Embedded System

Training 1

August 30, 2022,

Overview

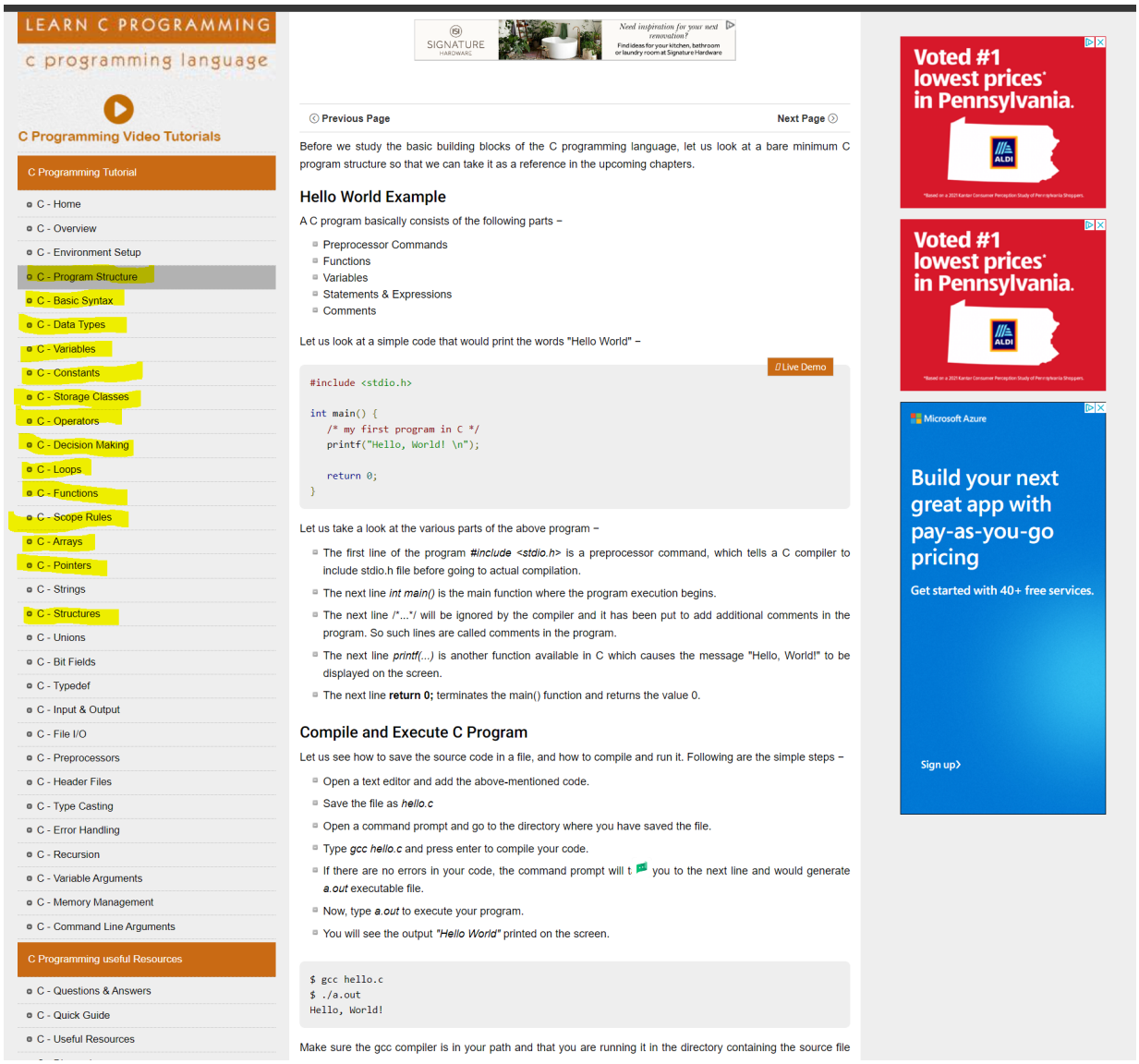
In our first training section, we are going to cover the following stuff:

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. stm32f427iih6
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

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" –

Live Demo

```
#include <stdio.h>

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

   return 0;
}
```

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

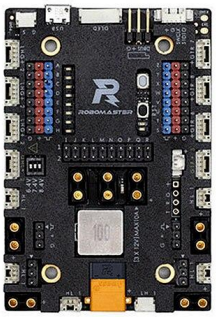
Let us see how to save the source code in a file, and how to compile and run it. Following are the simple steps –

- 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 t 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.

