

Embedded System

Training 1

August 30, 2022,

Overview

In our first training section, we are going to cover the following stuff:

1. The basic grammar of the C language
 - a. <https://www.tutorialspoint.com/cprogramming/index.htm>
2. The basic knowledge of Hardware
 - a. A board
 - i. manual
 - ii. schematics
 - b. STM32
 - i. stm32f427ih6
3. How to light up a LED?
 - a. GPIO
 - i. push pull / open drain
 - ii. pull up/ pull down

The basic knowledge of the C language

You can read only the highlight part :)

Tutorial Link: <https://www.tutorialspoint.com/cprogramming/index.htm>

LEARN C PROGRAMMING

c programming language

C Programming Video Tutorials

C Programming Tutorial

C - Home

C - Overview

C - Environment Setup

C - Program Structure

C - Basic Syntax

C - Data Types

C - Variables

C - Constants

C - Storage Classes

C - Operators

C - Decision Making

C - Loops

C - Functions

C - Scope Rules

C - Arrays

C - Pointers

C - Strings

C - Structures

C - Unions

C - Bit Fields

C - Typedef

C - Input & Output

C - File I/O

C - Preprocessors

C - Header Files

C - Type Casting

C - Error Handling

C - Recursion

C - Variable Arguments

C - Memory Management

C - Command Line Arguments

C Programming useful Resources

C - Questions & Answers

C - Quick Guide

C - Useful Resources


SIGNATURE
HARDWARE

Need inspiration for your next
renovation? Visit SignatureHardware.com

[Previous Page](#)
[Next Page](#)

Before we study the basic building blocks of the C programming language, let us look at a bare minimum C program structure so that we can take it as a reference in the upcoming chapters.

Hello World Example

A C program basically consists of the following parts –

- Preprocessor Commands
- Functions
- Variables
- Statements & Expressions
- Comments

Let us look at a simple code that would print the words "Hello World" –

```
#include <stdio.h>

int main() {
    /* my first program in C */
    printf("Hello, World! \n");
}

return 0;
}
```

Live Demo

Let us take a look at the various parts of the above program –

- The first line of the program `#include <stdio.h>` is a preprocessor command, which tells a C compiler to include stdio.h file before going to actual compilation.
- The next line `int main()` is the main function where the program execution begins.
- The next line `/* ... */` will be ignored by the compiler and it has been put to add additional comments in the program. So such lines are called comments in the program.
- The next line `printf(...)` is another function available in C which causes the message "Hello, World!" to be displayed on the screen.
- The next line `return 0;` terminates the main() function and returns the value 0.

Compile and Execute C Program

Let us see how to save the source code in a file, and how to compile and run it. Following are the simple steps –

- Open a text editor and add the above-mentioned code.
- Save the file as `hello.c`.
- Open a command prompt and go to the directory where you have saved the file.
- Type `gcc hello.c` and press enter to compile your code.
- If there are no errors in your code, the command prompt will take you to the next line and would generate `a.out` executable file.
- Now, type `a.out` to execute your program.
- You will see the output "Hello World" printed on the screen.

```
$ gcc hello.c
$ ./a.out
Hello, World!
```

Make sure the gcc compiler is in your path and that you are running it in the directory containing the source file

Voted #1
lowest prices'
in Pennsylvania.



*Based on a 2021 Market Consumer Perception Study of Pennsylvania Shoppers.

Voted #1
lowest prices'
in Pennsylvania.



*Based on a 2021 Market Consumer Perception Study of Pennsylvania Shoppers.

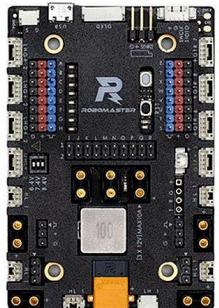
Microsoft Azure

Build your next great app with pay-as-you-go pricing

Get started with 40+ free services.

[Sign up](#)

Basic Knowledge of Hardware



Development Board Type A



Development Board Type B



Development Board OLED



RM Development Board Cables

Important document about those Development Boards

- Manual:
 - It contains the usage of the board, pin diagram, and model of onboard devices
- Schematic(Heavily used in our project):
 - It shows what components will be used in a circuit and how they are connected together.

Documents' location

- They will be on the Teams
 - The path is: manual collection/Development_Boards_Manual/A_Board

The screenshot shows a digital workspace interface. At the top, there's a search bar and a navigation bar with 'Posts' and 'Files' tabs. Below that is a toolbar with 'New', 'Upload', 'Edit in grid view', 'Share', 'Copy link', 'All Documents', and a filter icon. The main area displays a hierarchical file structure: 'embedded system' > 'manual collection' > 'Development_Boards_Manual' > 'A_Board'. The 'A_Board' folder is underlined with a red wavy line. Inside 'A_Board', there are two items: 'Manual' and 'Schematic'. Both items have a small yellow folder icon, a timestamp '2 days ago', and the name 'Yan, Haonan' next to them. There are also dropdown menus for 'Name', 'Modified', and 'Modified By'.

