

**laboratory research**

**#2**

**Developing tasks in FreeRTOS**

Reporters: Juntao Gao

Date:2020-6-24

**Table of contents**

[1 Description of the task 3](#_Toc43902121)

[2 Specific implementation 3](#_Toc43902122)

[2.1 The description of project creation 3](#_Toc43902123)

[2.2 Compile and download 6](#_Toc43902124)

[3 answer the questions 7](#_Toc43902125)

# 1 Description of the task

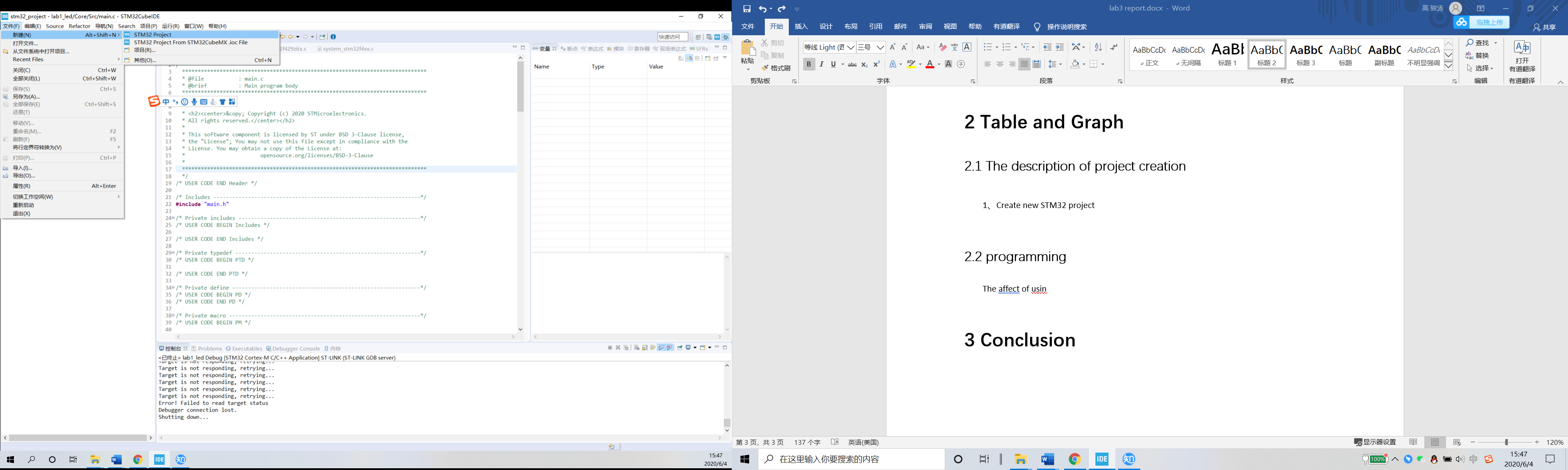
1. Read the FreeRTOS and Git documentation
2. Install FreeRTOS using Stm32CubeIDE
3. Create a project
4. Create two tasks
5. From each task, control the LEDs at different frequencies
6. Put the source code of the project on GitHub
7. Read the FreeRTOS documentation and answer the questions in the end of this

