1. Introduction

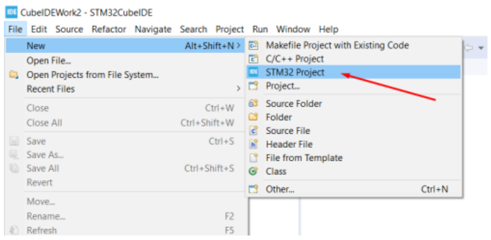
The purpose of this lab is to write a simple program to explore the platform and demonstrate a working system.

2. Prerequisites

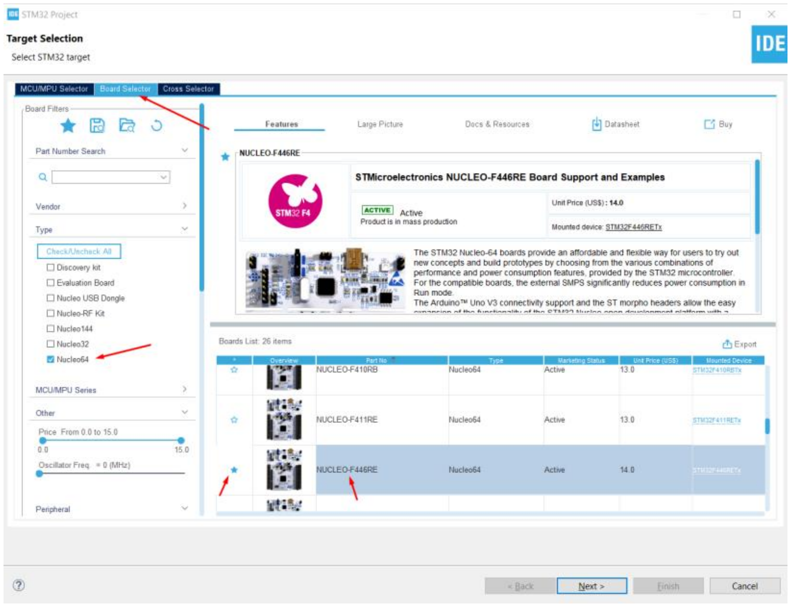
- StmCubeIDE is installed on your machine.
- Real-term or another terminal program is installed on your machine.
- The Nucleo-F411RE board connected to PC with USB cable. (Note: The dev board doesn't need to be completed for this lab.)

3. Create Serial Template Project

1. Create new C project with File -> New -> STM32 Project.



2. The Target Selector will be invoked. Switch to the Board tab and locate the NUCLEO-F411RE. The filters can help narrow down the selections. Clicking the star will make it easier to find in the future. Click Next when ready.



STMicroelectronics NUCLEO-F446RE Board Support and Examples

Active
Product is in mass production

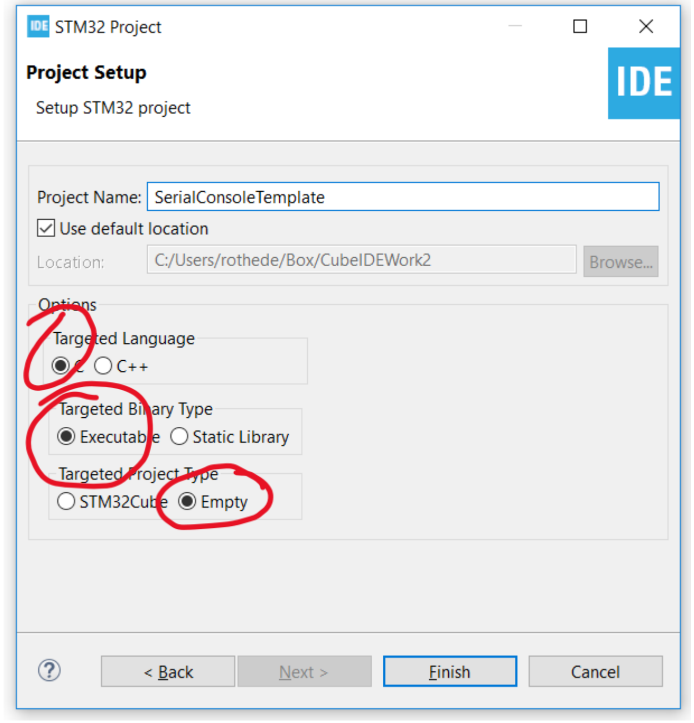
Unit Price (US$) : 14.0

Mounted device: STM32F446RETx

The STM32 Nucleo-64 boards provide an affordable and flexible way for users to try out new concepts and build prototypes by choosing from the various combinations of performance and power consumption features, provided by the STM32 microcontroller. For the compatible boards, the external SMPS significantly reduces power consumption in Run mode.
The Arduino™ Uno V3 connectivity support and the ST morpho headers allow the easy

*	Overview	Part No	Type	Marketing Status	Unit Price (US$)	Mounted Device
☆		NUCLEO-F410RB	Nucleo64	Active	13.0	STM32F410RBTx
☆		NUCLEO-F411RE	Nucleo64	Active	13.0	STM32F411RETx
★		NUCLEO-F446RE	Nucleo64	Active	14.0	STM32F446RETx

3. On the following dialog, give the project a name such as “SerialConsoleTemplate,” C language, Executable, and Empty project. Click Finish to create the new project.