Microcontroller Unit(MCU)

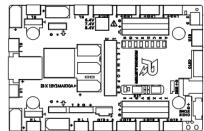
From the A board Manual, we can see it uses a STM32F427IIH6 microcontroller unit.

Introduction

The RoboMaster Development Board Type A is a highly flexible controller board designed to be used in a wide range of robotics projects. It uses an STM32F427IIH6 as its main controller chip and features multiple extension and communication interfaces. It is compatible with the RoboMaster M3508 and M2006 Brushless DC Gear Motors, the RoboMaster UWB Locating System, the DJI Onboard SDK, and the DJI Manifold high-performance embedded computer.

In the Box

Board Type A x 1



Power Cable x 1



A microcontroller is an integrated circuit that's used to run specific operations. So, for example, how does your washing machine translate your input into an efficient wash cycle? It's all possible because of a microcontroller.

stm32 is one kind of MCU.

here is a super youtube video about MCU:

[What is a Microcontroller Unit? \(A Very Different Kind of MCU\)](#)

Important document about STM32IIH6

- RM0090 Reference manual
- STM32F427xx STM32F429x datasheet

Document Location

Those documents are on the Teams as well.

The screenshot shows a Microsoft Teams file library interface. The top navigation bar includes a search bar, a 'Posts' tab, and a 'Files' tab which is currently selected. Below the navigation bar are standard file operations: '+ New', 'Upload', 'Edit in grid view', 'Share', 'Copy link', and a three-dot menu. A 'All' dropdown is also present. The main area displays a list of files under the path 'embedded system > manual collection > STM32_Chips_Manual > STM32F427IIH6--A_Board'. The list includes three PDF files: 'pm0214-stm32-cortexm4-mcus-and-mpus...' (modified 2 days ago by Yan, Haonan), 'rm0090-stm32f405415-stm32f407417-stm3...' (modified 2 days ago by Yan, Haonan), and 'stm32f429zi.pdf' (modified 2 days ago by Yan, Haonan). The columns for the list are 'Name', 'Modified', 'Modified By', and '+ Ad'.

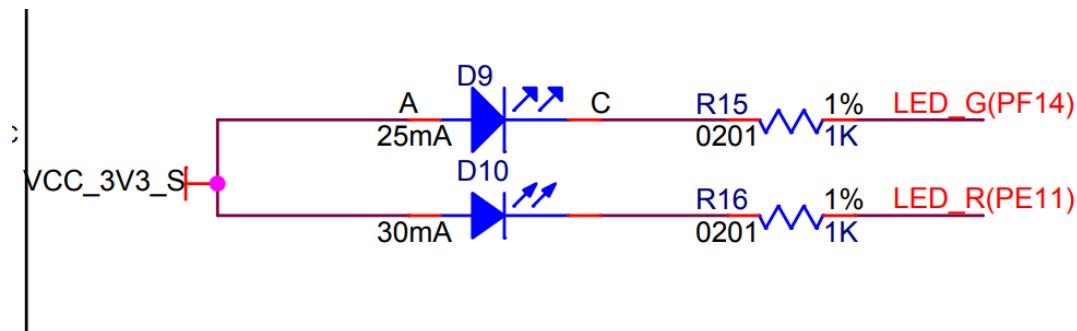
Name	Modified	Modified By	+ Ad
pm0214-stm32-cortexm4-mcus-and-mpus...	2 days ago	Yan, Haonan	
rm0090-stm32f405415-stm32f407417-stm3...	2 days ago	Yan, Haonan	
stm32f429zi.pdf	2 days ago	Yan, Haonan	

How to light up a LED?

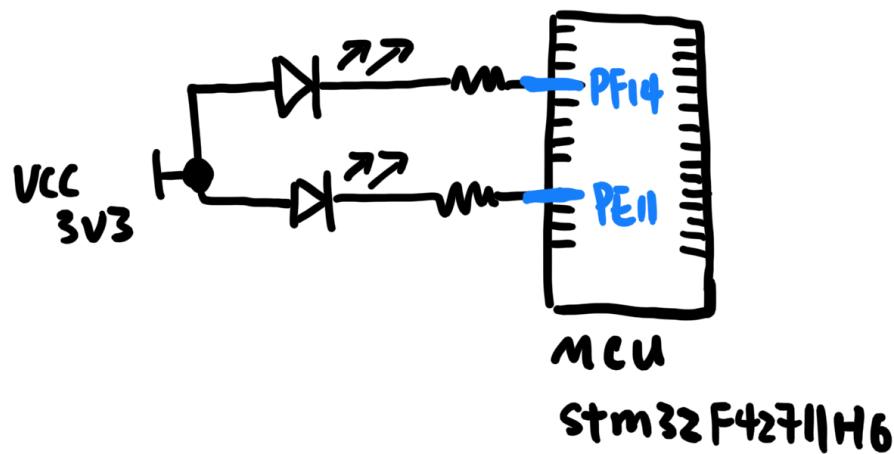
Basic Idea:

