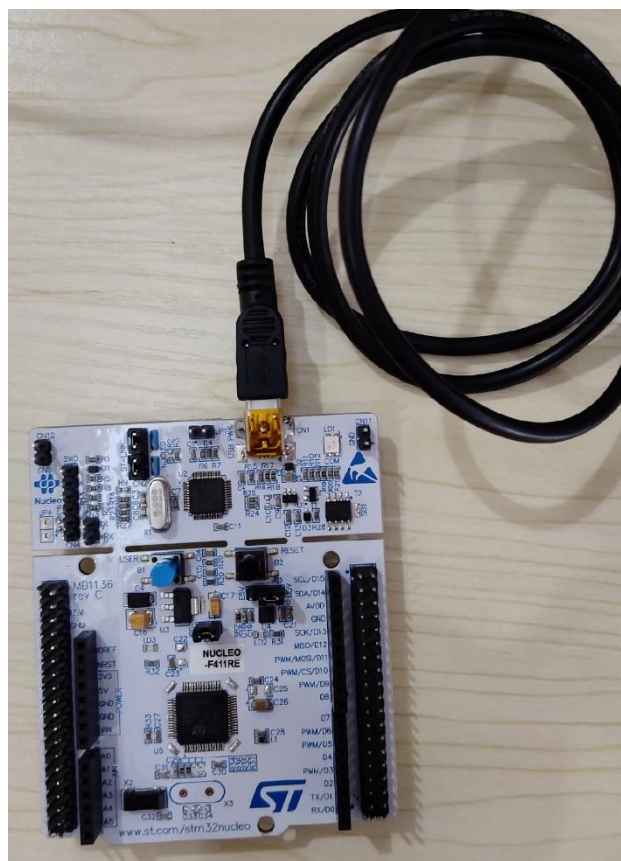
STM32F411RET6

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



硬件连接

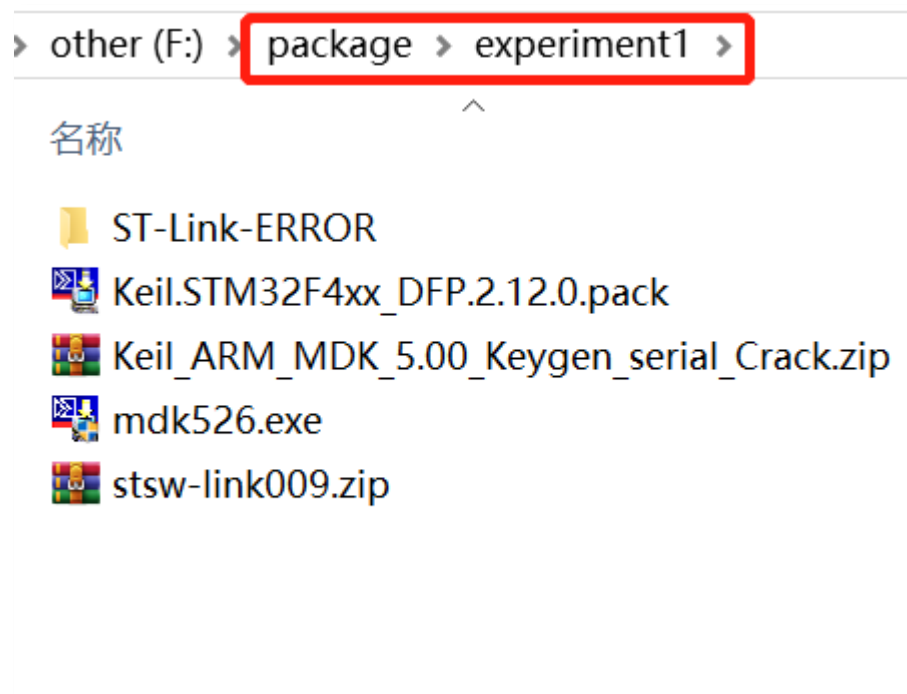
- 使用 Type-A to Mini-B 线连接开发板和 PC



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

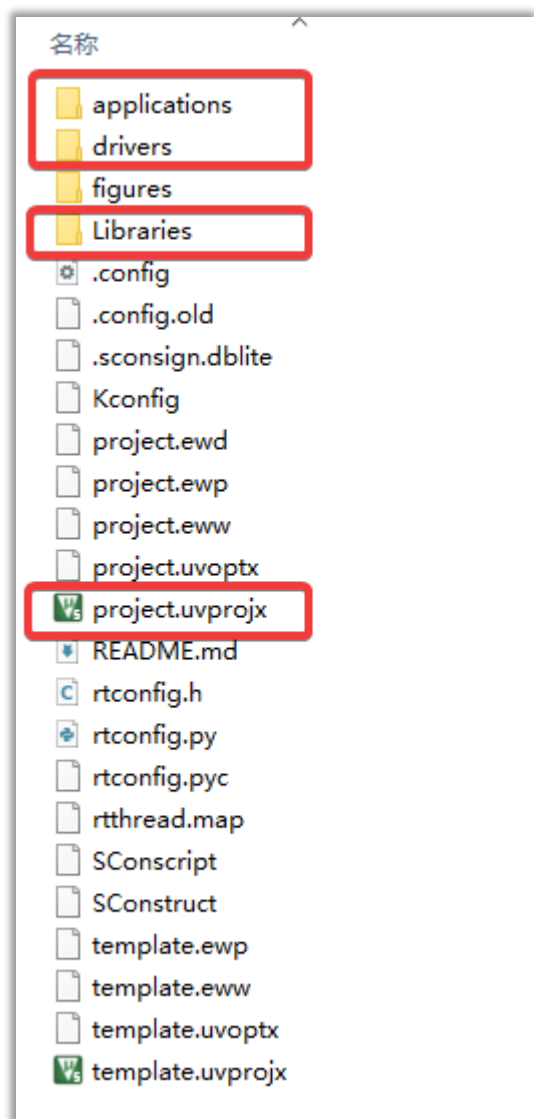
源码结构说明

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

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

工程

W:\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

Project

- Project: project
 - rt-thread_es32f065x
 - Kernel 编译
 - Applications
 - main.c
 - Drivers
 - Libraries
 - cpu
 - DeviceDrivers
 - finsh

main.c

```
8  * 2019-01-28    wangyq    the first version
9  */
10
11 #include <rtthread.h>
12 #include <rtdevice.h>
13
14 #define LED_PIN 39
15
16 int main(void)
17 {
18     int count = 1;
19     /* set PC08 pin mode to output */
20     rt_pin_mode(LED_PIN, PIN_MODE_OUTPUT);
21
22     while (count++)
23     {
24         rt_pin_write(LED_PIN, PIN_HIGH);
25         rt_thread_mdelay(500);
26         rt_pin_write(LED_PIN, PIN_LOW);
27         rt_thread_mdelay(500);
28     }
29     return RT_EOK;
30 }
31
```

工程配置

用户程序入口

Build Output

Load "F:\\rt-thread-no-save\\bsp\\es32f0654\\build\\keil\\rtthread.axf"

Erase Done.

Programming Done.

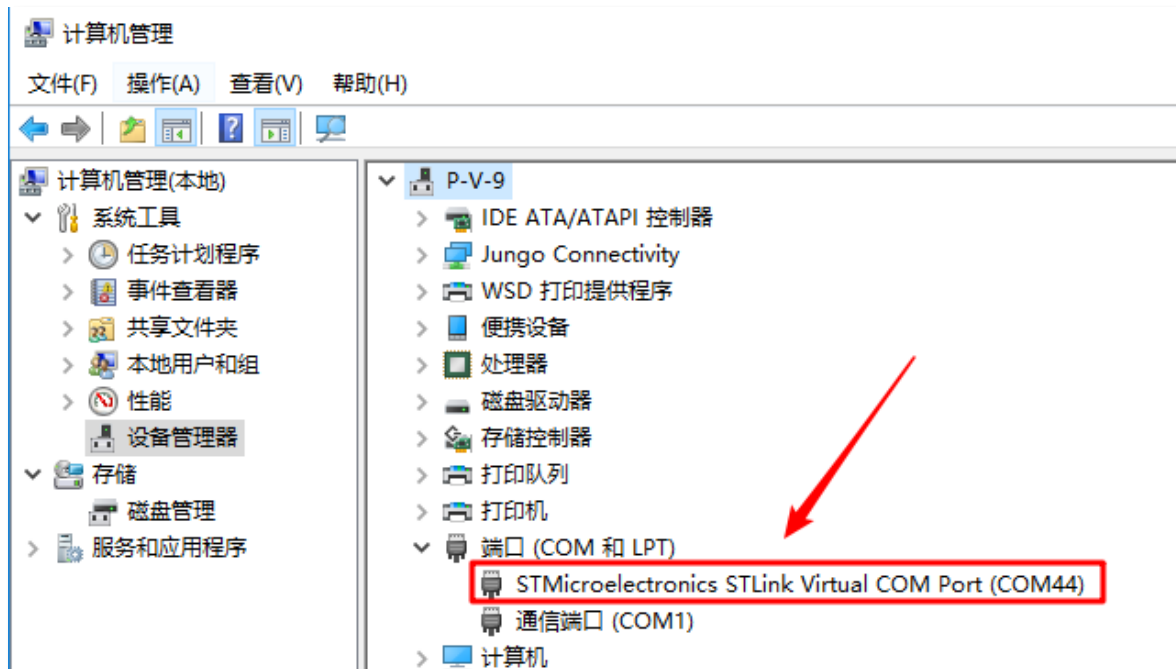
Verify OK.

Application running ...

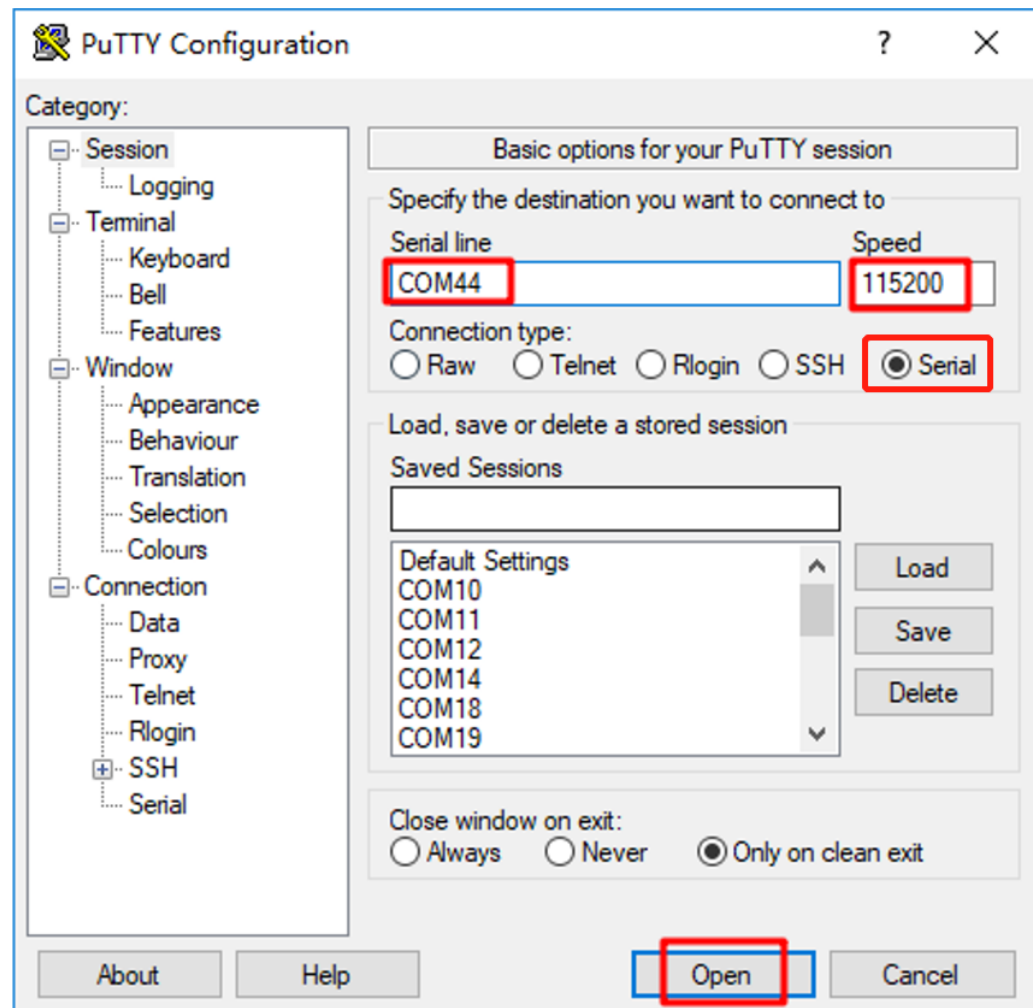
Flash Load finished at 21:23:36

CMSIS-DAP Debugger

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。



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 >
```

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.
```

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 >
```

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 >
```