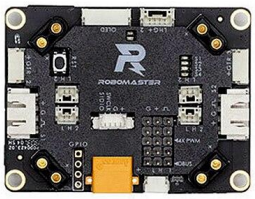
```
$ gcc hello.c
$ ./a.out
Hello, World!
```

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

Basic Knowledge of Hardware



Development Board Type A



Development Board Type B

Development Board OLED

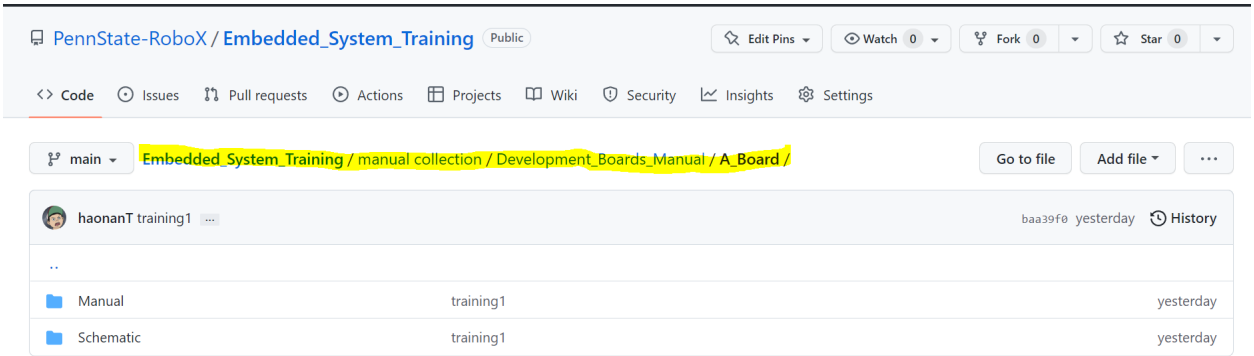
RM Development Board Cables

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 GitHub
  - The path is: manual collection/Development_Boards_Manual/A_Board



Microcontroller Unit(MCU)

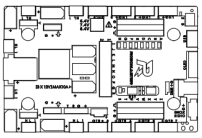
From the A board Manual, we can see it uses a STM32F427IIH6 microcontroller unit.

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 × 1



Power Cable × 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 an awesome youtube video about MCU:

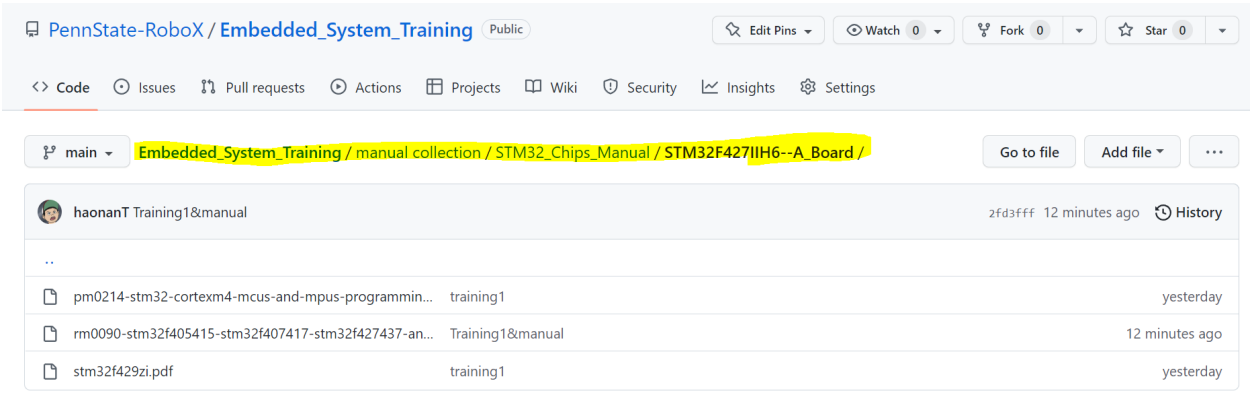
What is a Microcontroller Unit? (A Very Different Kind of MCU)

Important document about STM32IIH6

- RM0090 Reference manual
- STM32F427xx STM32F429x datasheet

Document Location

Those documents are on GitHub as well.

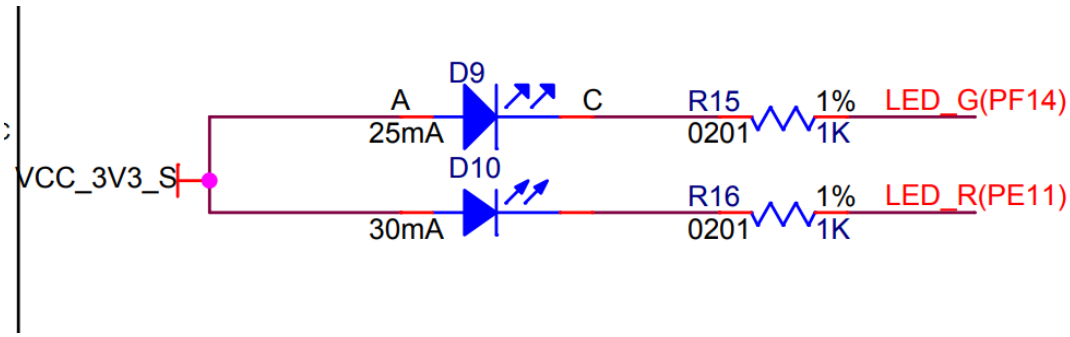


How to light up a LED?