We are going to look at the schematics first. You can find the document following the instruction on [page 4](#)

Schematics

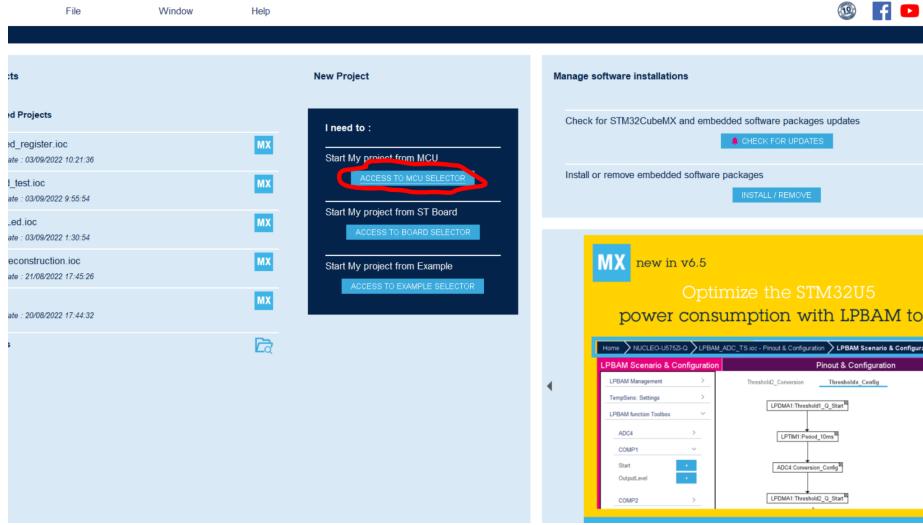


This is a LED schematic from A board. We can see the LED has a 3.3V, so if we set the voltage of the pin PF14 and PE11 low, there will be a voltage drop in the circuit, and LED_G and LED_R will light up. If you don't know why setting those two pins to 0V will light up the led you can look at this video [How to Build a Simple LED Circuit - Electronics for Absolute Beginners](#).



How can we do it?

we are going to use CubeMX to do the task. Here are the steps to initialize the code.



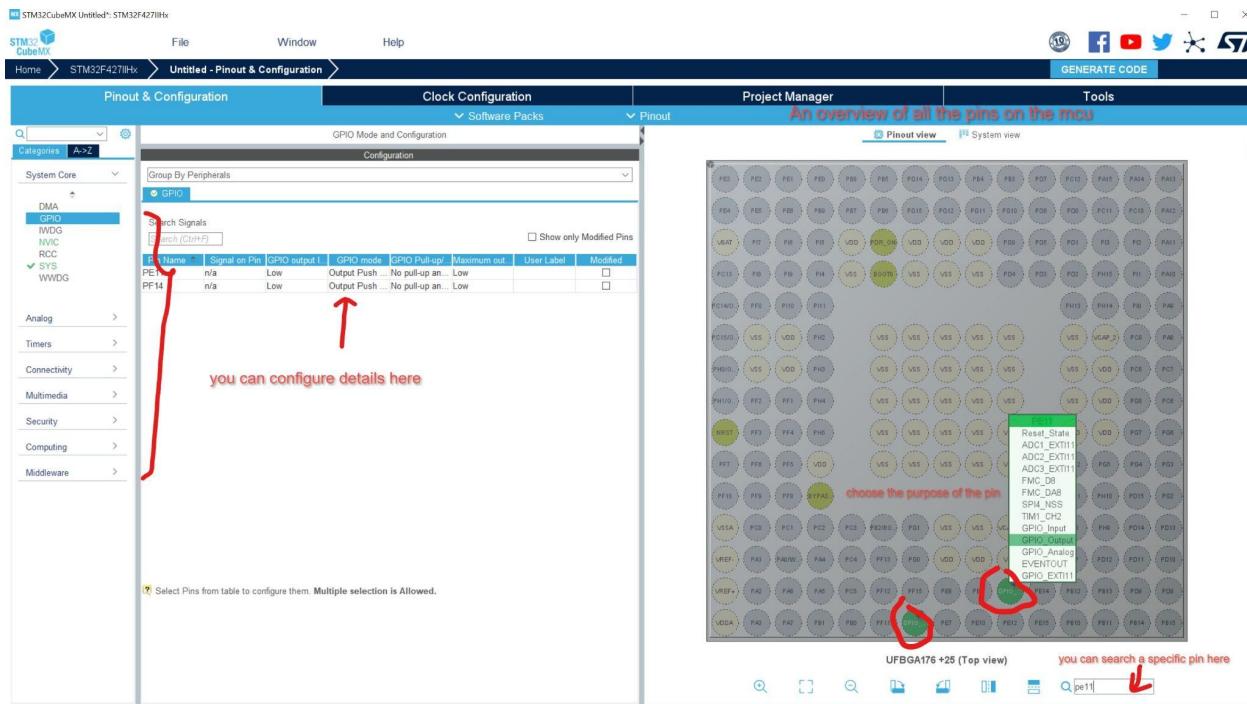
Then, type STM32F427IH6 on the top left box, and select the top one on the right. Then hit start project on the top right button.