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 >
```



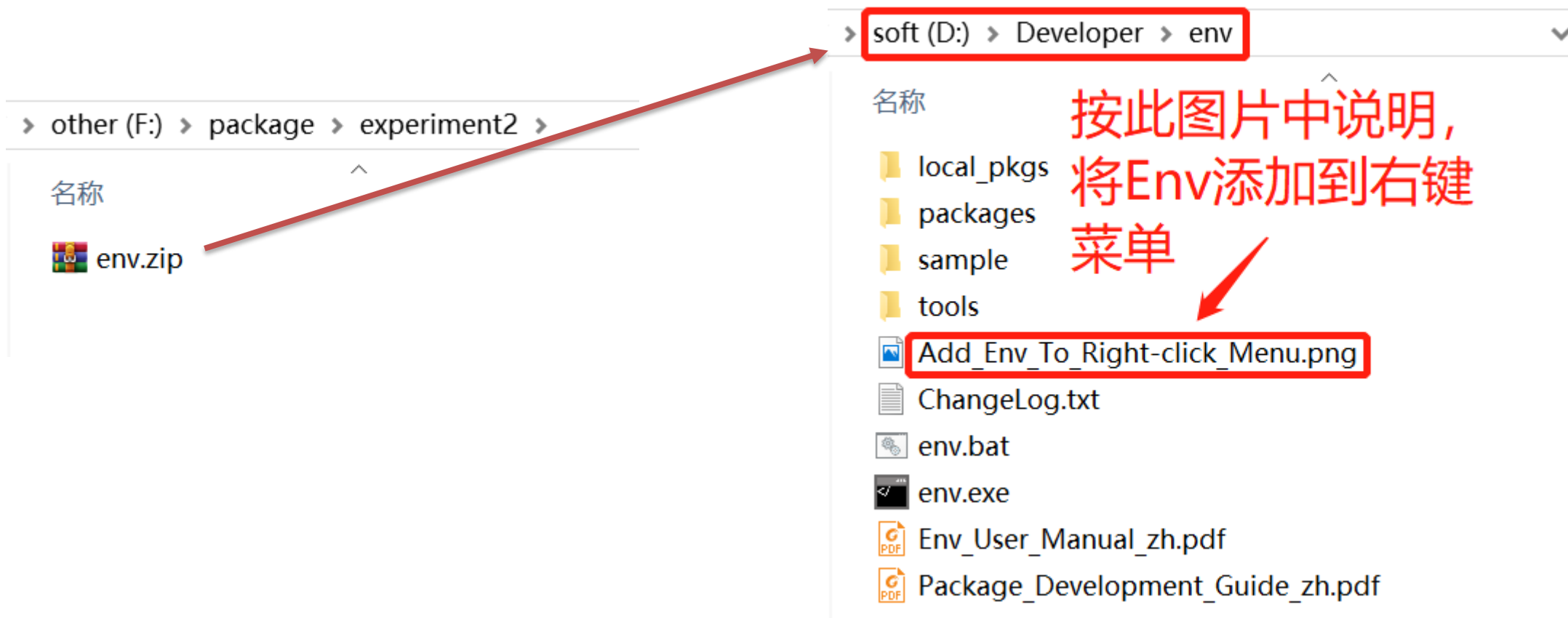

实验二、快速搭建RT-Thread项目框架

需求分析



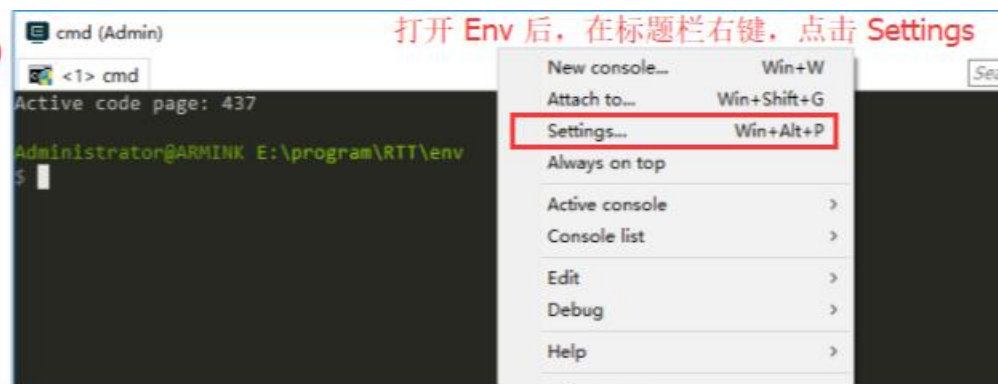
环境准备

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

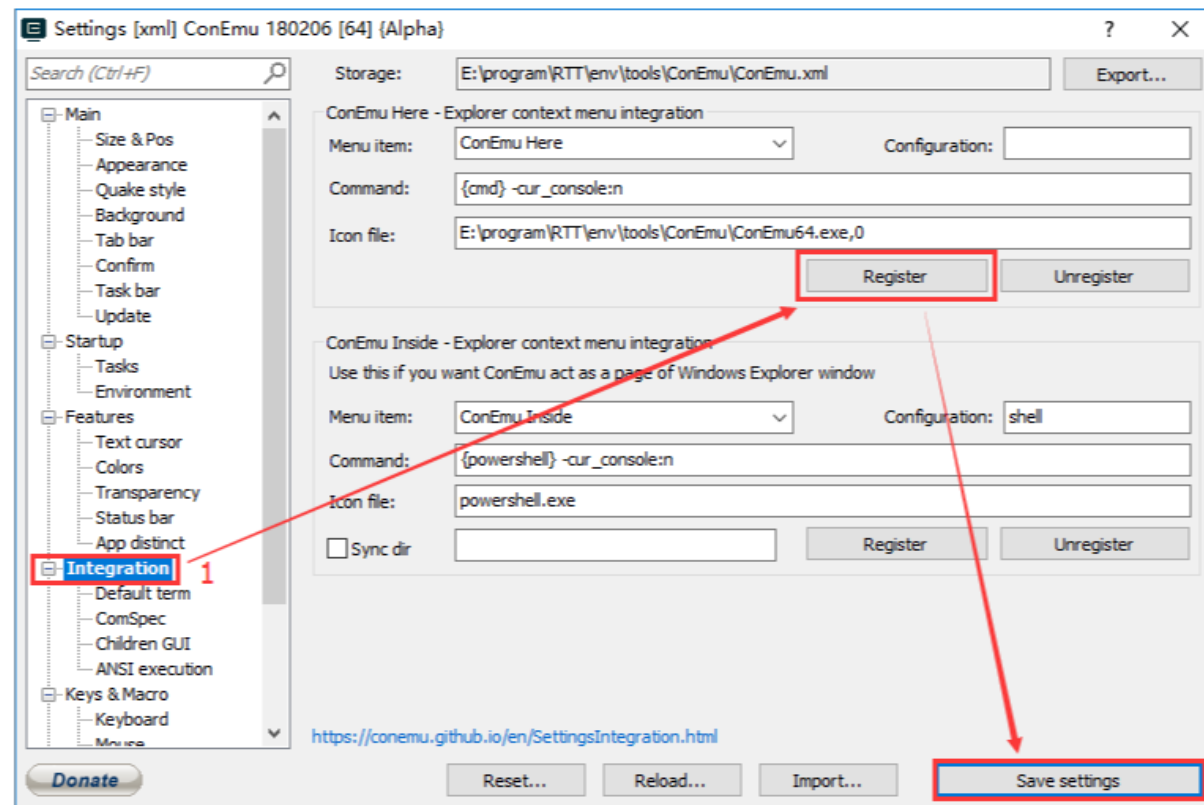


Env注册到右键快捷菜单

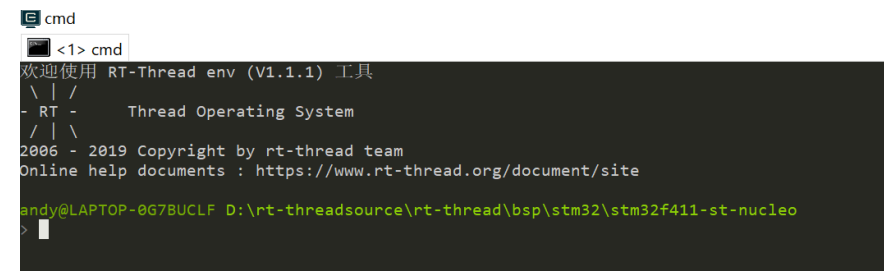
①



②

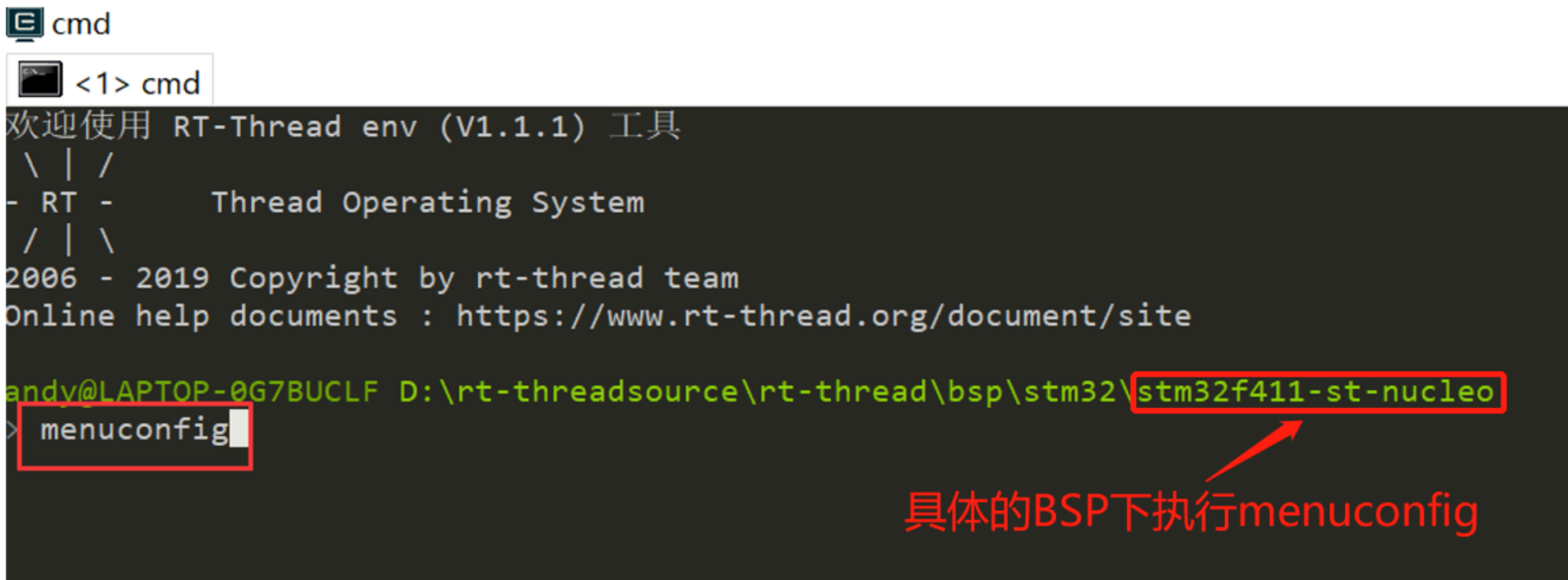


Env在具体的BSP下使用



Env在具体的BSP下使用

输入 **menuconfig** 回车



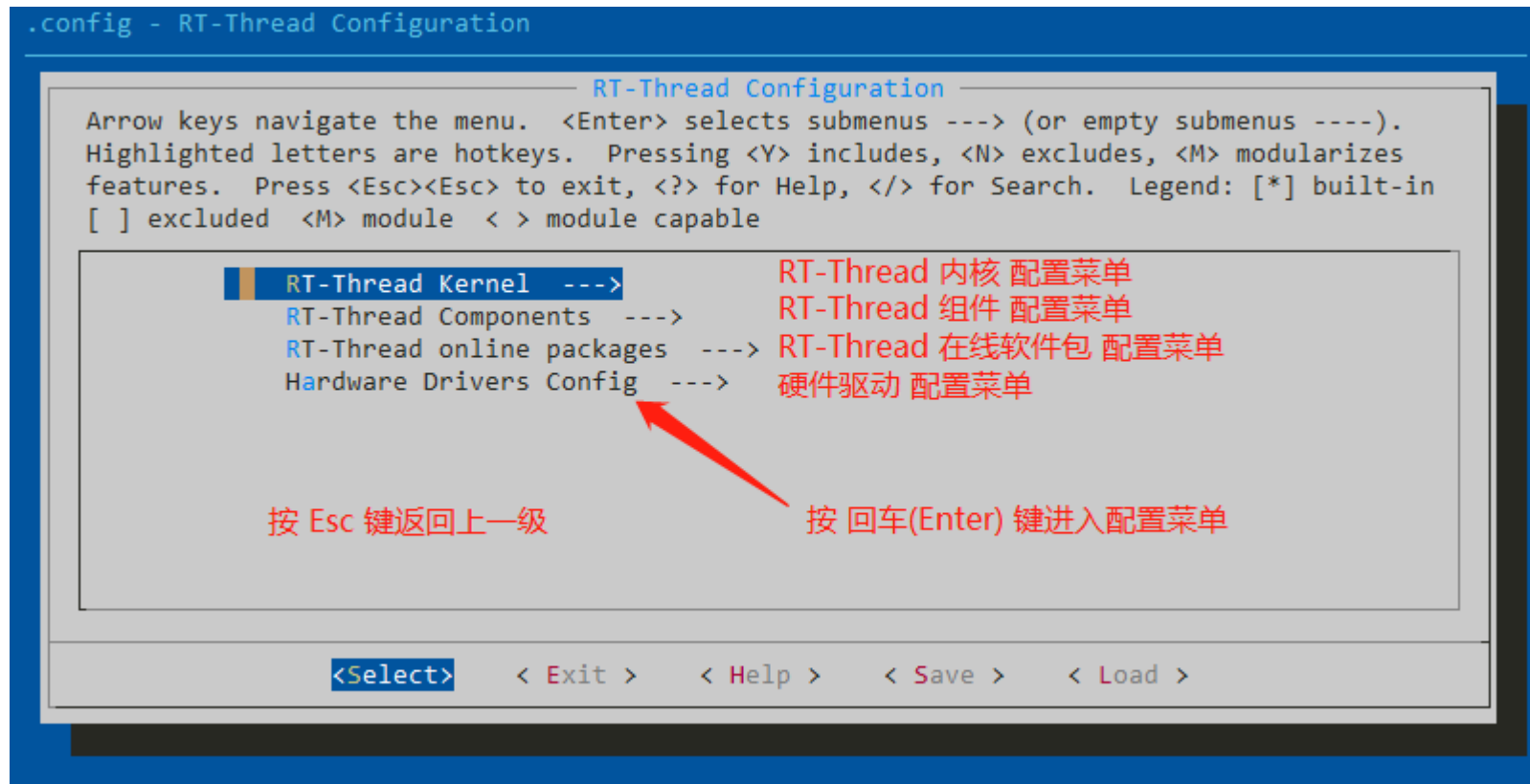
```
cmd
<1> cmd
欢迎使用 RT-Thread env (V1.1.1) 工具
\ | /
- RT - Thread Operating System
/ | \
2006 - 2019 Copyright by rt-thread team
Online help documents : https://www.rt-thread.org/document/site
andy@LAPTOP-0G7BUCLF D:\rt-threadsource\rt-thread\bsp\stm32\stm32f411-st-nucleo
> menuconfig
```

具体的BSP下执行menuconfig

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

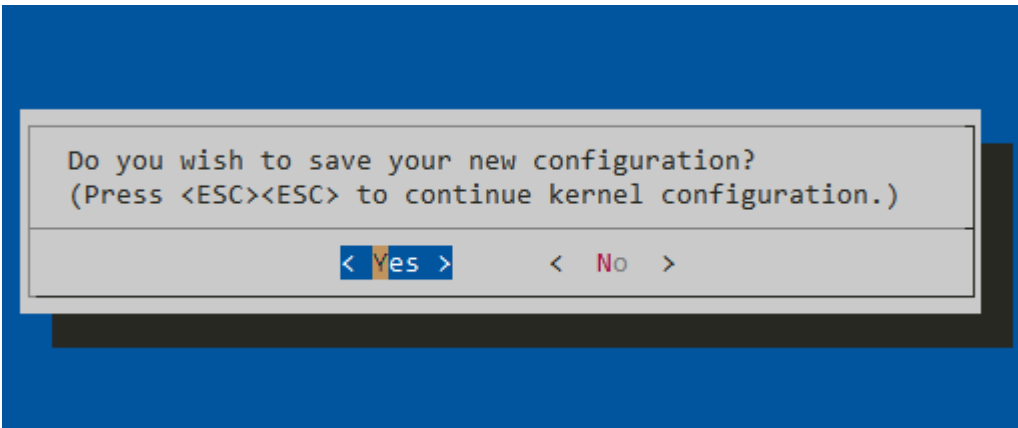
[*] Enable GPIO
[*] Enable UART --->
[*] Enable I2C1 BUS (software simulation) --->
[*] Enable SPI BUS --->
[ ] Enable pwm ----
[ ] Enable RTC ----
[ ] Enable on-chip FLASH
```

```
-- 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)
```

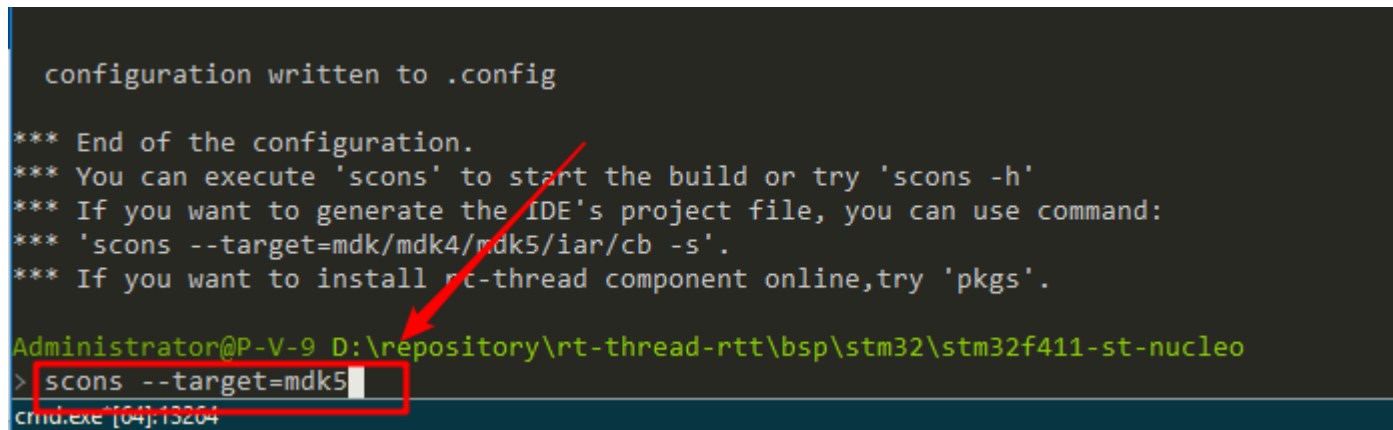

Env在具体的BSP下使用

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



Do you wish to save your new configuration?
(Press <ESC><ESC> to continue kernel configuration.)