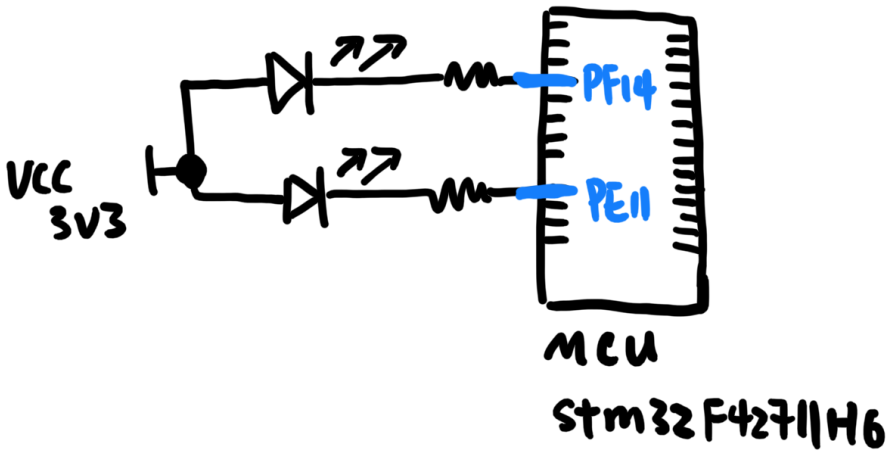
Basic Idea:

We are going to look at the schematics first. You can find the document following the instruction on page 4

Schematics

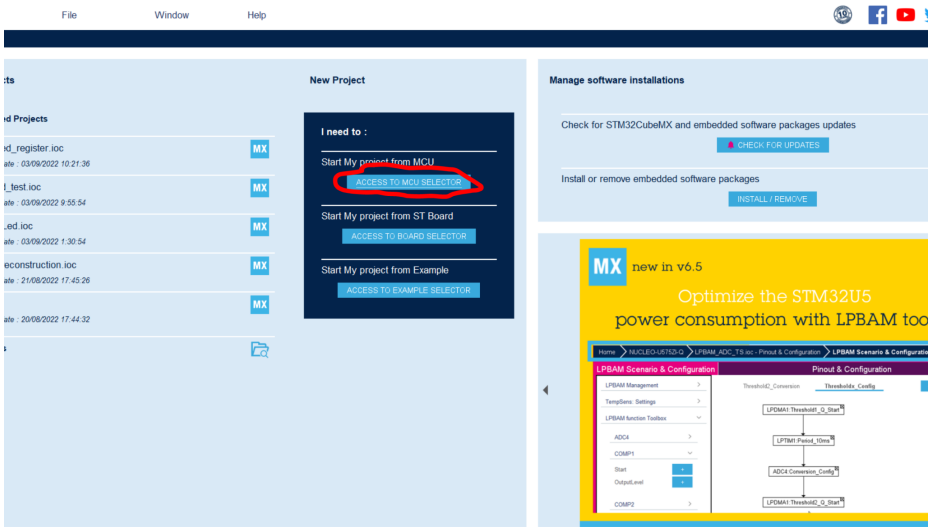


This is a LED schematic from A board. We can see the LED has a 3.3V, so if we set the voltage of the pin PF14 and PE11 low, there will be a voltage drop in the circuit, and LED_G and LED_R will light up. If you don't know why setting those two pins to 0V will light up the led you can look at this video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



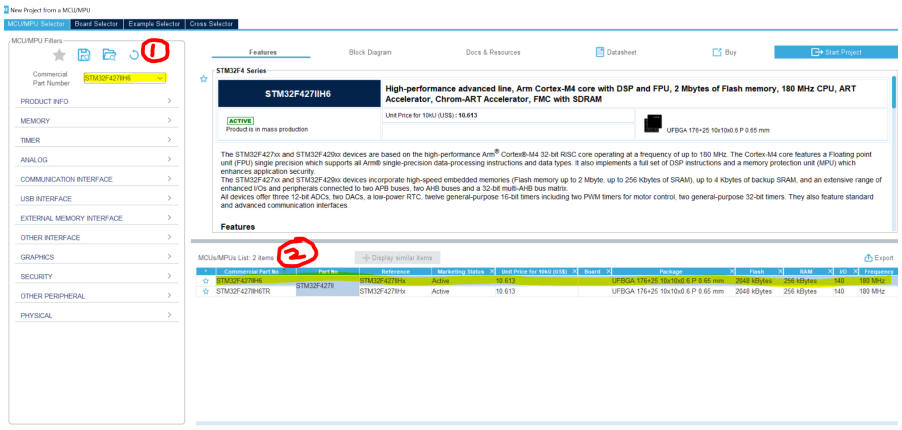
How can we do it?