document

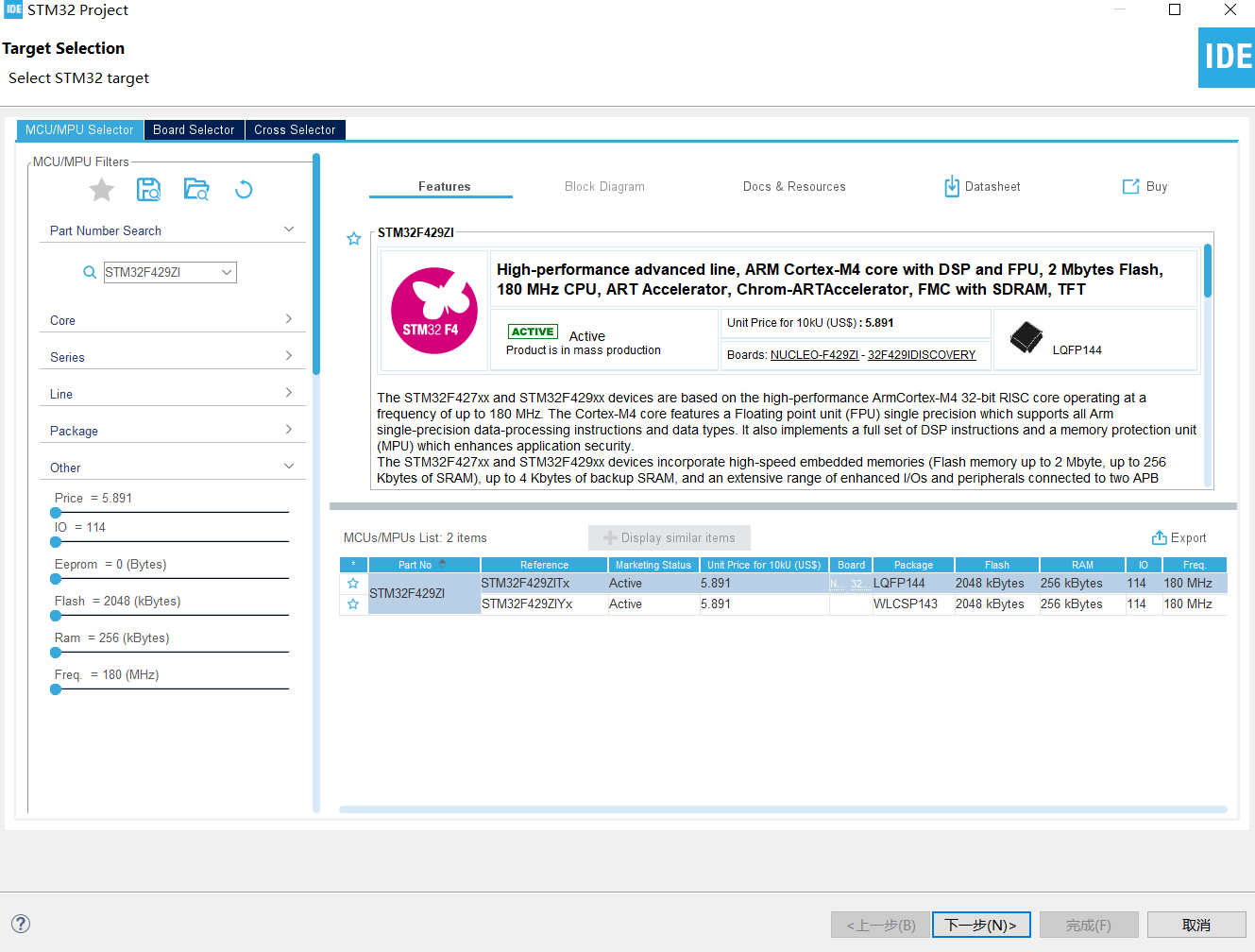
# 2 Specific implementation

## 2.1 The description of project creation

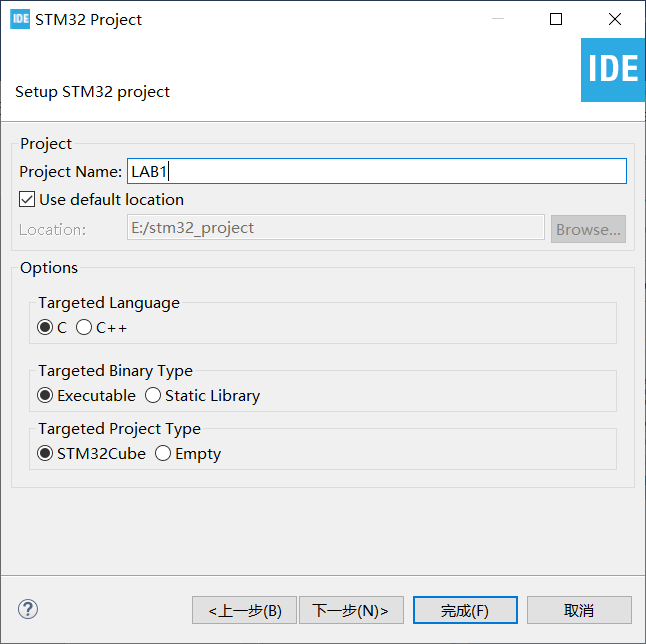
1、Create new STM32 project