< Yes > < No >



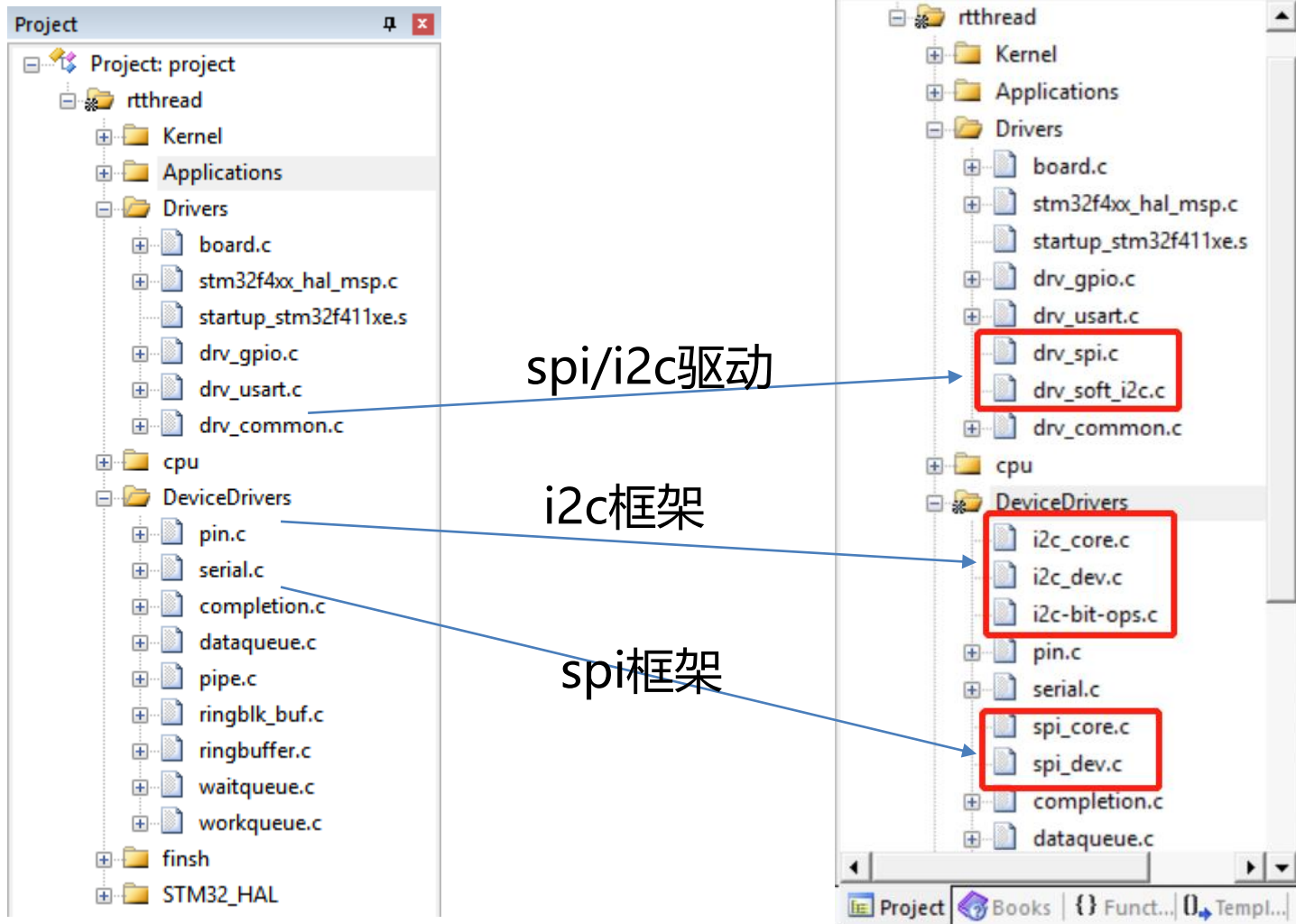
```
configuration written to .config

*** End of the configuration.
*** You can execute 'scons' to start the build or try 'scons -h'
*** If you want to generate the IDE's project file, you can use command:
*** 'scons --target=mdk/mdk4/mdk5/iar/cb -s'.
*** If you want to install rt-thread component online, try 'pkgs'.

Administrator@P-V-9 D:\repository\rt-thread-rtt\bsp\stm32\stm32f411-st-nucleo
> scons --target=mdk5
cmd.exe [64]:13204
```

Env在具体的BSP下使用

配置工程前后工程文件对比



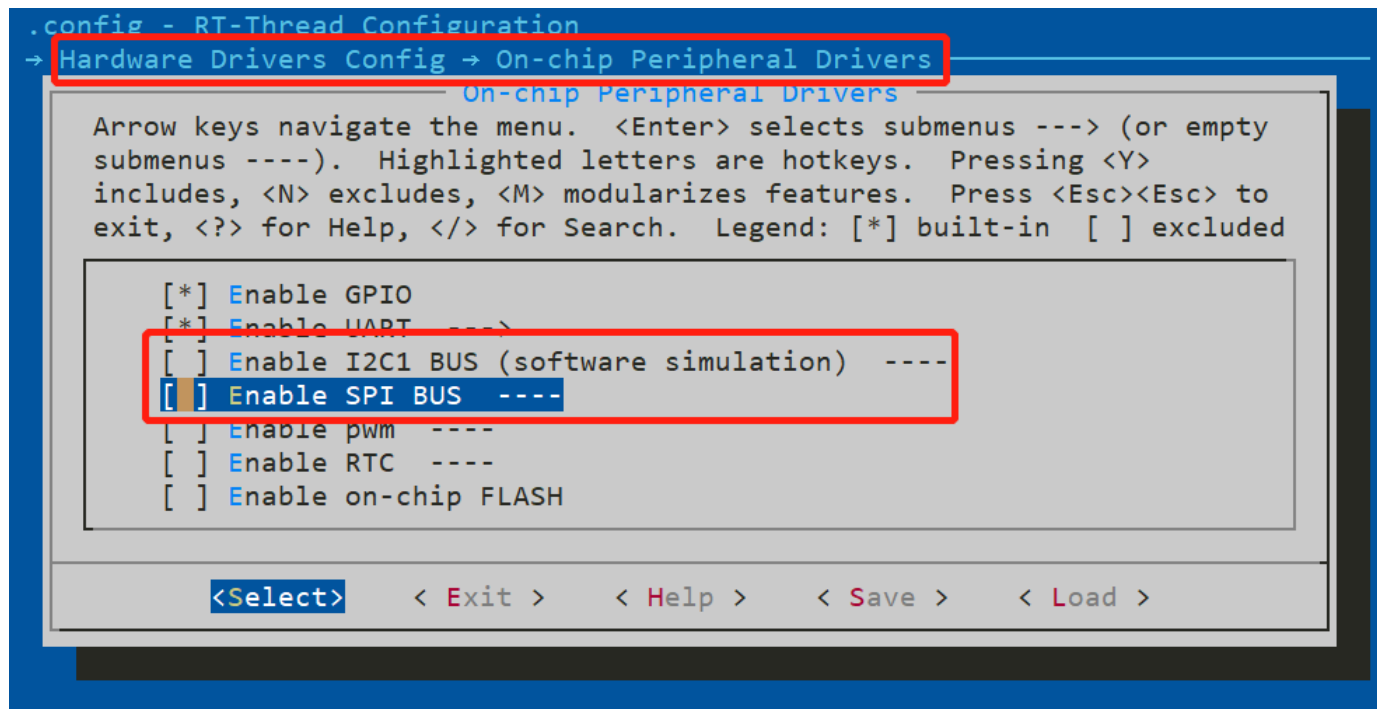
Env在具体的BSP下使用

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



如何关闭 i2c/spi 设备

2. 关闭i2c/spi设备框架

```
.config - RT-Thread Configuration
→ RT-Thread Components → Device Drivers
    Device Drivers
    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
    ↑(-)
    [ ] Using CAN device drivers
    [ ] Using hardware timer device drivers
    [ ] Enable CPU time for high resolution clock counter
    [ ] Using I2C device drivers
    [*] Using generic GPIO device drivers
    [ ] Using ADC device drivers
    [ ] Using PWM device drivers
    ↓(+)
```

```
.config - RT-Thread Configuration
→ RT-Thread Components → Device Drivers
    Device Drivers
    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
    ↑(-)
    [ ] Using Power Management device drivers
    [ ] Using RTC device drivers
    [ ] Using SD/MMC device drivers
    [ ] Using SPI Bus/Device device drivers
    [*] Using Watch Dog device drivers
    [ ] Using Audio device drivers
    [ ] Using Sensor device drivers
    ↓(+)
```

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

Do you wish to save your new configuration?
(Press <ESC><ESC> to continue kernel configuration.)

< Yes > < No >

configuration written to .config

```
*** End of the configuration.
*** You can execute 'scons' to start the build or try 'scons -h'
*** If you want to generate the IDE's project file, you can use command:
*** 'scons --target=mdk/mdk4/mdk5/iar/cb -s'.
*** If you want to install rt-thread component online, try 'pkgs'.

Administrator@P-V-9 D:\repository\rt-thread-rtt\bsp\stm32\stm32f411-st-nucleo
> scons --target=mdk5
cmd.exe [64]:13204
```

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.

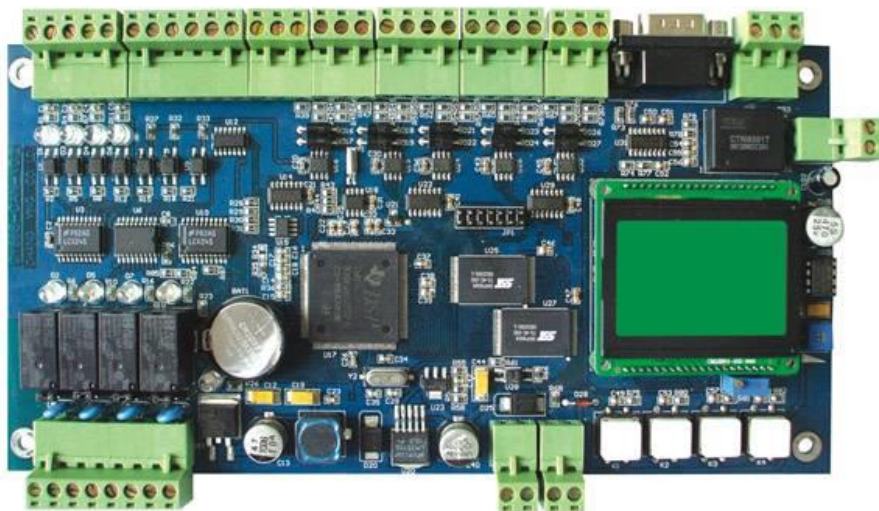
msh > list device
device          type          ref count
-----
uart2           Character Device    2
pin             Miscellaneous Device 0
msh >
```



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



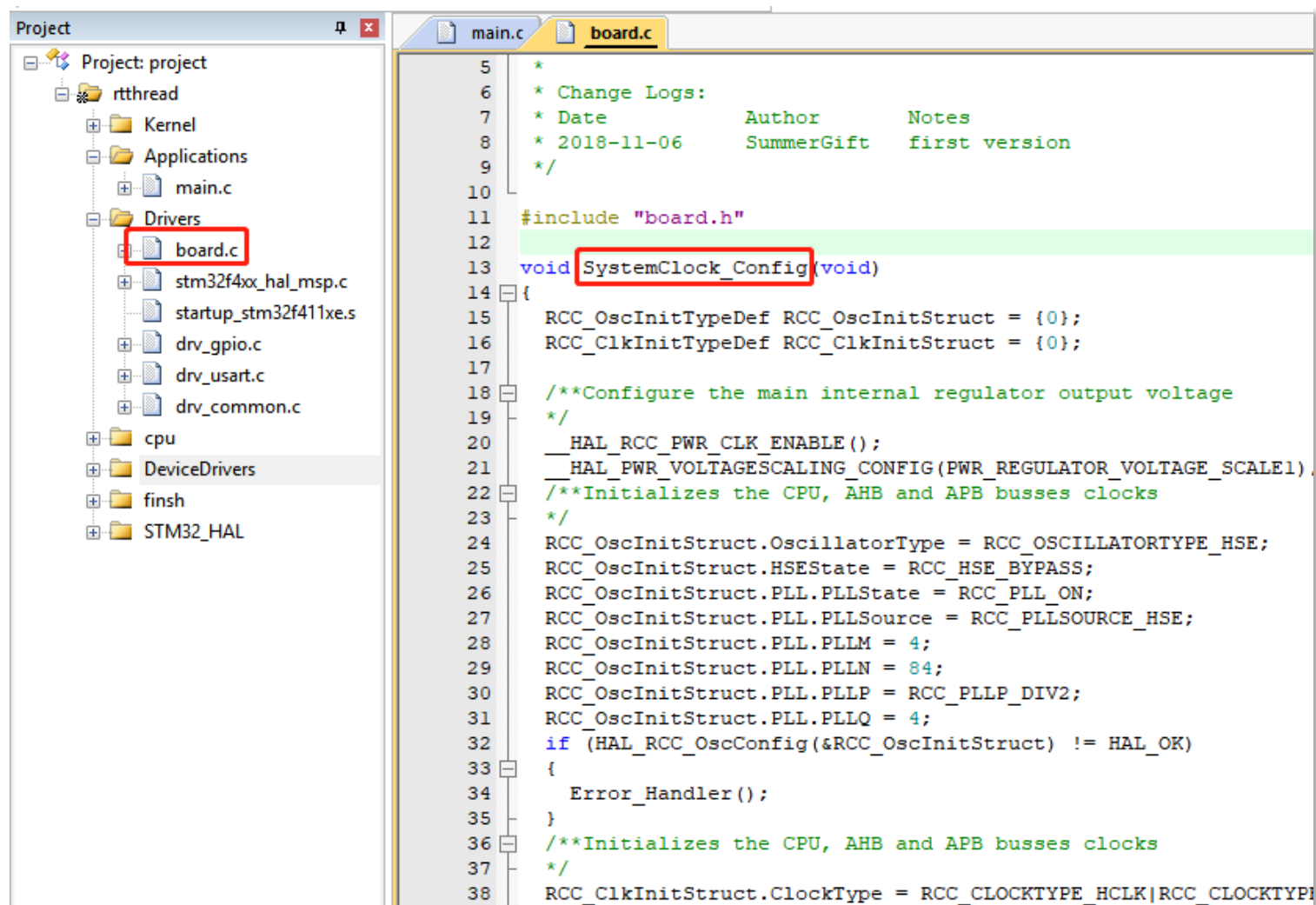
□ 配置系统时钟

□ 配置 FinSH 使用的串口。

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

配置系统时钟



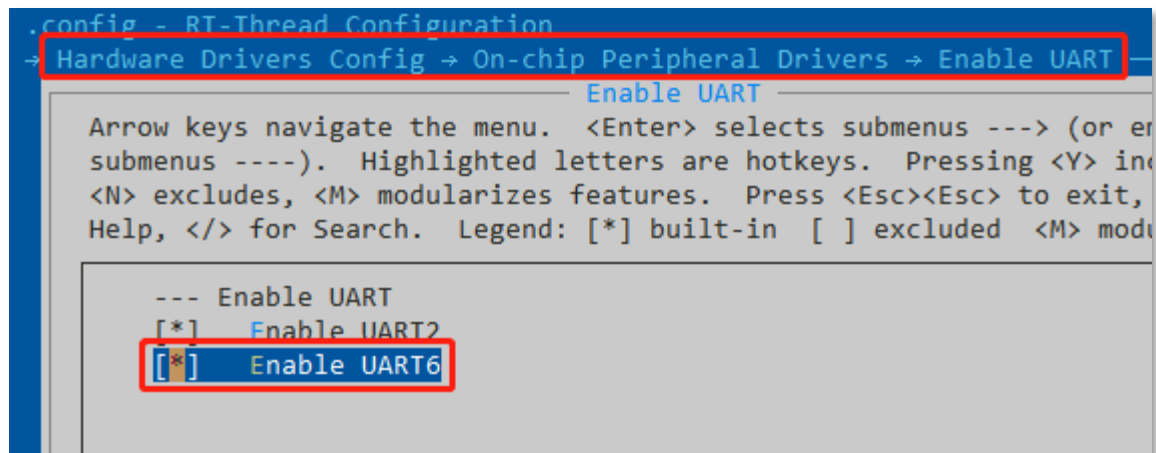
The screenshot displays an IDE interface. On the left, the 'Project' pane shows a tree structure: 'Project: project' contains 'rtthread', which includes 'Kernel', 'Applications', and 'Drivers'. The 'Drivers' folder is expanded, showing 'board.c' (highlighted with a red box), 'stm32f4xx_hal_msp.c', 'startup_stm32f411xe.s', 'drv_gpio.c', 'drv_usart.c', and 'drv_common.c'. Below this are 'cpu', 'DeviceDrivers', 'finsh', and 'STM32_HAL'. The main editor on the right shows the content of 'board.c'. It starts with a comment block (lines 5-9) and an include for 'board.h' (line 11). The function 'void SystemClock_Config(void)' (line 13, also highlighted with a red box) is defined. It initializes RCC structures (lines 15-16), configures the main internal regulator output voltage (lines 18-19), and initializes the CPU, AHB, and APB buses clocks (lines 21-23). It then configures the RCC_OscInitStruct (lines 24-31) and checks if HAL_RCC_OscConfig returns HAL_OK (line 32). If not, it calls Error_Handler() (lines 34-35). Finally, it initializes the CPU, AHB, and APB buses clocks (lines 36-37) and sets the RCC_ClkInitStruct.ClockType (line 38).