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



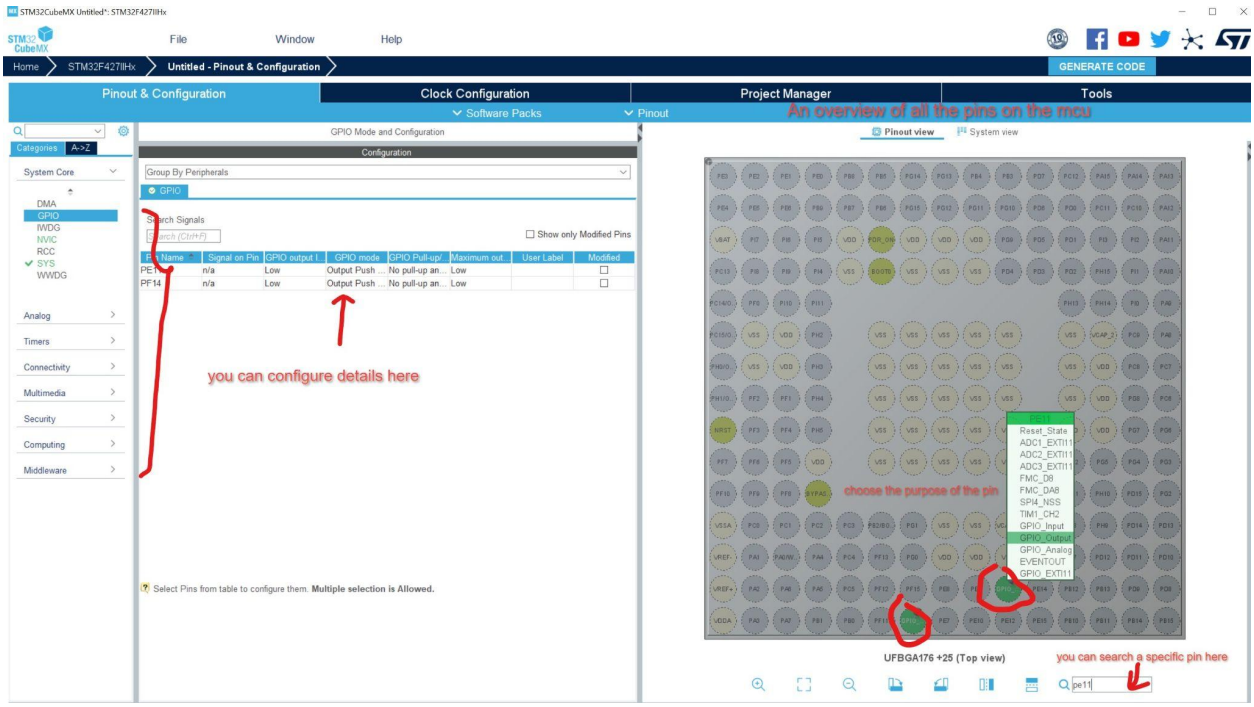
Then, type STM32F427IIH6 on the top left box, and select the top one on the right. Then hit start project on the top right button.

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

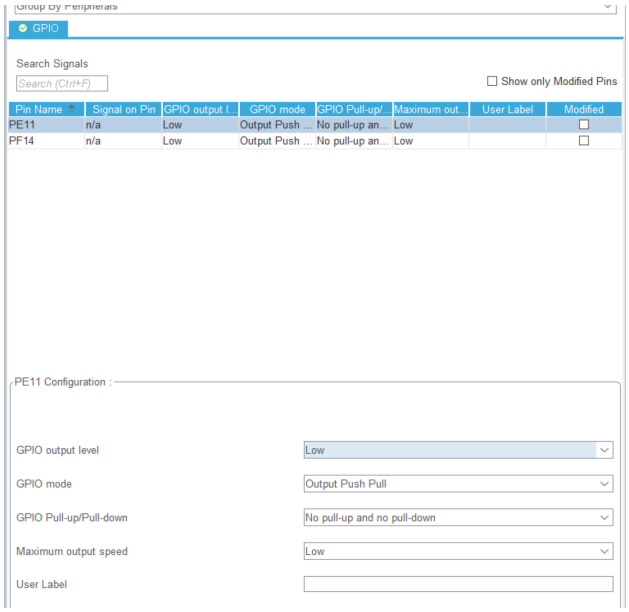


CubeMX

Then, you will have a page like this.