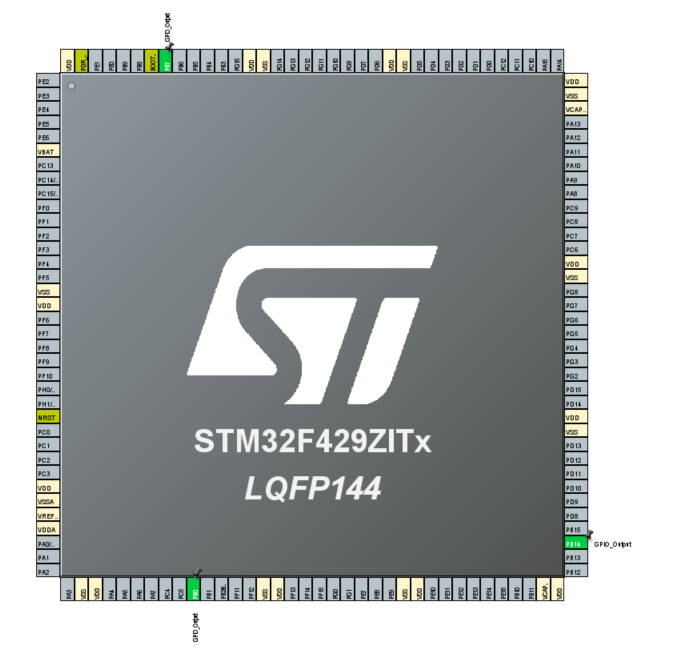
2、According to your hardware to select MCU



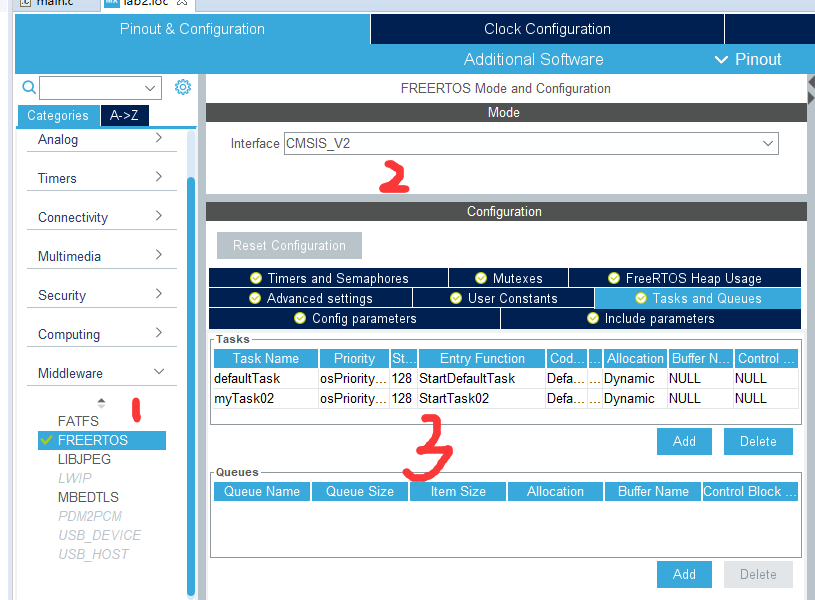
3、Select the name of your project



4、Switch IO port to the output.