```
5  *
6  * Change Logs:
7  * Date       Author      Notes
8  * 2018-11-06  SummerGift  first version
9  */
10
11 #include "board.h"
12
13 void SystemClock_Config(void)
14 {
15     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
16     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
17
18     /**Configure the main internal regulator output voltage
19     */
20     __HAL_RCC_PWR_CLK_ENABLE();
21     __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
22     /**Initializes the CPU, AHB and APB busses clocks
23     */
24     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
25     RCC_OscInitStruct.HSEState = RCC_HSE_BYPASS;
26     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
27     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
28     RCC_OscInitStruct.PLL.PLLM = 4;
29     RCC_OscInitStruct.PLL.PLLN = 84;
30     RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
31     RCC_OscInitStruct.PLL.PLLQ = 4;
32     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
33     {
34         Error_Handler();
35     }
36     /**Initializes the CPU, AHB and APB busses clocks
37     */
38     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
```

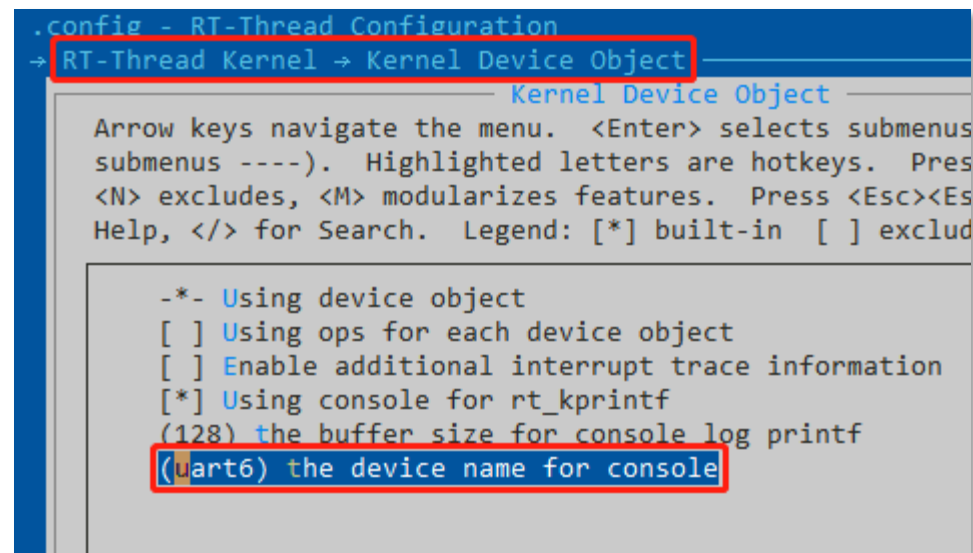
适配项目开发板

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

配置 FinSH 使用的串口



1. 开启对应串口



2. 配置console对应串口

env配置

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

Do you wish to save your new configuration?
(Press <ESC><ESC> to continue kernel configuration.)

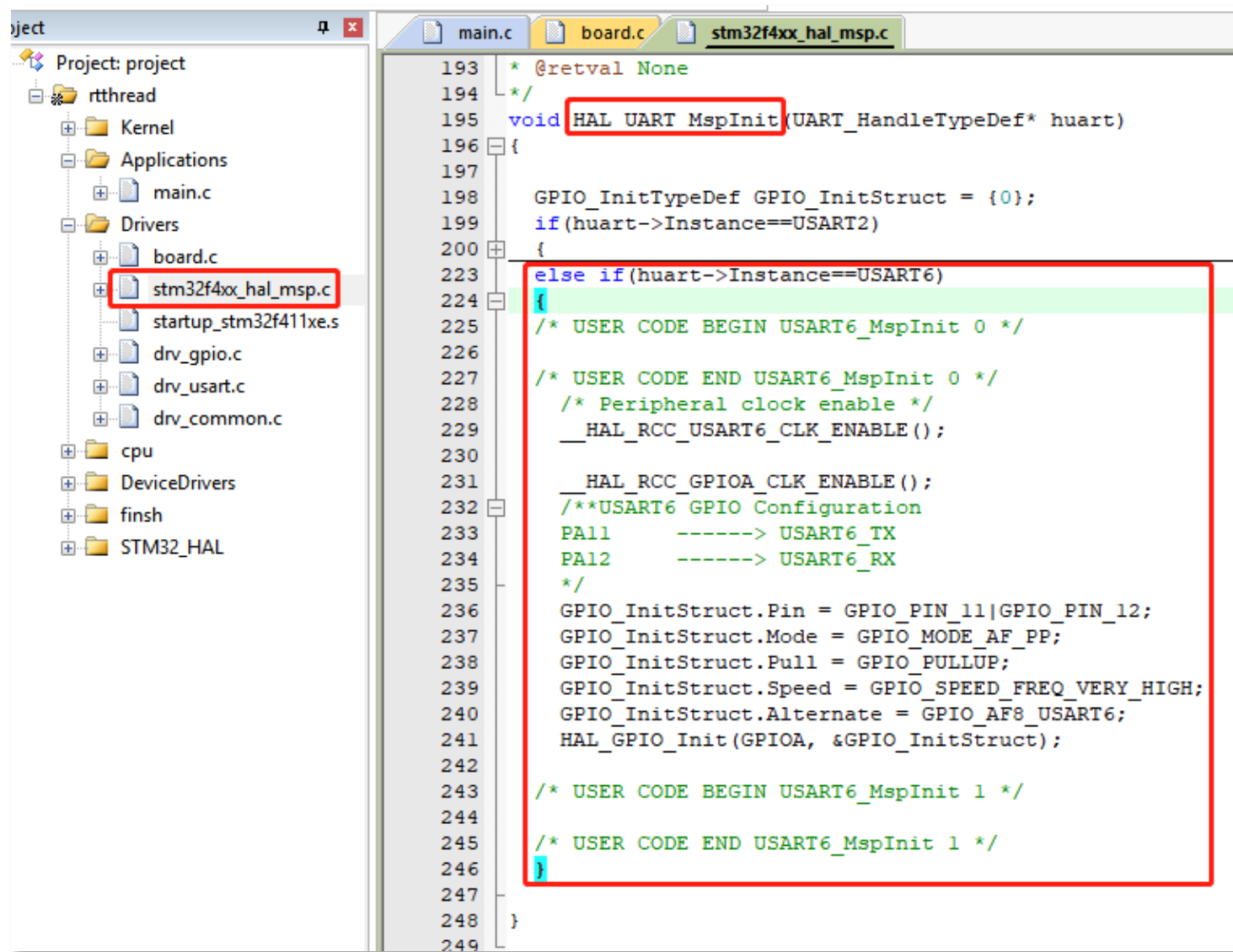
< Yes > < No >

```
configuration written to .config

*** End of the configuration.
*** You can execute 'scons' to start the build or try 'scons -h'
*** If you want to generate the IDE's project file, you can use command:
*** 'scons --target=mdk/mdk4/mdk5/iar/cb -s'.
*** If you want to install rt-thread component online, try 'pkgs'.
Operation completed successfully.
=====>The packages have been updated completely.