we get the commercial part number from the A board manual guide. You can find that info on [page 4](#)

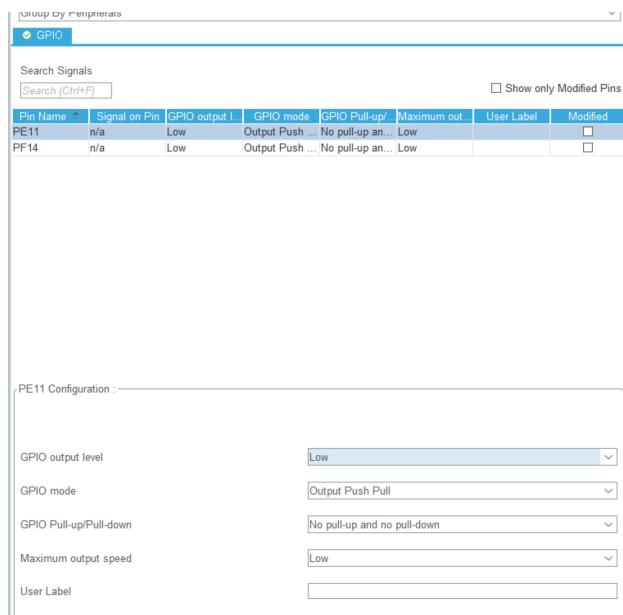
Commercial Part Number	Reference	Marketplace Status	Last Price for Model (USD)	Board	Package	Flash	RAM	I/O	Frequency
STM32F427IH6	STM32F427IH6	Active	\$0.413		UFBGA 176x25 10x10x0.6 P 0.05 mm	256 Kbytes	256 Kbytes	140	160 MHz
STM32F427IHTR	STM32F427IHTR	Active	\$0.613		UFBGA 176x25 10x10x0.6 P 0.05 mm	2048 Kbytes	256 Kbytes	140	160 MHz

CubeMX

Then, you will have a page like this.



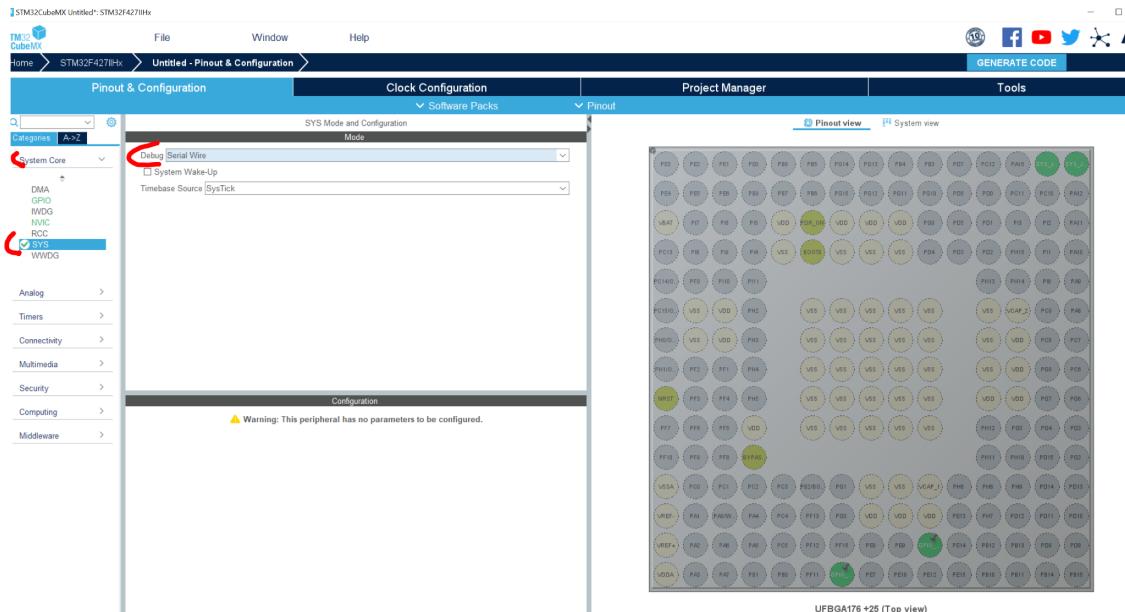
1. General Purpose Input Output(GPIO)
2. Search PE11 on the right bottom search box. Click the pin in the picture and select GPIO_Output
3. Search PF14 on the right bottom search box. Click the pin in the picture and select GPIO_Output
4. Then, we look at the left side. Click System Core first and then click GPIO. If you select one of the pins, you can see a page like this.