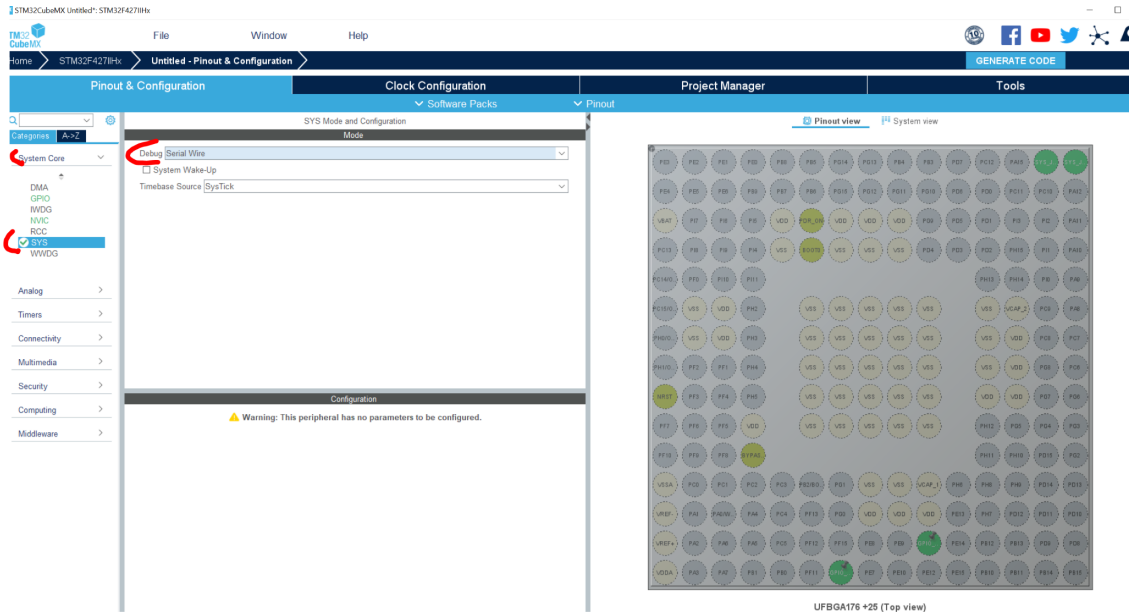


1. General Purpose Input Output(GPIO)
2. Search PE11 on the right bottom search box. Click the pin in the picture and select GPIO_Output
3. Search PF14 on the right bottom search box. Click the pin in the picture and select GPIO_Output
4. Then, we look at the left side. Click System Core first and then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

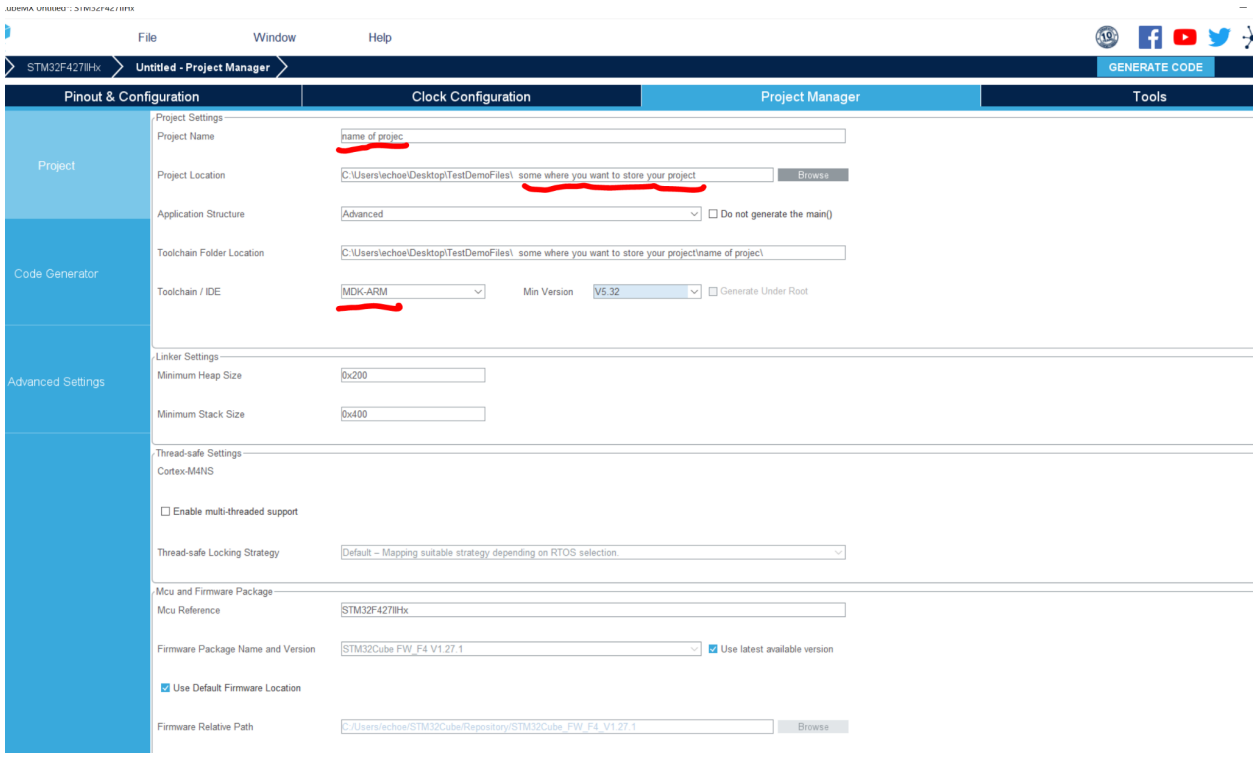


5. You can see there are 5 things in the configuration.
    a. GPIO output level: low means low voltage, and high means high voltage. In our case, we want to set it too low to light up the LED.
    b. GPIO mode: it decides which circuit design you want to drive the output.
        i. push pull
            1. How GPIO works | General Purpose Input Output | GPIO Behin...
        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.