11714@DESKTOP-FLFR36Q F:\rtt-car\bsp\es32f0654
> scons --target=mdk5
cmd.exe*[64]:13392
```

确认串口引脚



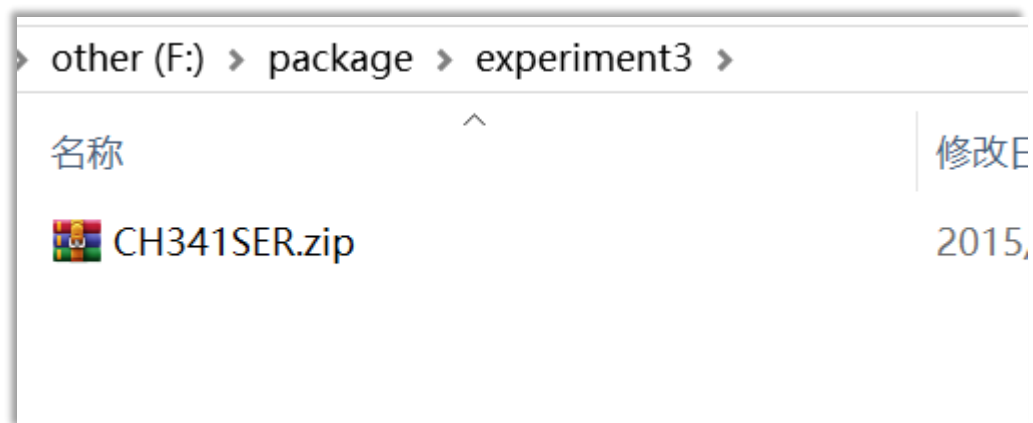
```
193 * @retval None
194 */
195 void HAL_UART_MspInit(UART_HandleTypeDef* huart)
196 {
197     GPIO_InitTypeDef GPIO_InitStruct = {0};
198     if(huart->Instance==USART2)
199     {
200     }
201     else if(huart->Instance==USART6)
202     {
203         /* USER CODE BEGIN USART6_MspInit 0 */
204
205         /* USER CODE END USART6_MspInit 0 */
206         /* Peripheral clock enable */
207         __HAL_RCC_USART6_CLK_ENABLE();
208
209         __HAL_RCC_GPIOA_CLK_ENABLE();
210
211         /**USART6 GPIO Configuration
212             PA11 -----> USART6_TX
213             PA12 -----> USART6_RX
214         */
215         GPIO_InitStruct.Pin = GPIO_PIN_11|GPIO_PIN_12;
216         GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
217         GPIO_InitStruct.Pull = GPIO_PULLUP;
218         GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
219         GPIO_InitStruct.Alternate = GPIO_AF8_USART6;
220         HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
221
222         /* USER CODE BEGIN USART6_MspInit 1 */
223
224         /* USER CODE END USART6_MspInit 1 */
225     }
226 }
```

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

驱动安装

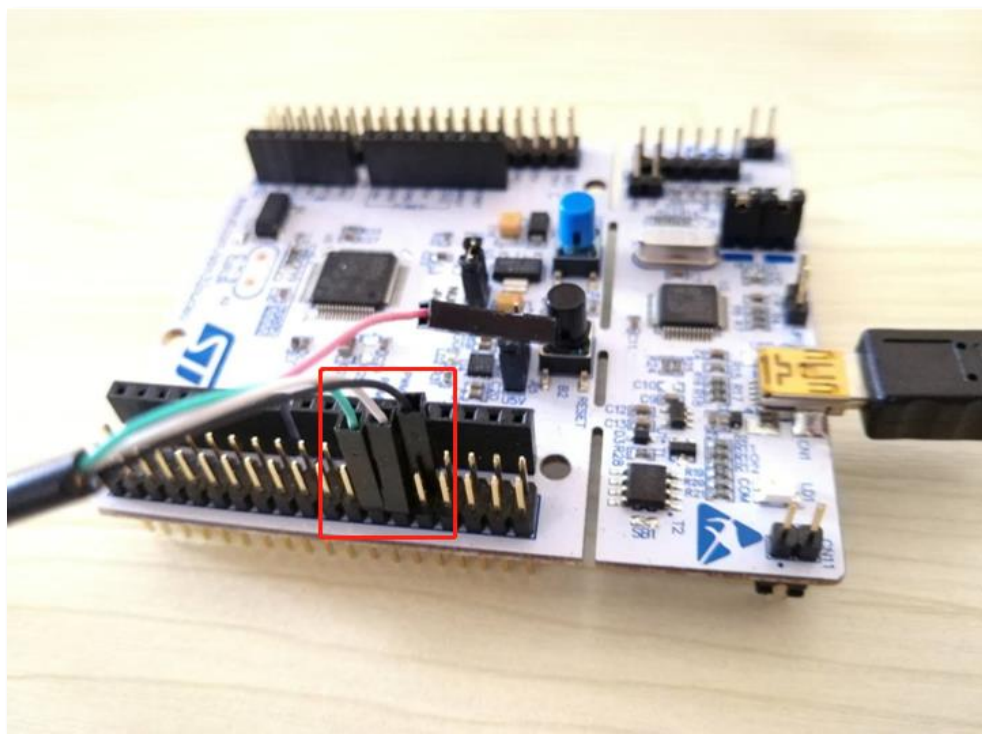
- 安装 CH340 USB转串口模块的驱动



硬件连接

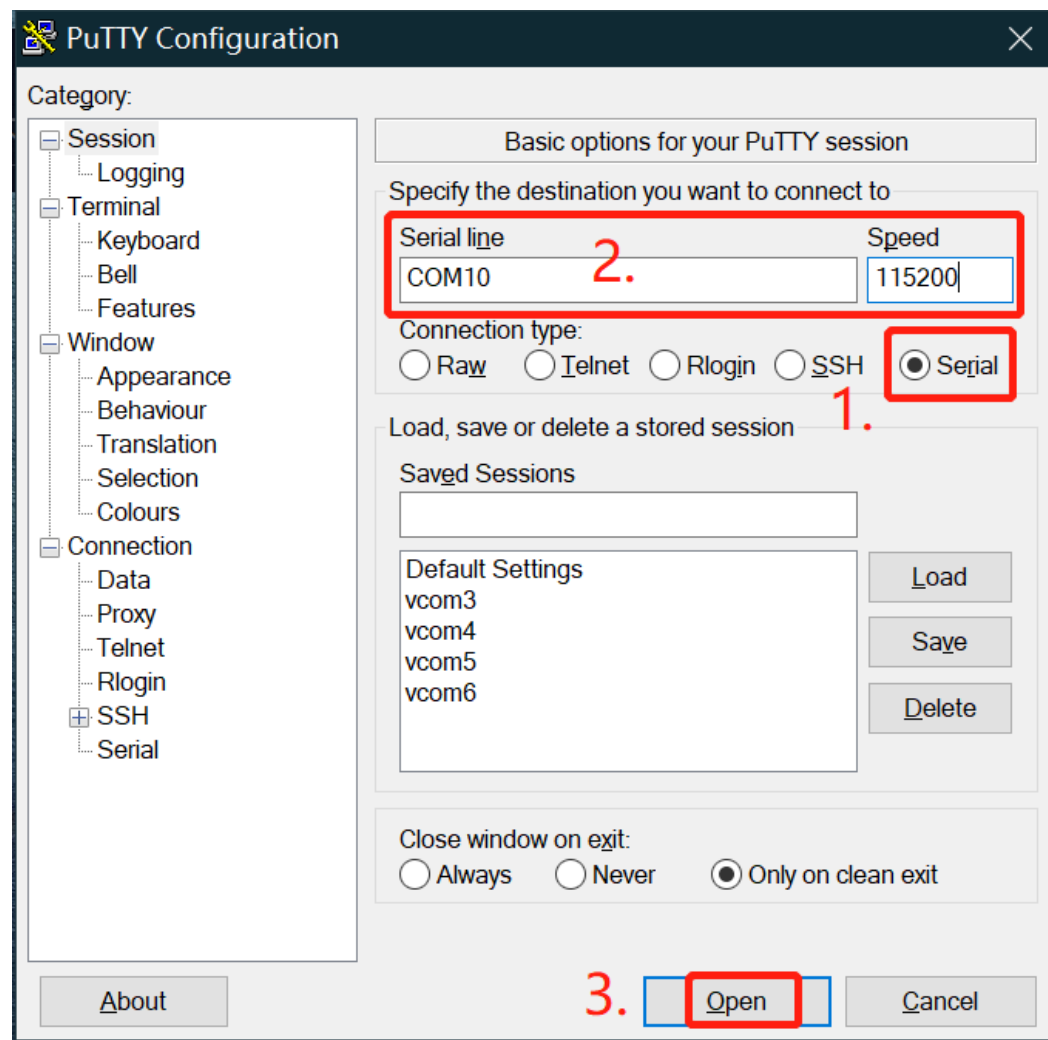
- 使用USB转串口工具连接对应引脚到PC

绿色接PA12
白色接PA11



注意：引脚定义在
开发板的包装上

连接终端工具



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

编译运行

- 编译下载，观察现象。

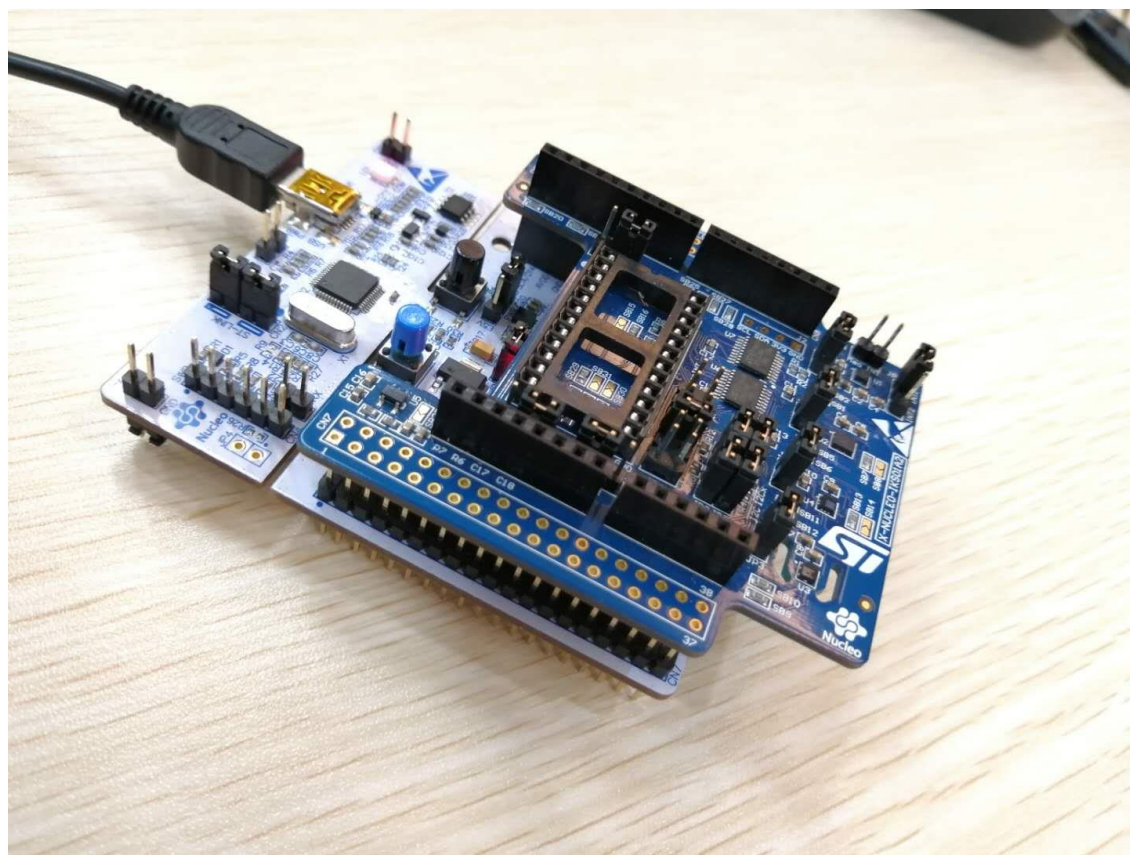
```
  \ | /  
- RT -      Thread Operating System  
  / | \  
2006 - 2019 Copyright by rt-thread team  
msh >
```



实验四、读取传感器数据

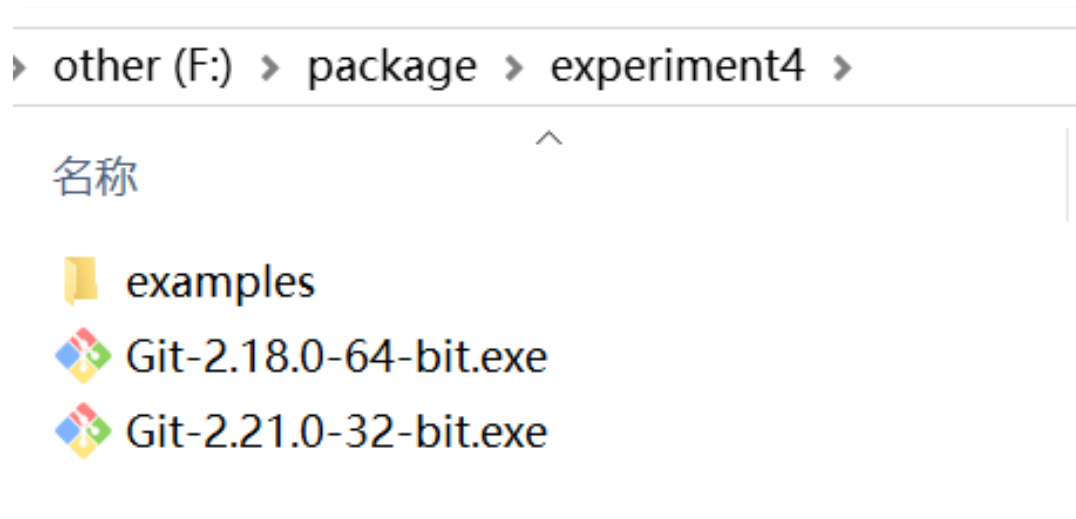
硬件连接

- 传感器插入开发板上



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



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 --->

```
.config - RT-Thread Configuration
→ RT-Thread online packages → peripheral libraries and drivers → sensors drivers

sensors drivers
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

-- sensors drivers
[*] LSM6DSL sensor driver package, support: accelerometer, gyroscope, step. -
[ ] LPS22HB sensor driver package, support: barometric,temperature. ----
[ ] HTS221 sensor driver package, support: temperature, humidity. ----
[*] LSM303AGR sensor driver package, support: accelerometer, magnetometer. --
[ ] BME280 sensor driver package, support: barometric, humidity. ----
[ ] BMA400 sensor driver package, support: accelerometer, step. ----
[ ] BMI160/BMX160: Digital 6-axis and Digital 9-axis sensor ----
[ ] SPL0601: Digital pressure sensor ----
↓(+)
```

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

开启传感器需要的IIC设备

- 开启片上外设的I2C1

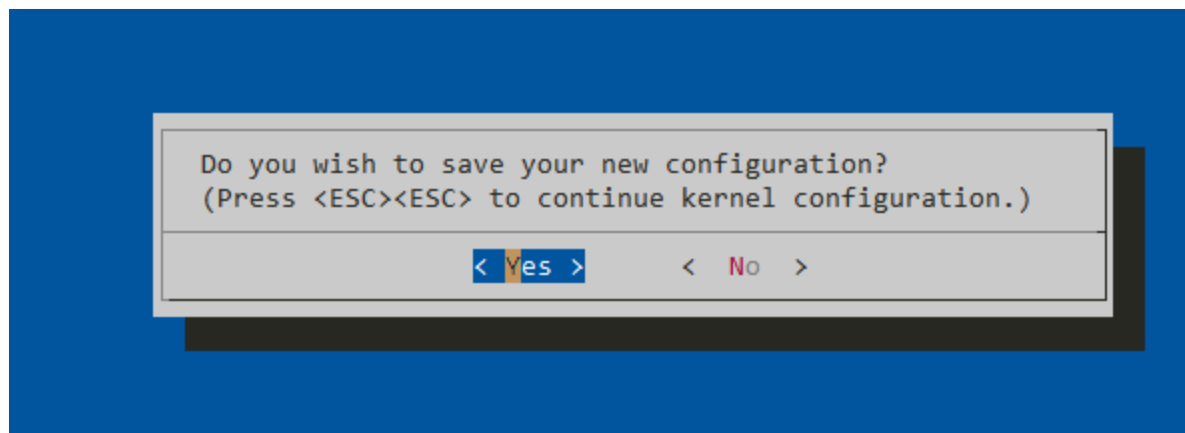
```
.config - RT-Thread Configuration
[... ] Config → On-chip Peripheral Drivers → Enable I2C1 BUS (software simulation)
      Enable I2C1 BUS (software simulation)
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes,
<N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for
Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < >

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

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

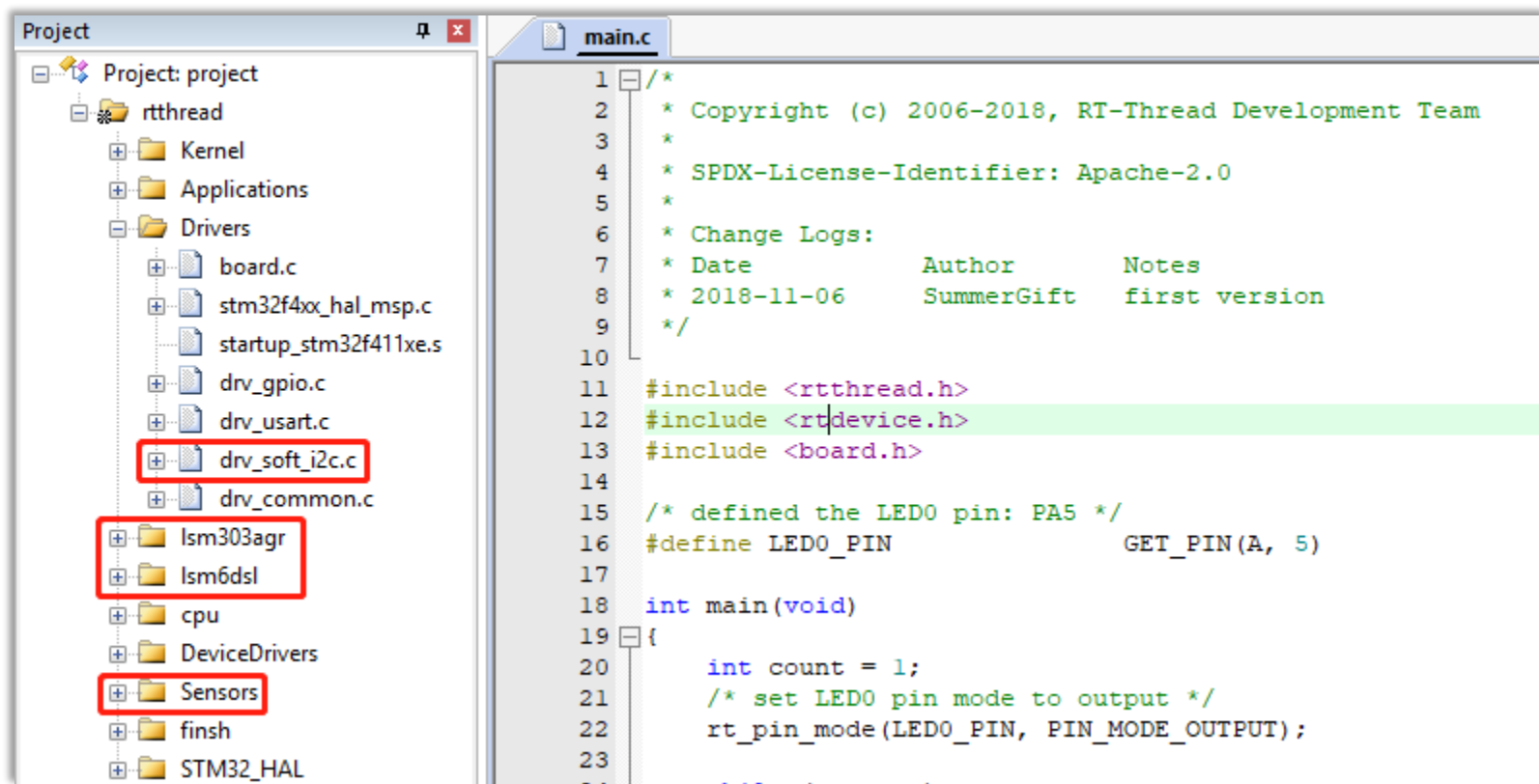
退出并保存配置

狂按 ESC 键退出，最后选中 YES 保存配置。



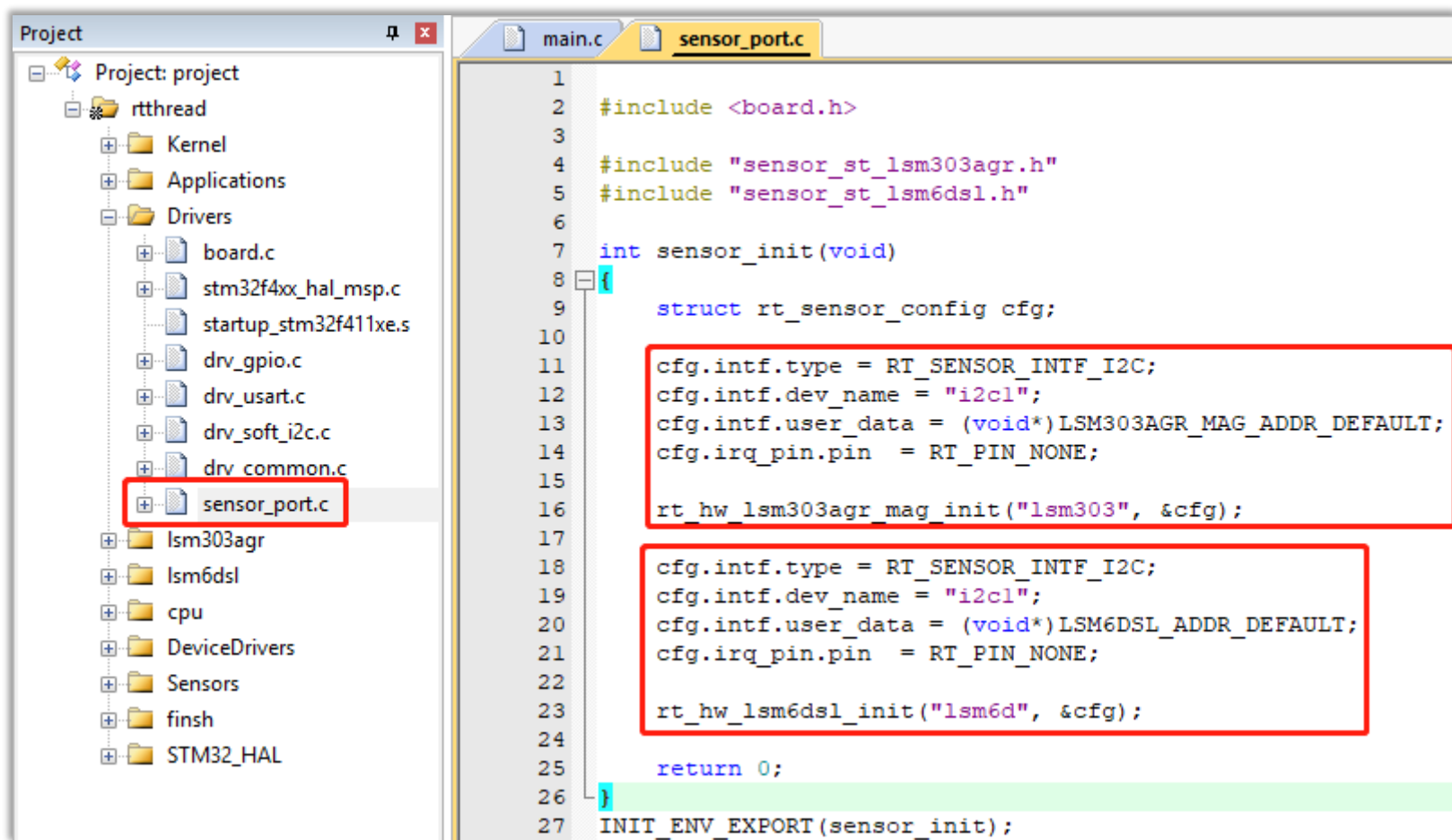
生成工程

- 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



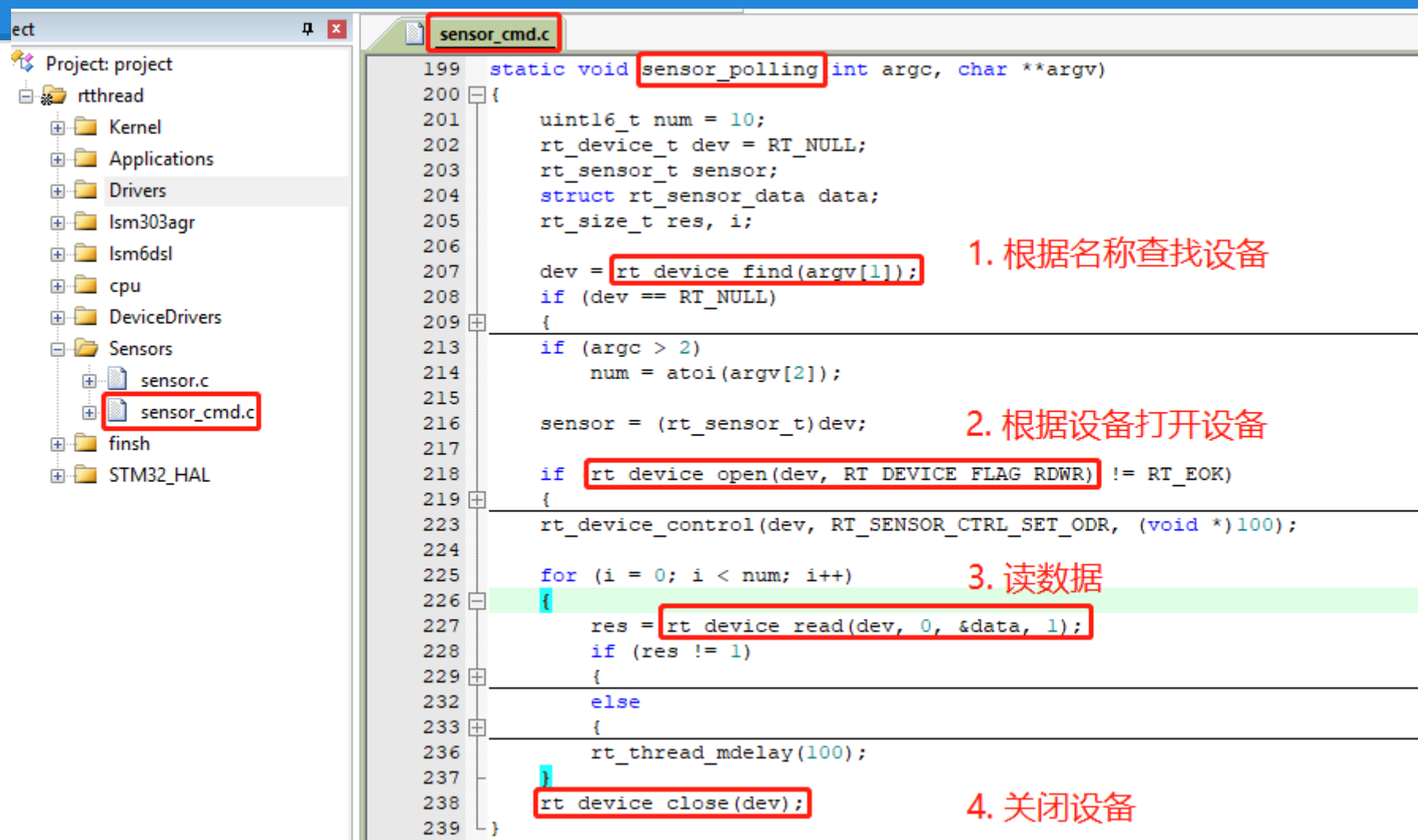
编译下载，测试传感器

```
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

API讲解



The screenshot shows an IDE with a project tree on the left and a code editor on the right. The project tree shows a folder named 'Sensors' containing 'sensor.c' and 'sensor_cmd.c'. The code editor shows the 'sensor_cmd.c' file with the following code:

```
199 static void sensor_polling(int argc, char **argv)
200 {
201     uint16_t num = 10;
202     rt_device_t dev = RT_NULL;
203     rt_sensor_t sensor;
204     struct rt_sensor_data data;
205     rt_size_t res, i;
206
207     dev = rt_device_find(argv[1]);
208     if (dev == RT_NULL)
209     {
210
211     }
212
213     if (argc > 2)
214         num = atoi(argv[2]);
215
216     sensor = (rt_sensor_t)dev;
217
218     if (rt_device_open(dev, RT_DEVICE_FLAG_RDWR) != RT_EOK)
219     {
220
221     }
222
223     rt_device_control(dev, RT_SENSOR_CTRL_SET_ODR, (void *)100);
224
225     for (i = 0; i < num; i++)
226     {
227         res = rt_device_read(dev, 0, &data, 1);
228         if (res != 1)
229         {
230
231         }
232         else
233         {
234
235         }
236         rt_thread_mdelay(100);
237     }
238     rt_device_close(dev);
239 }
```

Annotations in the image highlight specific API calls and their corresponding steps:

- 1. 根据名称查找设备 (Find device by name): `rt_device_find(argv[1]);`
- 2. 根据设备打开设备 (Open device by device): `rt_device_open(dev, RT_DEVICE_FLAG_RDWR) != RT_EOK`
- 3. 读数据 (Read data): `rt_device_read(dev, 0, &data, 1);`
- 4. 关闭设备 (Close device): `rt_device_close(dev);`

如何查找API

RT-Thread, RTOS, 物联网操作系统 | flyboy - RT-Thread开发者社区 | UART设备 - RT-Thread 文档中心

https://www.rt-thread.org/document/site/programming-manual/device/uart/uart/

应用 RT-Thread, RTOS, ... GitHub - RT-Thread ... Activity · Dashboa... Index of / 【整理】Android... PrimeSe

RT-Thread 文档中心

Search docs

文档首页
RT-Thread简介
快速上手
内核
Env工具
设备和驱动

I/O设备模型

UART设备

UART 简介

访问串口设备

串口设备使用示例

PIN设备

ADC设备

HWTIMER设备

I2C总线设备

PWM设备

- 停止位：表示一帧数据的结束。电平逻辑为“1”。
- 波特率：串口通信时的速率，它用单位时间内传输的二进制代码的有115200等，数值越大数据传输的越快，波特率为 115200 表示每秒传输115200个字节。

访问串口设备

应用程序通过 RT-Thread提供的 I/O 设备管理接口来访问串口硬件，相关函数如下：

函数	描述
rt_device_find()	根据串口设备名称查找设备获取设备句柄
rt_device_open()	打开设备
rt_device_read()	读取数据
rt_device_write()	写入数据
rt_device_control()	控制设备
rt_device_set_rx_indicate()	设置接收回调函数
rt_device_set_tx_complete()	设置发送完成回调函数
rt_device_close()	关闭设备

1. 文档中心

RT-Thread API参考手册

RT-Thread 3.1.1 嵌入式实时操作系统

首页 模块 结构体 示例

RT-Thread 简介

作者
RT-Thread Development Team

版本
3.1.1

目录(N) 索引(S) 搜索(S) 收藏

- RT-Thread 简介
- 模块
 - 内核
 - 文件系统
 - 网络
 - 设备管理
 - PinSH控制台
 - 日志组件
 - 动态模块
 - AT 组件
 - 示例程序
- 结构体
- 示例

2. API手册

F:\rt-thread-no-save\bsp\stm32\stm32f411-st-nucleo\project.uvprojx - μVision

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Project: project

- rtthread
- Kernel
- Applications
- Drivers
- cpu
- DeviceDrivers
 - pin.c**
 - serial.c
 - completion.c
 - dataqueue.c
 - pipe.c
 - ringblk_buf.c
 - ringbuffer.c
 - waitqueue.c
 - workqueue.c
- finsh
- STM32_HAL

pin.h