STM32 Project

Project Setup

Setup STM32 project

Project Name: SerialConsoleTemplate

☑ Use default location

Location: C:/Users/rothede/Box/CubeIDEWork2

Options

Targeted Language: [✓ C] C++

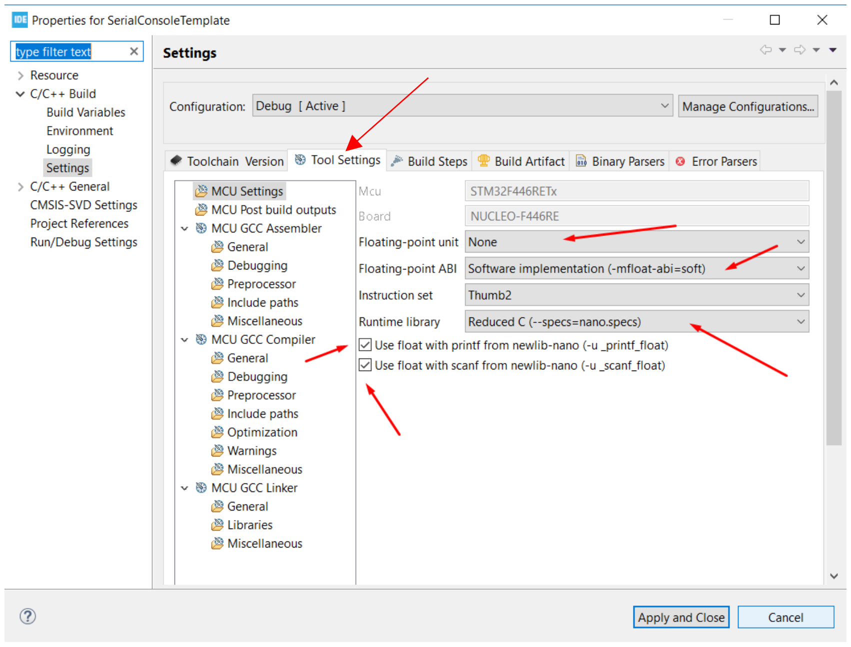
Targeted Binary Type: [✓ Executable] Static Library

Targeted Project Type: STM32Cube [✓ Empty]

4. The supplied zip file should contain three files – main.c, uart_driver.c, and uart_driver.h. Browse to project location in workspace and place files into the proper locations:
   a. main.c and uart_driver.c → Src (main.c will overwrite the existing main.c, leave other files in place)
   b. uart_driver.h → Inc

5. Return to CubeIDE and refresh project (F5 or right-click -> Refresh). The files you added should now show up in the Project Explorer.

6. For good measure, go into the project properties, and make the following settings (these are not strictly needed, but may reduce confusion later):

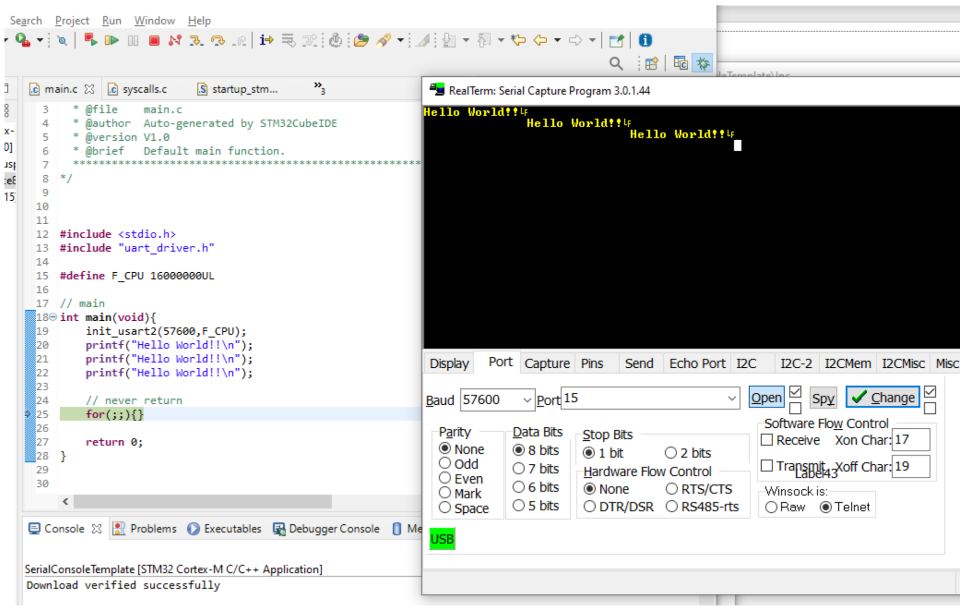


Configuration: Debug [Active]