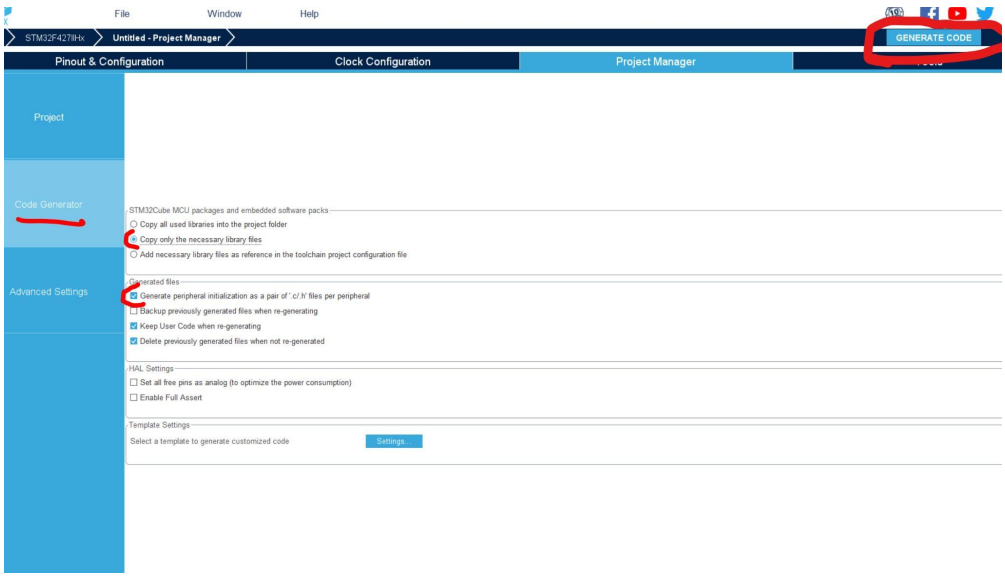
7. On the left, click System Core and Click SYS. Change Debug setting to Serial Wire.



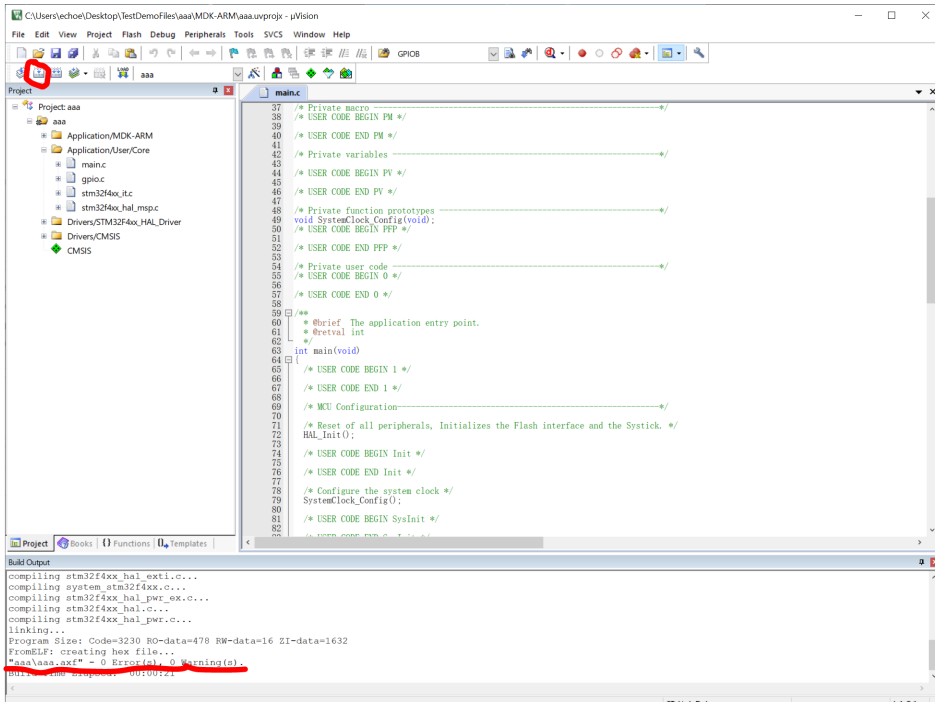
8. Go to Project Manager, give your project a name, store it somewhere you like, and change IDE to MDK–ARM.