```
69 void (*pin_write)(struct rt_device *device, rt_base_t pin, rt_base_t value);
70 int (*pin_read)(struct rt_device *device, rt_base_t pin);
71
72 /* TODO: add GPIO interrupt */
73 rt_err_t (*pin_attach_irq)(struct rt_device *device, rt_int32_t pin,
74 rt_uint32_t mode, void (*hdr)(void *args), void *args);
75 rt_err_t (*pin_detach_irq)(struct rt_device *device, rt_int32_t pin);
76 rt_err_t (*pin_irq_enable)(struct rt_device *device, rt_base_t pin,
77 rt_uint32_t mode);
78
79 int rt_device_pin_register(const char *name, const struct rt_pin_ops *ops);
80
81 void rt_pin_mode(rt_base_t pin, rt_base_t mode);
82 void rt_pin_write(rt_base_t pin, rt_base_t value);
83 int rt_pin_read(rt_base_t pin);
84 rt_err_t rt_pin_attach_irq(rt_int32_t pin, rt_uint32_t mode,
85 void (*hdr)(void *args), void *args);
86 rt_err_t rt_pin_detach_irq(rt_int32_t pin);
87 rt_err_t rt_pin_irq_enable(rt_base_t pin, rt_uint32_t enabled);
88
89 #ifdef __cplusplus
90 }
91 #endif
92
93 #endif
```

3. 源代码



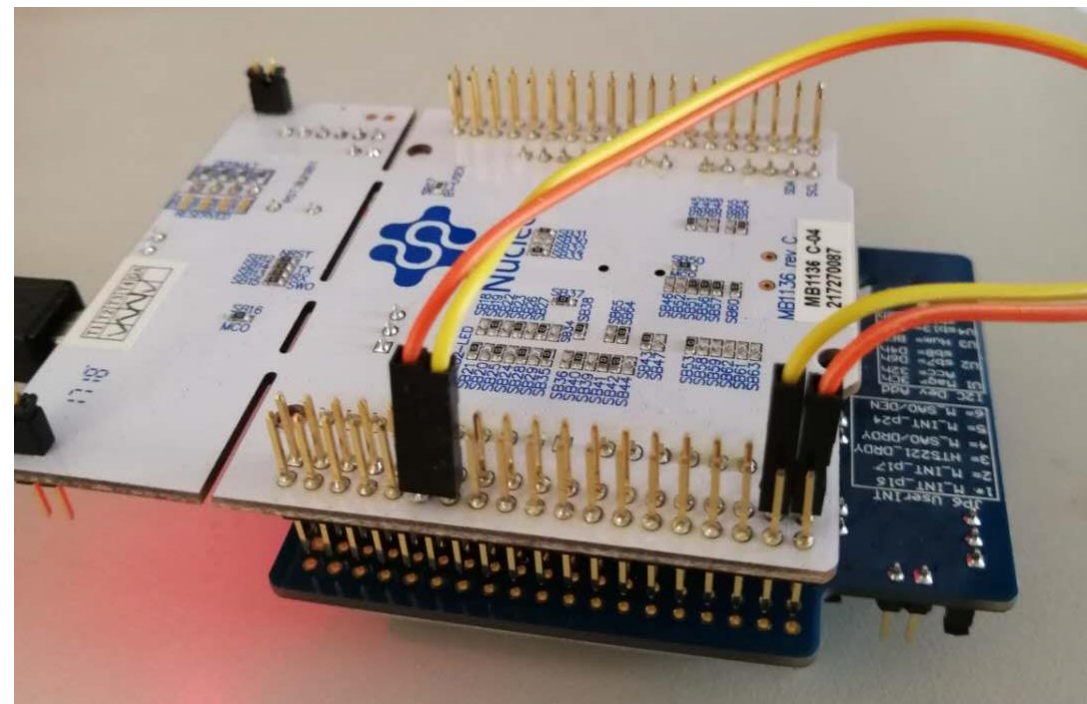
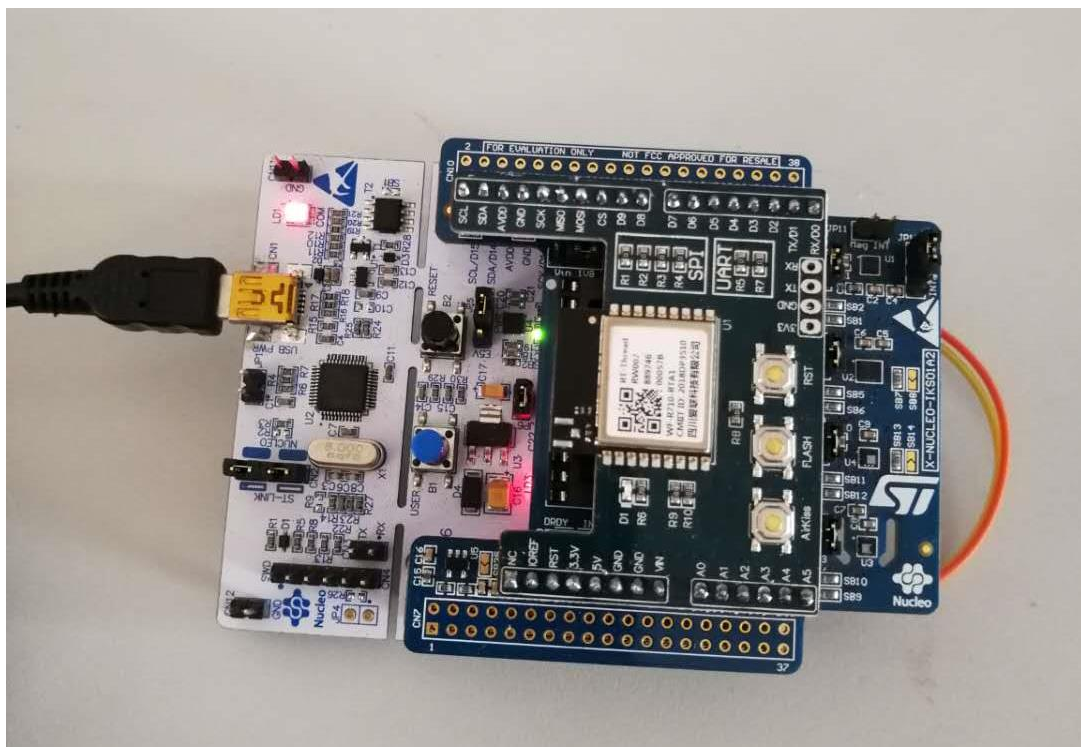
实验五、将传感器数据推送到本地服务器

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

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



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

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

```
.config - RT-Thread Configuration
[...] - internet of things -> AT DEVICE: RT-Thread AT component porting or samples for different device
    AT DEVICE: RT-Thread AT component porting or samples for different device
    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

    - AT DEVICE: RT-Thread AT component porting or samples for different device
      [ ] Enable at device init by thread (NEW)
      AT socket device modules (Realthread RW007) --->
      (uart6) AT client device name
      (512) The maximum length of client data accepted (NEW)
      (rtthread) rw007 wifi ssid (NEW)
      (12345678) rw007 wifi password (NEW)
      version (latest) --->

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

通讯用的设备

选择RW007

wifi热点的名字和密码

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

```
.config - RT-Thread Configuration
→ Hardware Drivers Config → On-chip Peripheral Drivers → Enable UART

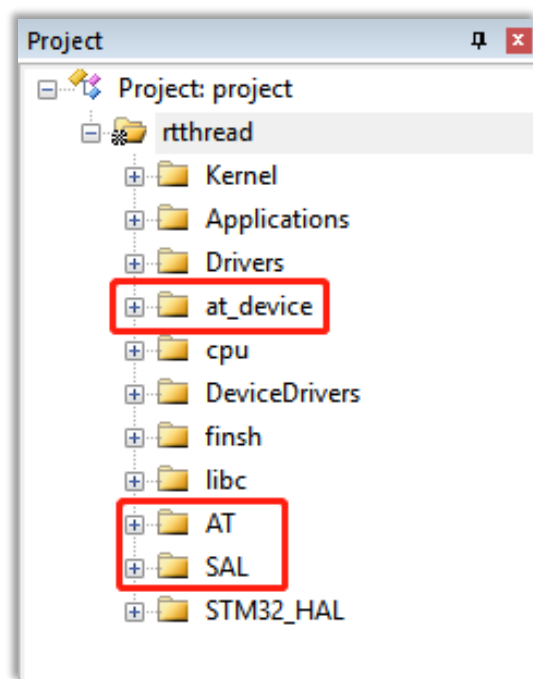
Enable UART
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenu ----).
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features.
Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded
<M> module < > module capable

--- Enable UART
[*] Enable UART2
[*] Enable UART6

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

生成工程

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

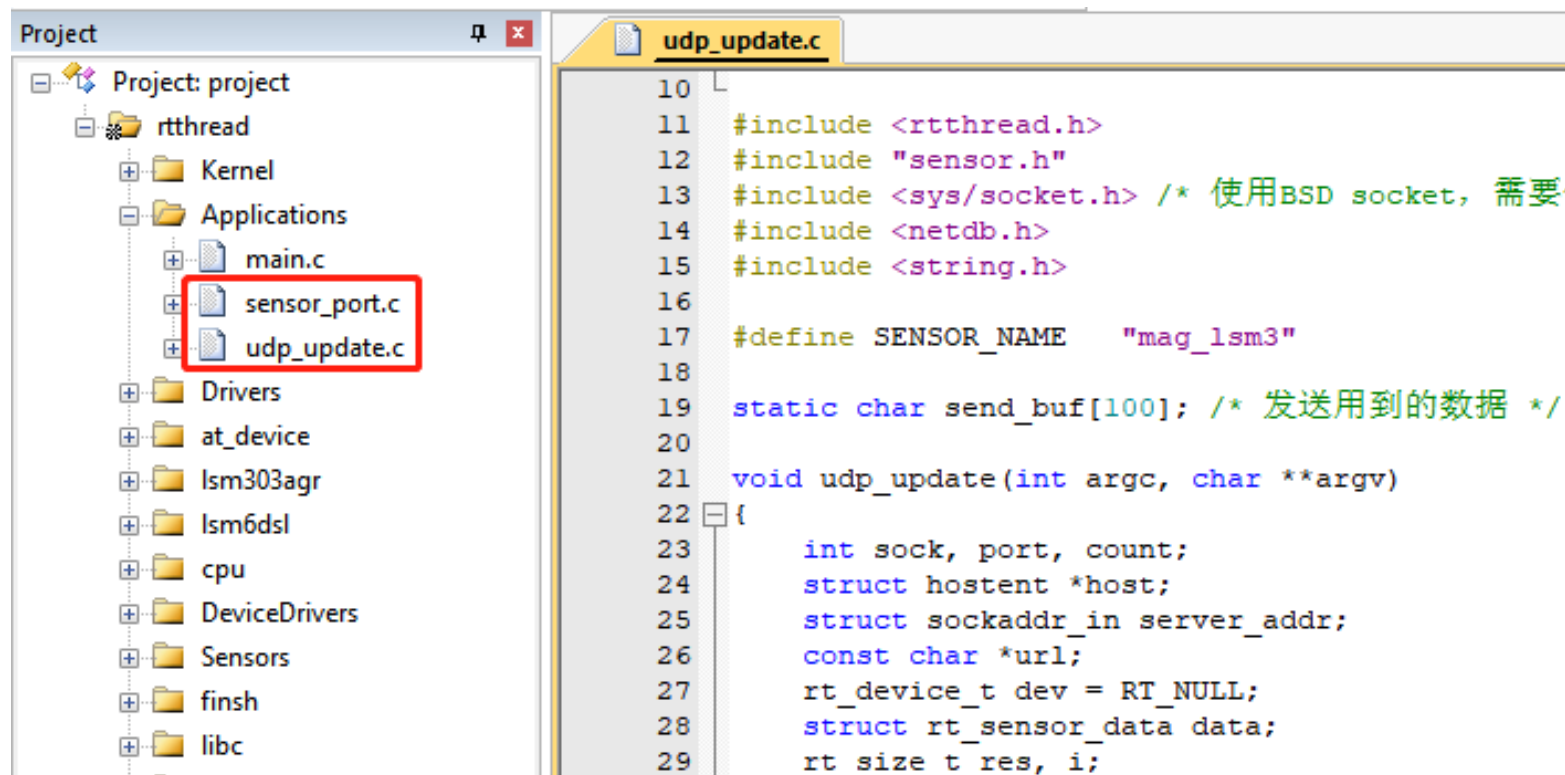


将传感器数据推送到本地服务器

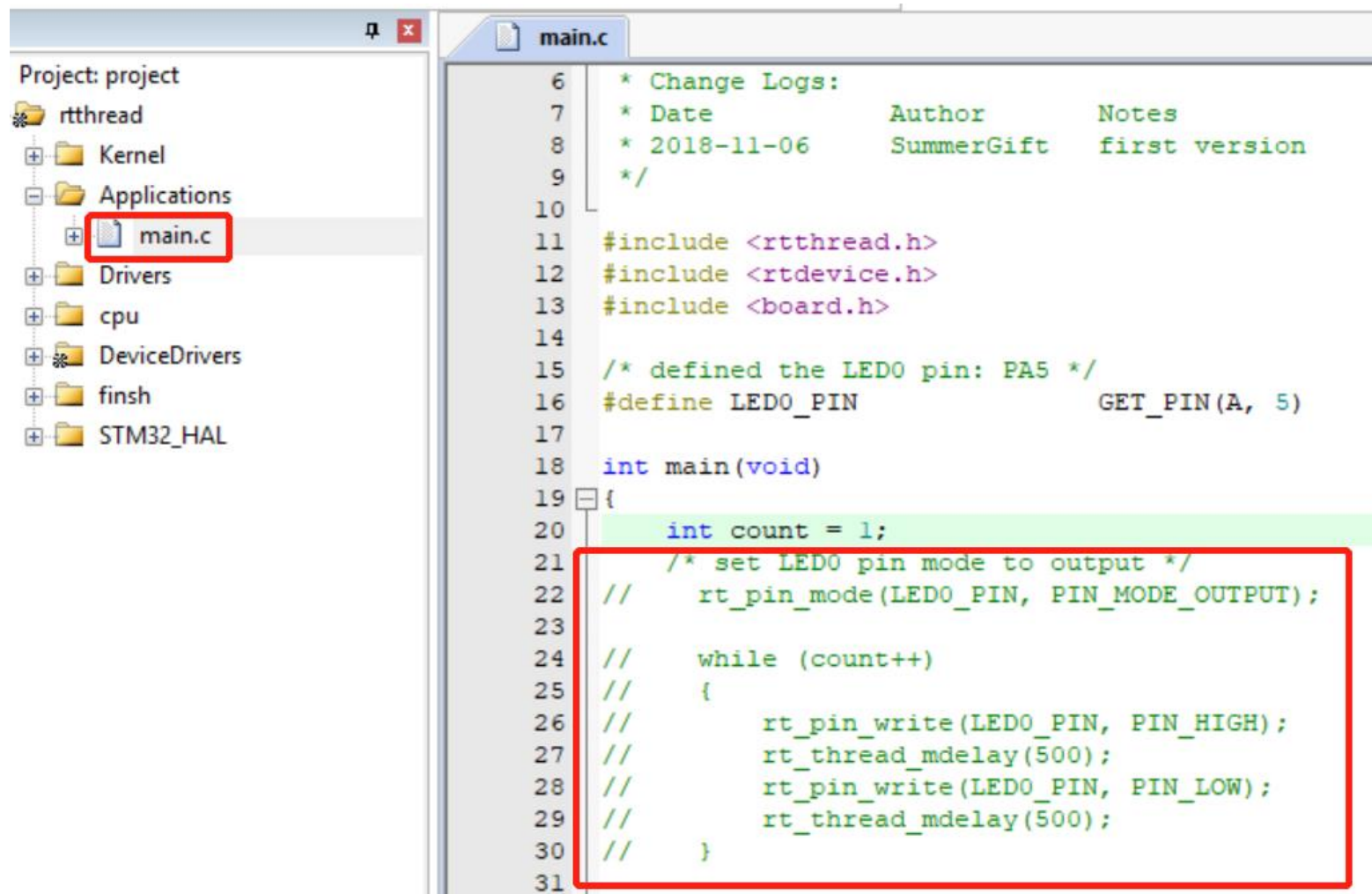
- 硬件连接
- 配置工程
- 添加示例代码
- 编译运行

添加示例代码

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



屏蔽mian.c中闪光灯程序