Mcu	STM32F446RETx
Board	NUCLEO-F446RE
Floating-point unit	None
Floating-point ABI	Software implementation (-mfloat-abi=soft)
Instruction set	Thumb2
Runtime library	Reduced C (--specs=nano.specs)

☑ Use float with printf from newlib-nano (-u _printf_float)
☑ Use float with scanf from newlib-nano (-u _scanf_float)

7. You can test the configuration by building and running the program. Be sure to also run Realterm so you can see the console output.



```c
 * @file     main.c
 * @author   Auto-generated by STM32CubeIDE
 * @version  V1.0
 * @brief    Default main function.
 ******************************************************************************
*/


#include <stdio.h>
#include "uart_driver.h"

#define F_CPU 16000000UL

// main
int main(void){
    init_usart2(57600,F_CPU);
    printf("Hello World!!\n");
    printf("Hello World!!\n");
    printf("Hello World!!\n");

    // never return
    for(;;){}

    return 0;
}
```

```
Hello World!!
             Hello World!!
                          Hello World!!
```

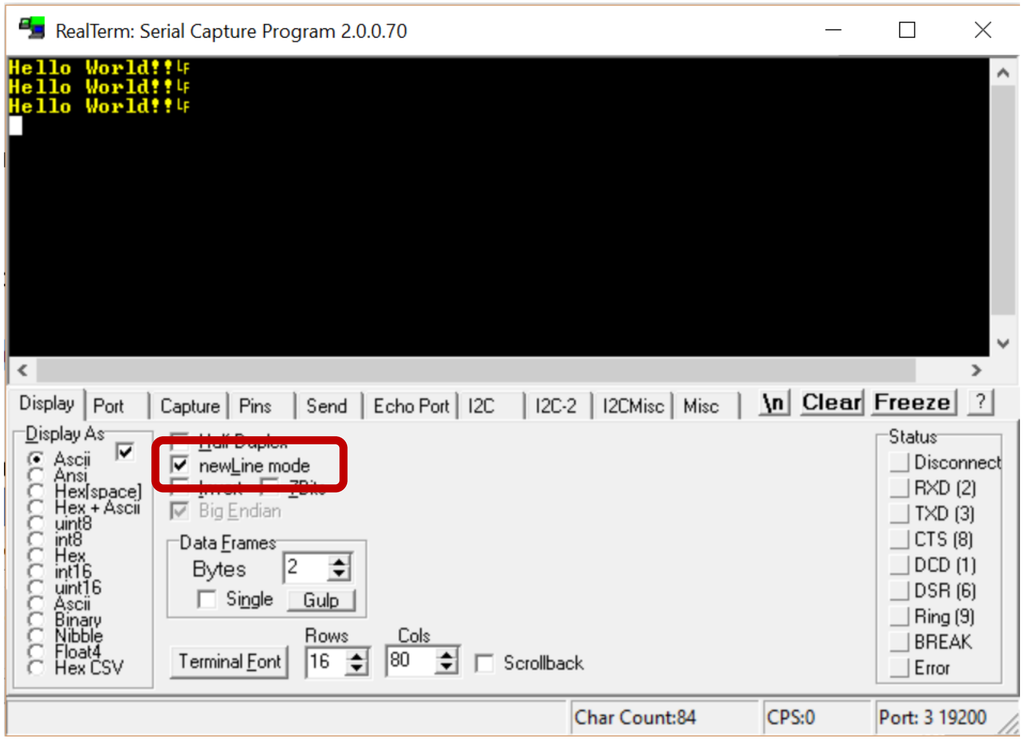
Baud 57600 Port 15

Parity	Data Bits	Stop Bits
● None	● 8 bits	● 1 bit ○ 2 bits
○ Odd	○ 7 bits	Hardware Flow Control
○ Even	○ 6 bits	● None ○ RTS/CTS
○ Mark	○ 5 bits	○ DTR/DSR ○ RS485-rts
○ Space		

Software Flow Control: ☐ Receive Xon Char: 17; ☐ Transmit Xoff Char: 19
Winsock is: ○ Raw ● Telnet

```
SerialConsoleTemplate [STM32 Cortex-M C/C++ Application]
Download verified successfully
```