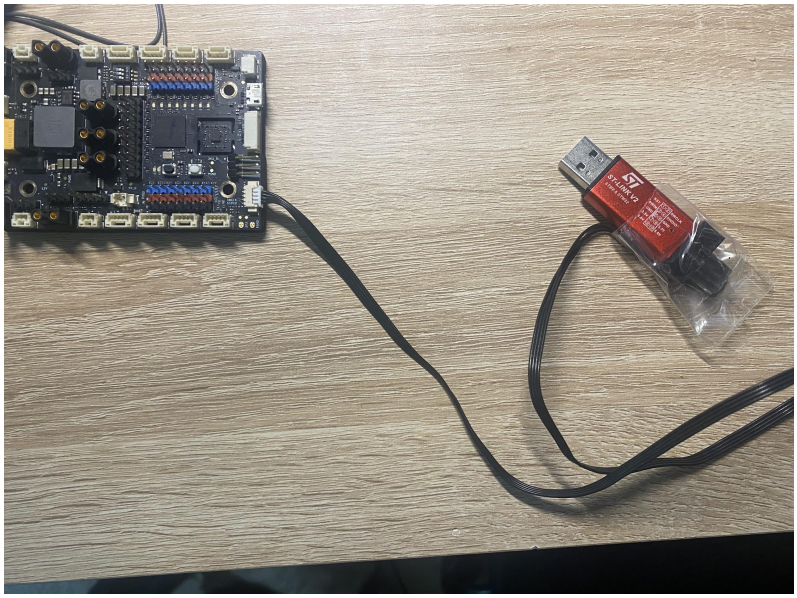
9. Go to Code Generator right now select copy only the necessary library files, and generate peripheral initialize as a pair of .c/.h files per peripheral. Then you can hit generate code on the top right.



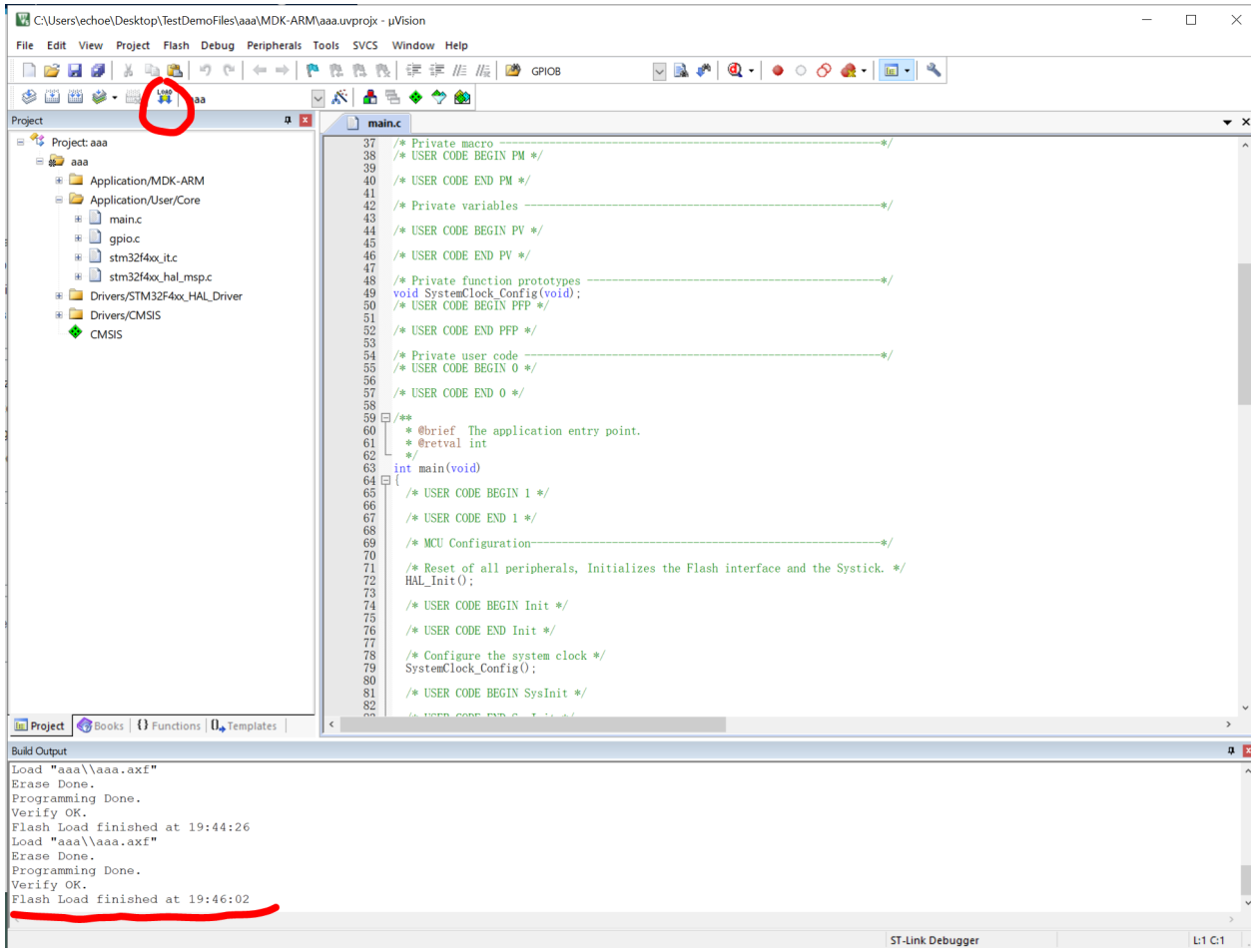
10. Then, select “open project”. A Keil project will pop up, press the build button on the top left. 0 error and 0 warning means we have already done building the project