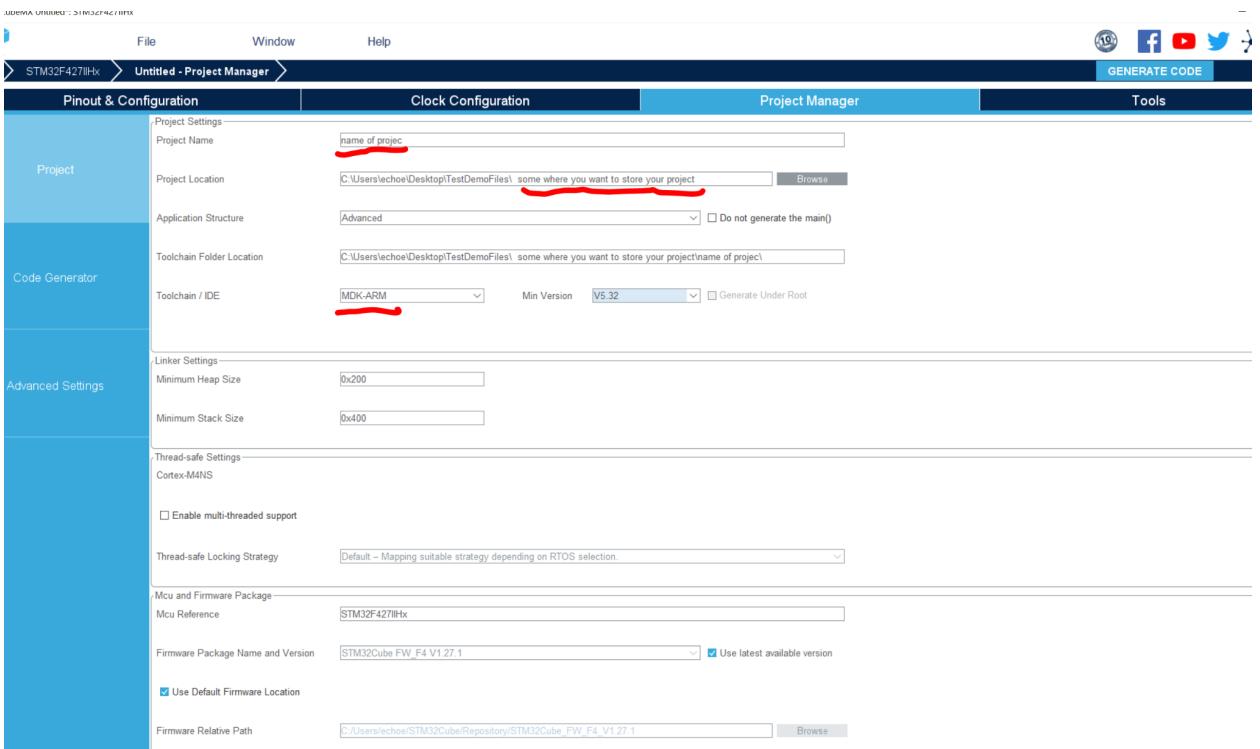
5. You can see there are 5 things in the configuration.

- a. GPIO output level: low means low voltage, and high means high voltage. In our case, we want to set it too low to light up the LED.
 - b. GPIO mode: it decides which circuit design you want to drive the output.
 - i. push pull
 1. [How GPIO works | General Purpose Input Output | GPIO Behind...](#)
 - ii. open drain
 1. [GPIO Output Configuration | Open Drain configuration | Push ...](#)
 - c. GPIO Pull-up/Pull-down: something about circuit design
 - i. [Pull up and pull down resistors preventing false triggering 2N3904 N...](#)
 - d. Maximum output speed:
6. set the configuration the same as in the picture.