8. In RealTerm, use “newLine mode” for terminal to behave most similar to an actual console in terms of linefeed and carriage return behavior. Select the port and baud rate under the Port tab.



```
Hello World!!LF
Hello World!!LF
Hello World!!LF
```

[✓] newLine mode

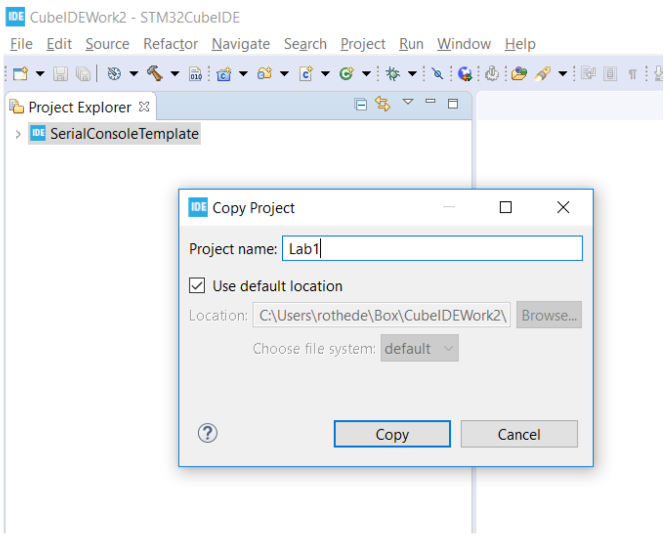
Bytes: 2

Rows: 16

Cols: 80

Char Count:84 CPS:0 Port: 3 19200

9. Now, instead of going through this configuration for every project, simply clone this project. You can do this in CubeIDE by copying-and-pasting the project in the Project Explorer. You can give the new copy a different name in the process. **You will want to delete the Debug subdirectory in the new project along with any Launch files** if it copies over from your template. If you do not do this last step, you may not be running the latest version of your code when you think you are.



Project name: Lab1

[✓] Use default location

Location: C:\Users\rothede\Box\CubeIDEWork2\

Choose file system: default

10. One final note – you may see that your program may execute more than once when running or debugging via CubeIDE. When transferring your code to the board, the board is reset and then CubeIDE takes control and transfers the new executable. In between the time the board is reset and CubeIDE takes control, the previous version of your code, the version that is currently on the board, will run and you may see output from it. You can generally ignore this.

4 Activities

4.1 Create Lab1Hello Project

- Copy the SerialConsoleTemplate Project within CubeIDE in the ProjectExplorer tab by copy-and-pasting.
- Name your project Lab1Hello

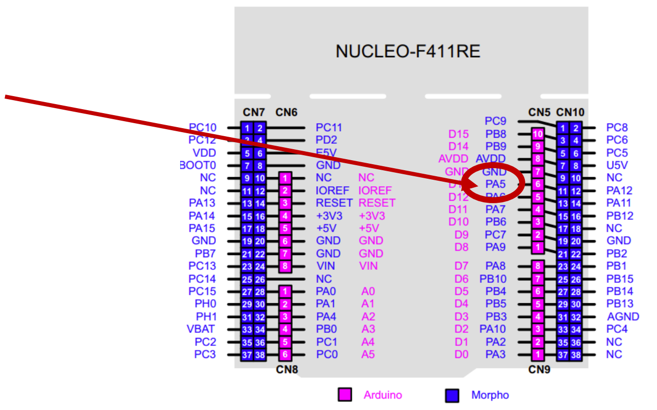
4.2 Add helper APIs

- Add flash.h and flash.c to your project.

3.3 Create a Delay API (delay.h, delay.c)

- Use the analog discovery calculate an approximate delay called N times with the following delay calls:
    - BusyMicro(uint32_t N);
    - BusyMilli(uint32_t N);
    - BusySec(uint32_t N);
- Note: You can attach to the PA5 in Pin6 of CN9 or Pin11 of CN10.

Figure 19. NUCLEO-F411RE



3.4 Write Embedded Hello World.

- Write a program that will:
    - Prompt the user for the number of times to flash the user light.
    - Prompt the user for the length of time in milliseconds the user light should be on/off each iteration.
    - After the light stops blinking the console should prompt the user again to enter another blink count.

- Run your program while attached to the Analog Discovery and show how close you can get to 0.5 seconds with the light on/off. Capture a screen shot for your deliverables.

4. Deliverables

- Demo of your working design.
- Print out of your documented driver code: main.c, delay.h, and delay.c.
- Print out of Analog discover output showing how close to 1/2 second intervals you achieved.
- Staple all deliverables with the distributed rubric.
- All code should be documented and well formatted.
- Print with line numbers and filename information.
- All code should include a header block with:
    - Your name
    - Course number and section
    - Assignment name
    - File name
    - List of any dependencies.