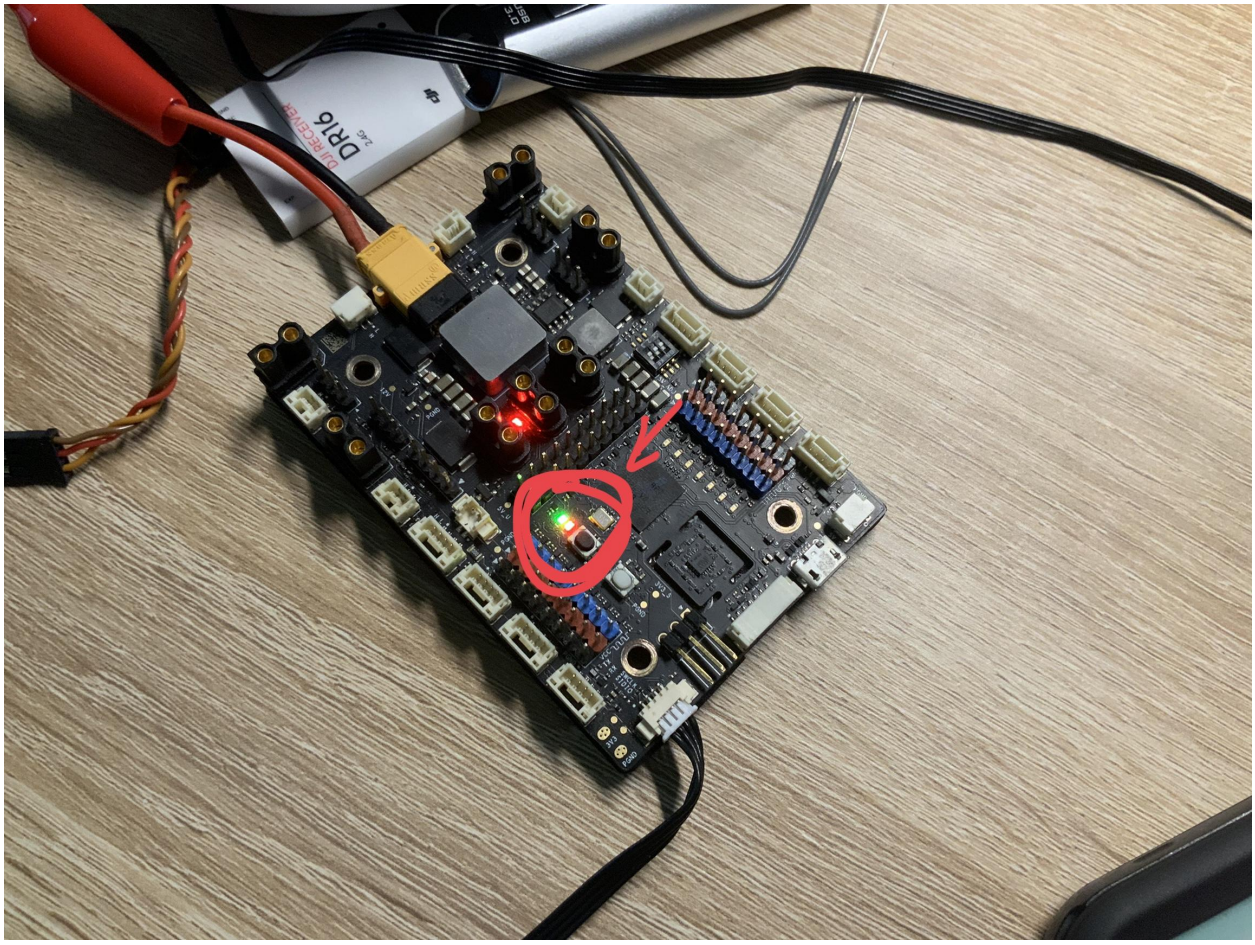
11. Then, connect ST-Link and your pc



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



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



Resource on the GitHub

Links: https://github.com/PennState-RoboX/Embedded_System_Training.git