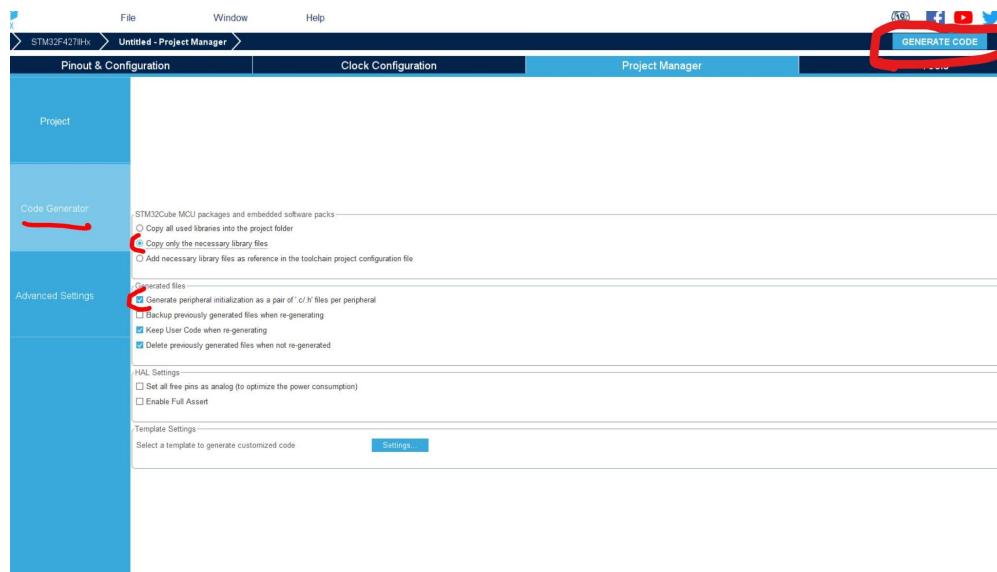
7. On the left, click System Core and Click SYS. Change Debug setting to Serial Wire.



8. Go to Project Manager, give your project a name, store it somewhere you like, and change IDE to MDK-ARM.



9. Go to Code Generator right now select copy only the necessary library files, and generate peripheral initialize as a pair of .c/.h files per peripheral. Then you can hit generate code on the top right.



10. Then, select “open project”. A Keil project will pop up, press the build button on the top left. 0 error and 0 warning means we have already done building the project

The screenshot shows the Keil uVision IDE interface. The title bar indicates the path: C:\Users\echoe\Desktop\TestDemoFiles\aaa\MDK-ARM\aaa.uvprojx - uVision. The menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The toolbar has various icons for file operations. The Project Explorer on the left shows the project structure: Project: aaa, Application/MDK-ARM, Application/UserCore, Drivers/STM32F4xx_HAL_Driver, Drivers/CMIS, and CMSIS. The main editor window displays the 'main.c' source code. The status bar at the bottom provides build information.

```

37 /* Private macro */
38 /* USER CODE BEGIN PM */
39
40 /* USER CODE END PM */
41
42 /* Private variables */
43
44 /* USER CODE BEGIN PV */
45
46 /* USER CODE END PV */
47
48 /* Private function prototypes */
49 void main(void);
50
51 /* USER CODE BEGIN PFP */
52
53 /* USER CODE END PFP */
54
55 /* Private user code */
56
57 /* USER CODE BEGIN 0 */
58
59 /**
60 * brief: The application entry point.
61 * detail: Int
62 */
63 int main(void)
64 {
65     /* USER CODE BEGIN 1 */
66
67     /* USER CODE END 1 */
68
69     /* MCU Configuration*/
70
71     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
72     HAL_Init();
73
74     /* USER CODE BEGIN Init */
75
76     /* USER CODE END Init */
77
78     /* Configure the system clock */
79     SystemClock_Config();
80
81     /* USER CODE BEGIN SysInit */
82
83     /* USER CODE END SysInit */