5、Set up Freertos

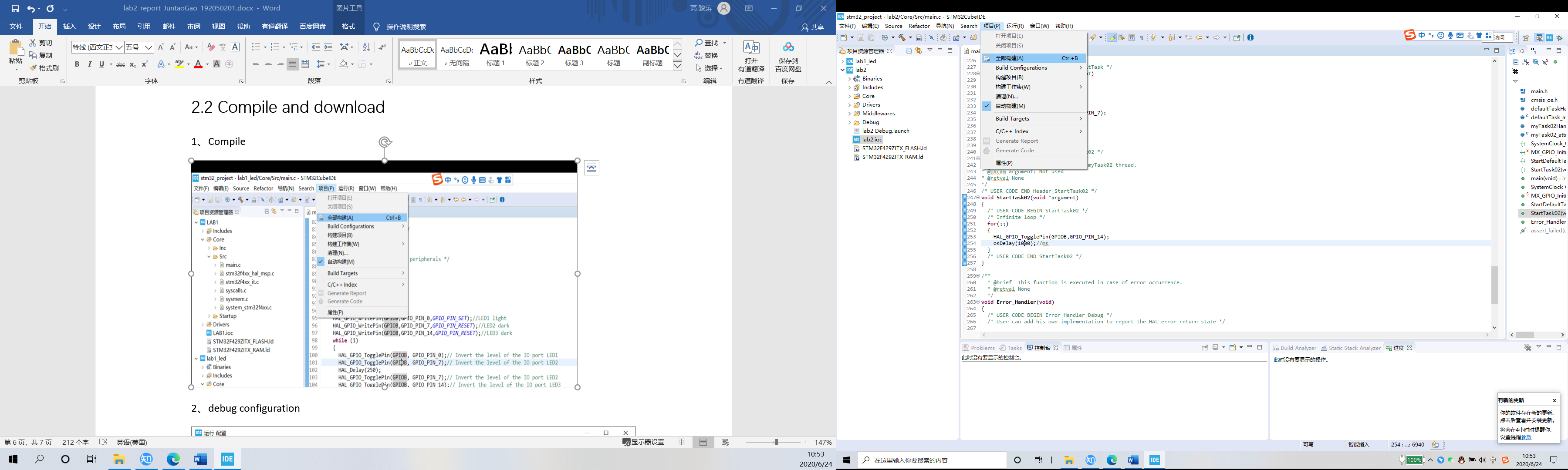


6、coding

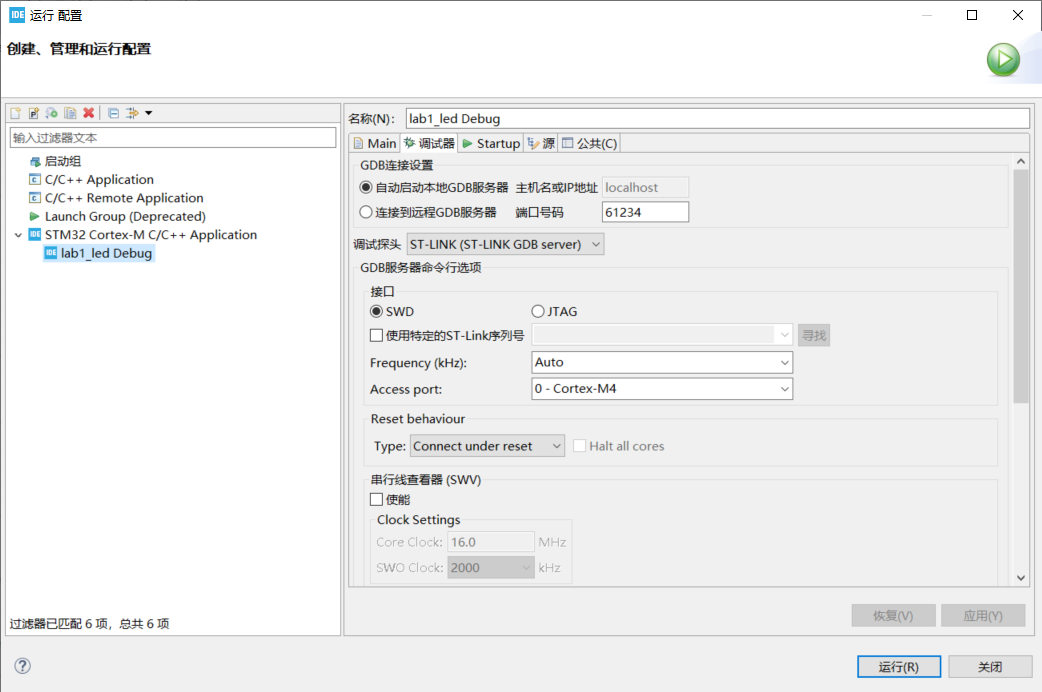
Source code is on github

## 2.2 Compile and download

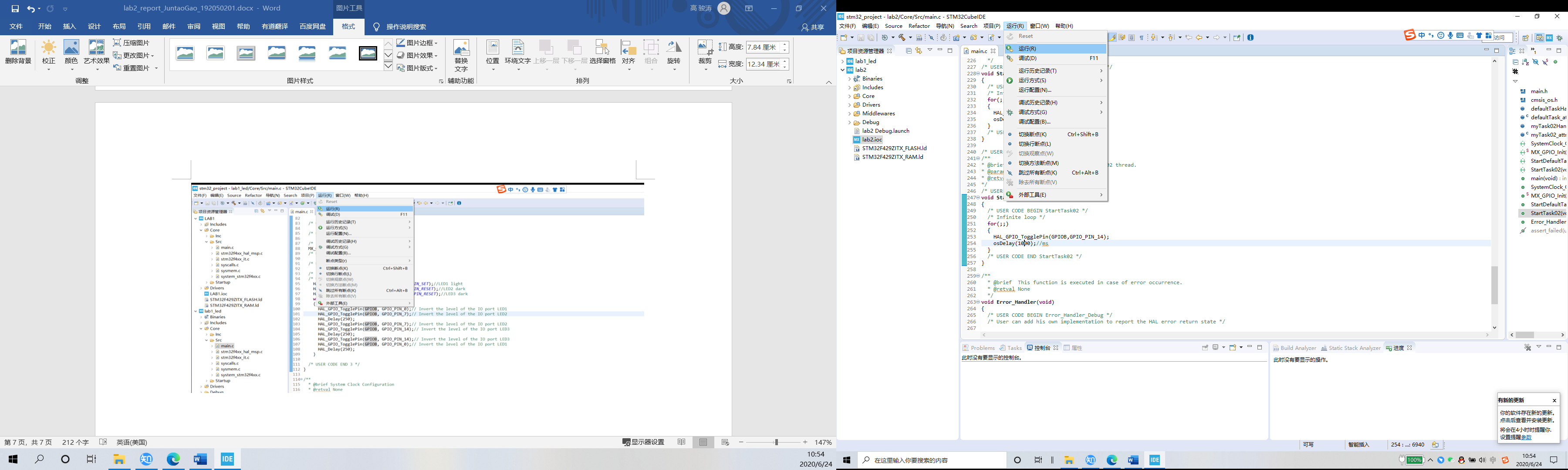
1、Compile



2、debug configuration



3、download



# 3 answer the questions

1. **What features of FreeRTOS do you remember?**

First, FreeRTOS is an open source, portable, and small embedded real-time operating system kernel. It supports both preemptive multitasking and collaborative multitasking.

In this experiment I found the following features of freertos:

1、Users can configure kernel functions according to their own needs

2、The object code is small and easy to use

3、A hard real-time operating system kernel, which can be modified to a non-real-time kernel based on knowledge

4、Can support multitasking

In addition, the features found through the documentation are:

1、FreeRTOS has no limit on the number of tasks, and there can be multiple tasks at the same priority

2、Very flexible task priority assignment, either preemptive or collaborative. The scheduler will decide the task scheduling every time the timed interrupt comes, and external asynchronous events will also cause the scheduler task scheduling

3、FreeRTOS supports queues and several basic task synchronization mechanisms

4、Multi-platform support, open source and free

1. **What are the differences between FreeRTOS, OpenRTOS and SafeRTOS?**

First of all, freertos is open source and free, so

1. FreeRTOS can be used for commercial applications.

2. FreeRTOS itself is still free for everyone to use.

3. FreeRTOS users retain their intellectual property rights.

OpenRTOS and FreeRTOS share the same source code. OpenRTOS is a commercial licensed version of FreeRTOS, provided by a third-party Real Time Engineers Ltd. license.

There are two main reasons for users to update from FreeRTOS to OpenRTOS:

1. In order to overcome the GPL license limitation of the modified version of FreeRTOS.

2. In order to obtain additional services, such as professional technical support, high-quality middleware, training, consulting and corresponding tools

SafeRTOS and FreeRTOS have the same usage model. A lot of optimizations have been done in terms of security. For example, high-risk functions such as dynamic memory allocation functions are optimized or simply not used. In addition, protection has been made in the stack allocation of tasks. Its development is based on the practices, procedures, and processes necessary to claim compliance with various internationally recognized safety-related standards. The most important thing is that SafeRTOS has passed a European safety certification, enabling it to be used in medical, industrial, and other high safety field.

1. **Why do we need the vTaskStartScheduler() function?**

Because the FreeRTOS scheduler is started by the vTaskStartScheduler() function. By starting the scheduler, the highest priority task created when the scheduler is in the "initialized" state will enter the "running" state.

It is usually encapsulated in osKernelStart(), and its work is as follows:

1、xTaskCreate() creates idle tasks with the lowest priority: 0;

2、Close the interrupt function and enable the task scheduling function;

3、system runtime statistics initialization;

4、Set the system tick timer and start the first task;

5、Return the idle task handle.

1. **Why do we need the xTaskCreate() function?**

In the case of using FreeRTOS-MPU, use xTaskCreate () function can create a task running in privileged mode or user mode. So we need the xTaskCreate() function to create a new task and add it to the task queue, ready to run. If the task is successfully created and added to the ready list, the function returns pdPASS, otherwise the function returns an error code.