```
6  * Change Logs:
7  * Date           Author       Notes
8  * 2018-11-06      SummerGift   first version
9  */
10
11 #include <rtthread.h>
12 #include <rtdevice.h>
13 #include <board.h>
14
15 /* defined the LED0 pin: PA5 */
16 #define LED0_PIN          GET_PIN(A, 5)
17
18 int main(void)
19 {
20     int count = 1;
21     /* set LED0 pin mode to output */
22     // rt_pin_mode(LED0_PIN, PIN_MODE_OUTPUT);
23
24     // while (count++)
25     // {
26     //     rt_pin_write(LED0_PIN, PIN_HIGH);
27     //     rt_thread_mdelay(500);
28     //     rt_pin_write(LED0_PIN, PIN_LOW);
29     //     rt_thread_mdelay(500);
30     // }
31 }
```

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

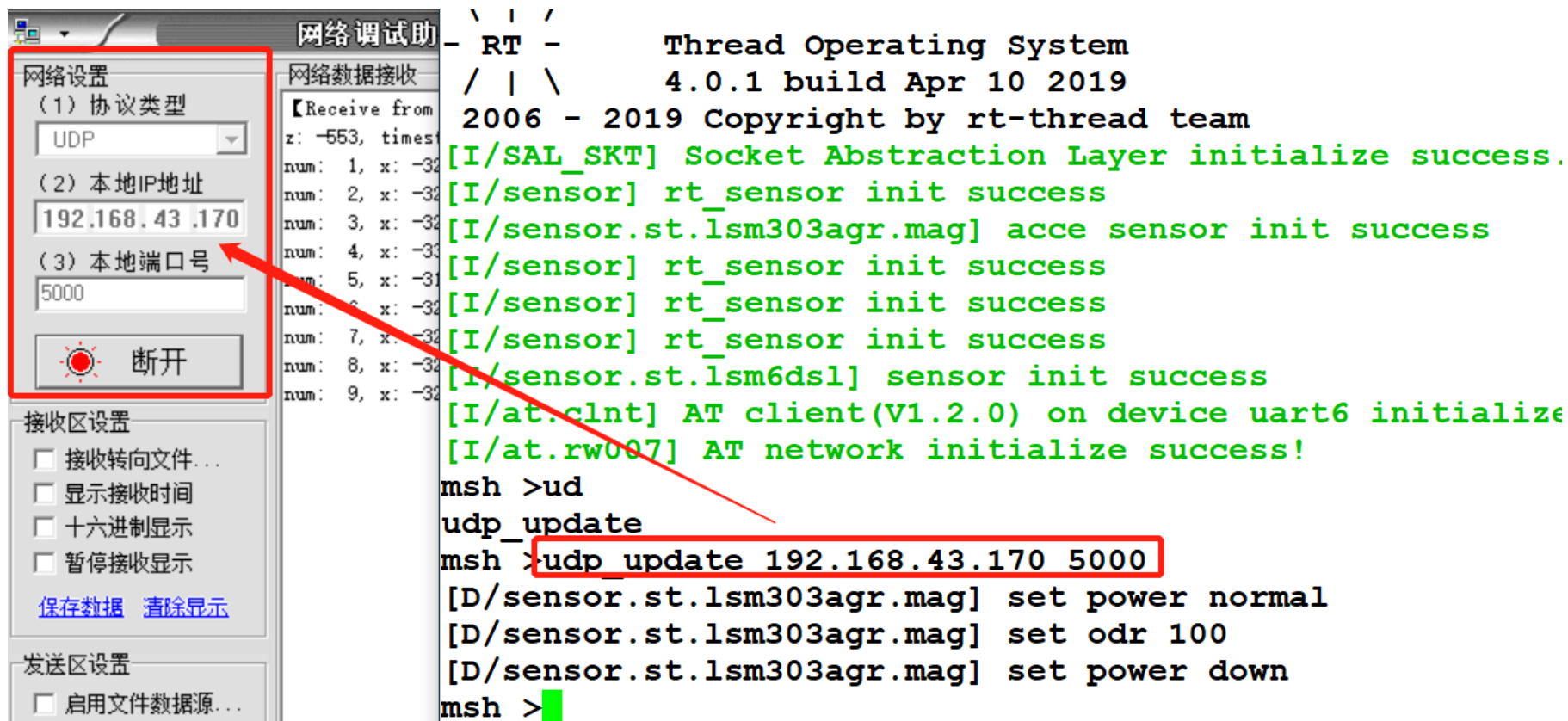
编译运行

- 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 >
```

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。



The image shows the Network Debug Assistant (网络调试助手) interface on the left and a terminal window on the right. The interface has a red box around the 'Network Settings' (网络设置) section, which includes a dropdown for 'Protocol Type' (协议类型) set to 'UDP', a text field for 'Local IP Address' (本地IP地址) set to '192.168.43.170', and a text field for 'Local Port Number' (本地端口号) set to '5000'. A red arrow points from the 'Local Port Number' field to the terminal. The terminal window shows the output of the program, including the RT-Thread logo, version information (4.0.1 build Apr 10 2019), copyright notice (2006 - 2019 Copyright by rt-thread team), and various initialization messages for the Socket Abstraction Layer (SAL_SKT), sensors (rt_sensor, lsm303agr, lsm6dsl), and the AT client (AT client(V1.2.0) on device uart6). The terminal also shows the command 'msh >udp udp_update 192.168.43.170 5000' being executed, which is highlighted with a red box. The output of this command shows the sensor settings being updated: '[D/sensor.st.lsm303agr.mag] set power normal', '[D/sensor.st.lsm303agr.mag] set odr 100', and '[D/sensor.st.lsm303agr.mag] set power down'.

```
网络调试助手
网络数据接收
【Receive from
z: -553, times
num: 1, x: -32
num: 2, x: -32
num: 3, x: -32
num: 4, x: -32
num: 5, x: -32
num: 6, x: -32
num: 7, x: -32
num: 8, x: -32
num: 9, x: -32

- 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
[I/at.rw007] AT network initialize success!

msh >ud
udp_update
msh >udp_update 192.168.43.170 5000
[D/sensor.st.lsm303agr.mag] set power normal
[D/sensor.st.lsm303agr.mag] set odr 100
[D/sensor.st.lsm303agr.mag] set power down
msh >
```



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/>



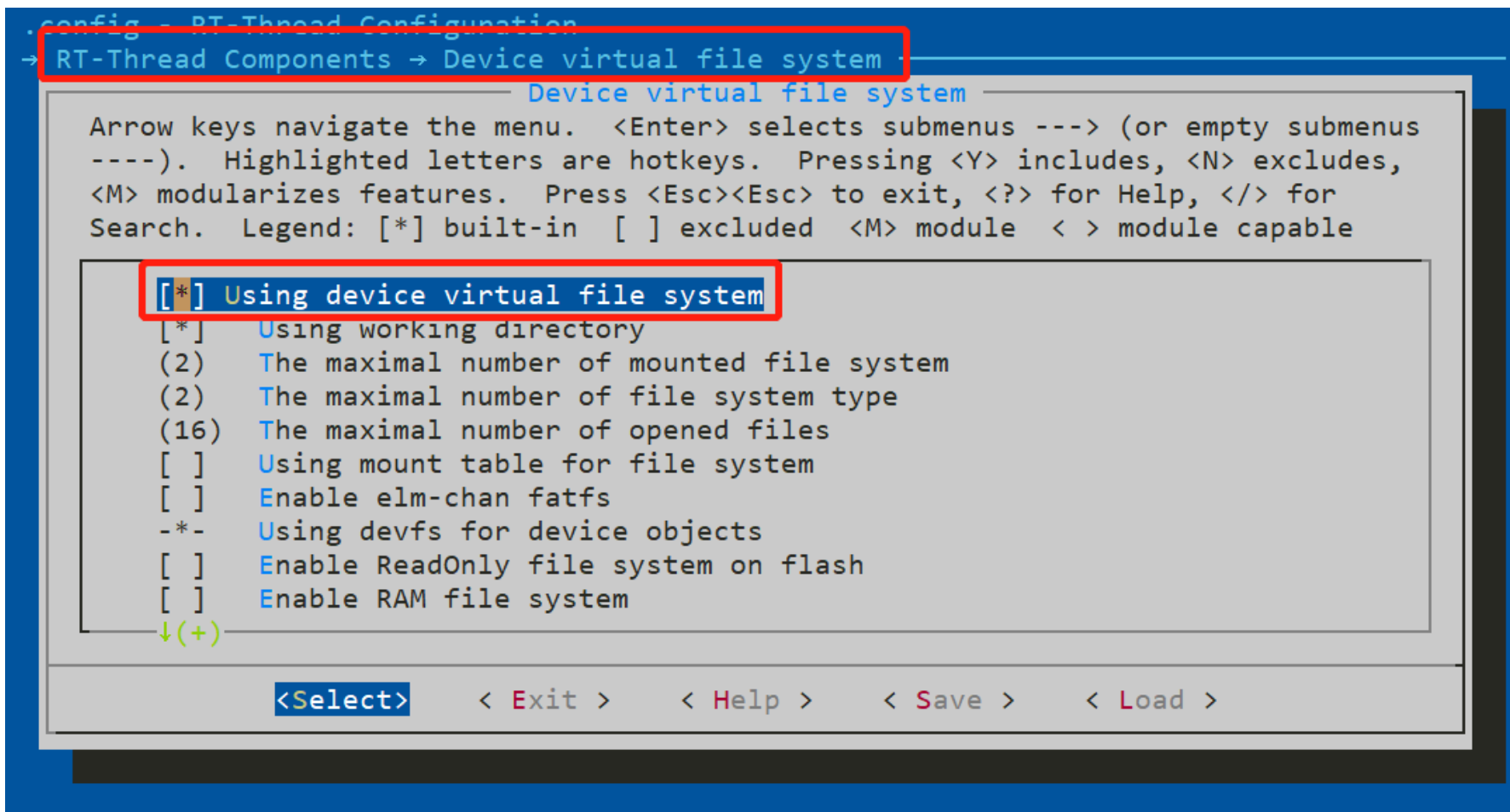
演示：移植游戏2048

演示：移植游戏2048

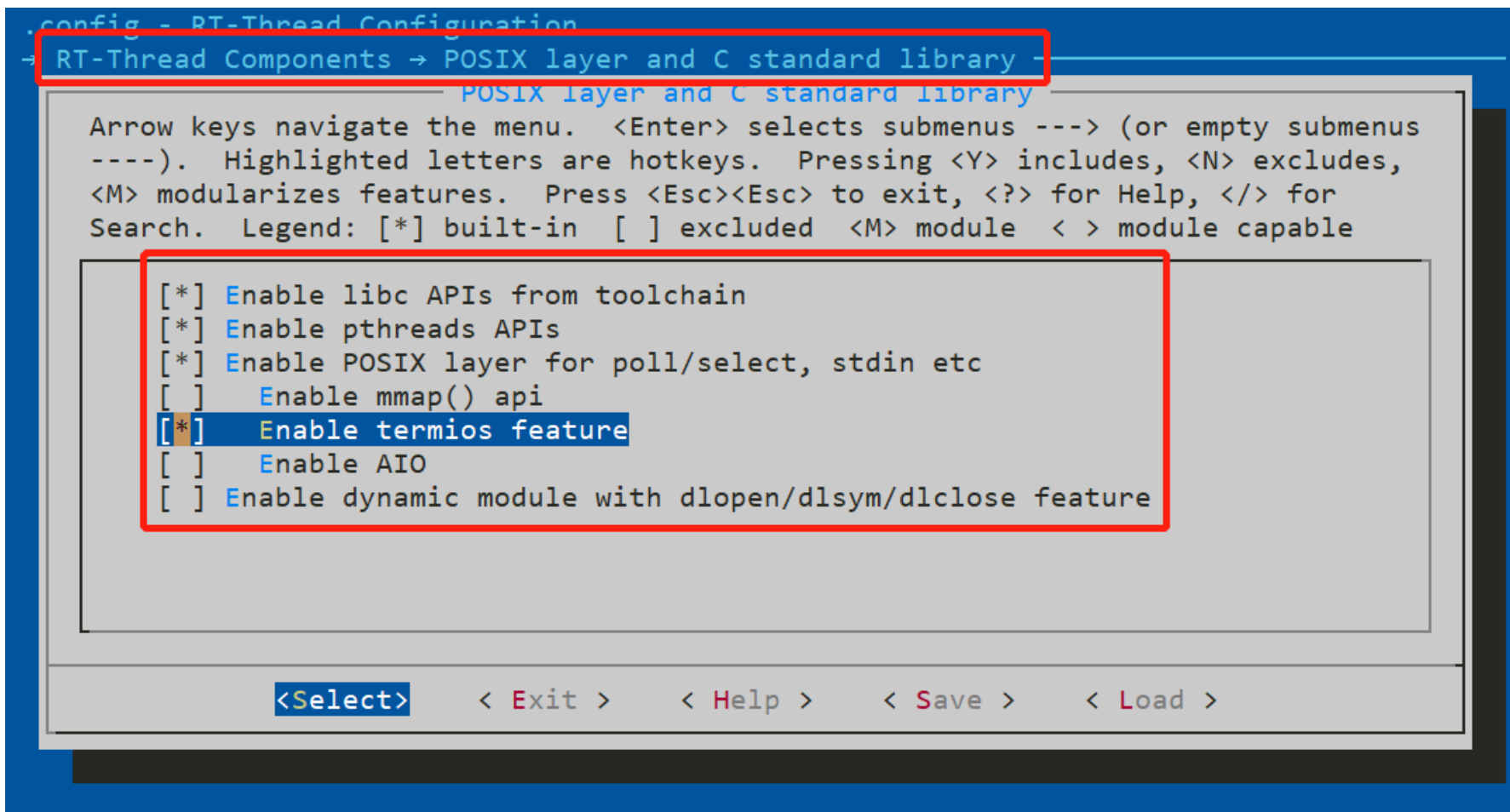
- 仓库地址：<https://github.com/mevdschee/2048.c>



开启虚拟文件系统

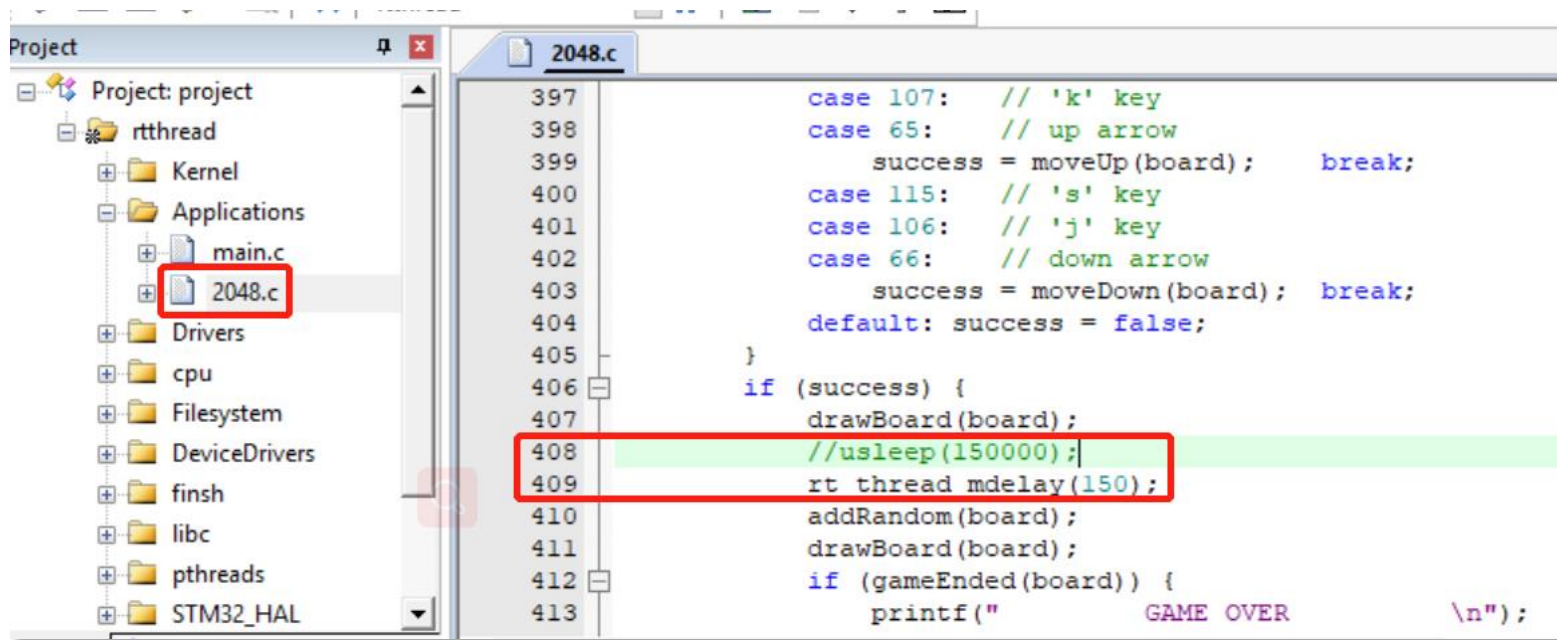


打开POSIX接口



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

```
358 L
359 //int main(int argc, char *argv[]) {
360 int start_2048(int argc, char *argv[]) {
361     uint8_t board[SIZE][SIZE];
362     char c;
363     bool success;
364
```

- 导出函数start_2048到控制台

```
440 }
441 MSH_CMD_EXPORT(start_2048, start 2048)
442
```

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
```