```

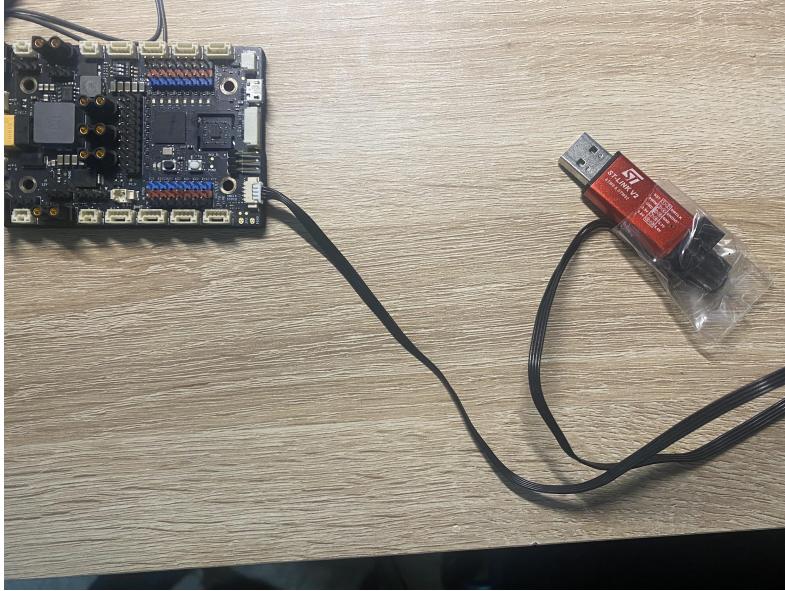
Build Output

```

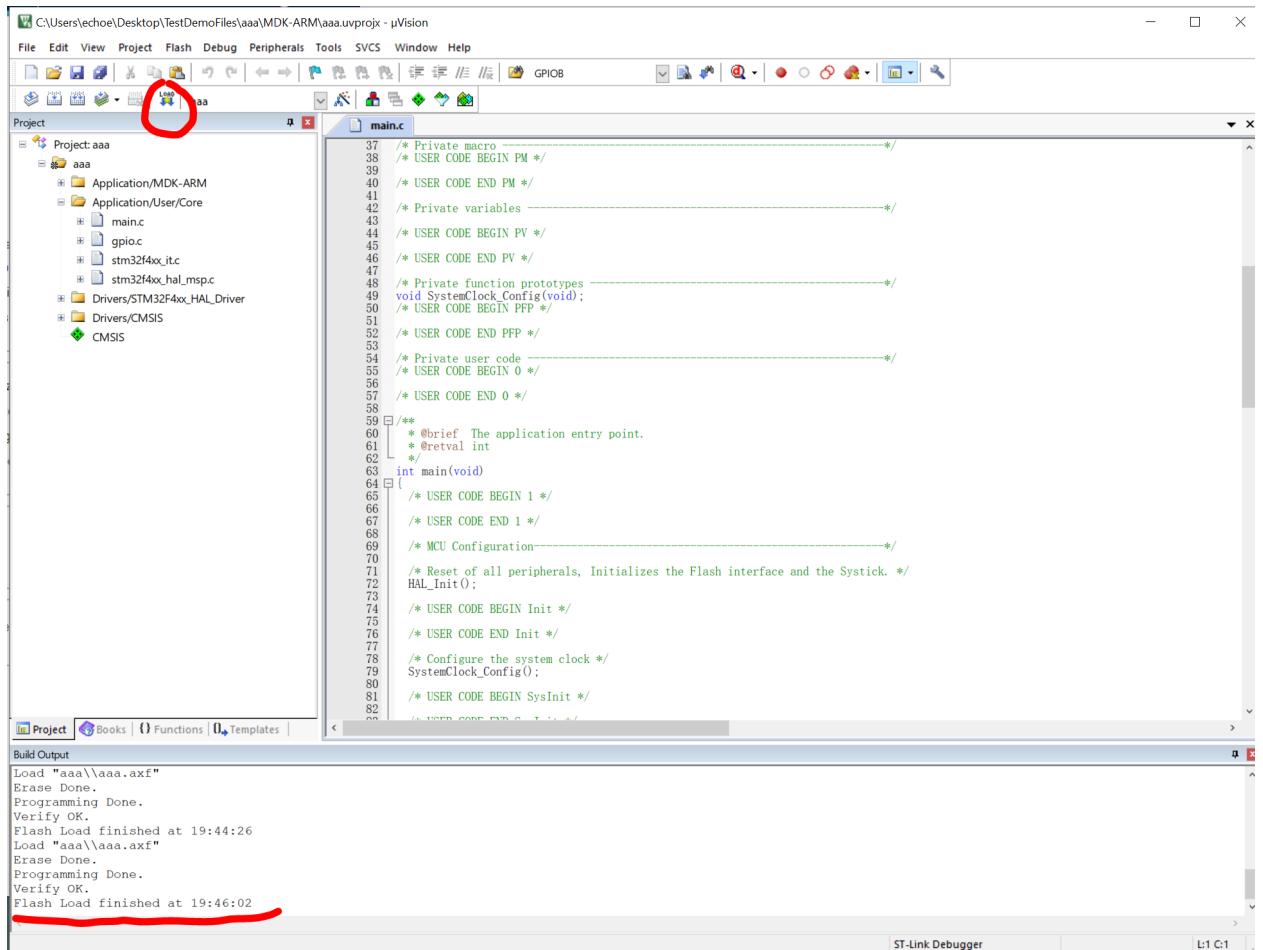
compiling stm32f4xx_hal_exti.c...
compiling system_stm32f4xx.c...
compiling stm32f4xx_hal_pwr_exx.c...
compiling stm32f4xx_hal_cortex.c...
compiling stm32f4xx_hal_pwr.c...
linking...
Project size: Code=3230 RO=data=478 RW=data=16 ZI=data=1632
FromELF: creating hex file...
"aaa\aaa.axf" - 0 Error(s), 0 Warning(s).
Built 1 target(s) in 00:00:21.

```

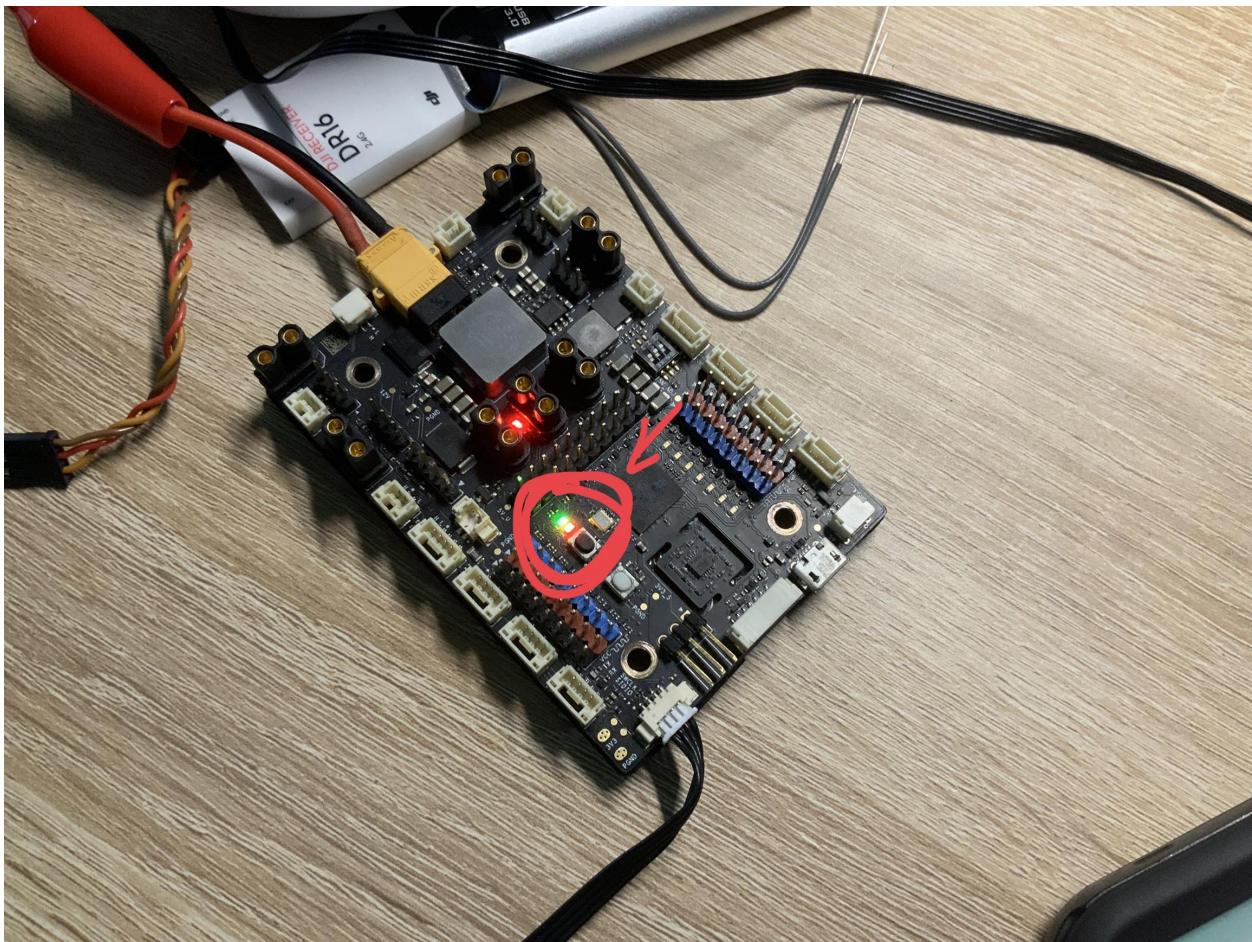
11. Then, connect ST-Link and your pc



12. Then press the load button on the top left.



13. You can see we successfully light up those 2 LEDs!!!



Resource on the Teams

The screenshot shows the Microsoft Teams interface. On the left, there's a sidebar with icons for Activity, Chat, Teams, Assignments, Calendar, Calls, Files, and Apps. The main area shows the 'PSU RoboX' team, specifically the 'General' channel. Within this channel, the 'embedded system' folder is selected. The 'Files' tab is active, displaying a list of files:

Name	Modified	Modified By
Keep all training materials		
Embedded System Trainings	A few seconds ago	Yan, Haonan
this is where you can find all manuals		
manual collection	2 days ago	Yan, Haonan
manual collection -- some in chinese	2 days ago	Yan, Haonan
this is the code that is on current standard robot		
Robomaster_Embdead_Standard-main.zip	About a minute ago	Yan, Haonan

Red annotations are present: 'Keep all training materials', 'this is where you can find all manuals', 'this is the code that is on current standard robot', and the file 'Robomaster_Embdead_Standard-main.zip'.