

실험 실습 자료 머리말

이 pdf 파일은 생능 출판사의 **제어 시스템 공학**(임동진 지음)과 함께 배포되는 추가 내용이다. 이 파일에 포함된 내용은 본 책의 각 장의 마지막에 추가되는 절이며, 실험 실습에 대한 내용을 기술하고 있다.

제어 시스템의 실험 실습은 제어 시스템 공학을 배우는 학생들의 이해를 돕는 과정에서 매우 중요한 역할을 할 수 있다. 특히, 많은 학생들이 제어 시스템 공학에 관한 과목을 수강한 후에도 제어 시스템 이론이 실제로 어떻게 구현되고 적용되는지에 대한 개념이 부족한 경우가 많다. 따라서, 제어 공학의 이론 학습과 함께 제어 시스템의 실험 실습은 학생들의 교육에 큰 도움이 될 수 있다.

제어 시스템의 실험을 위해서는 실습 장비가 필요하며, 교육 기기를 제조 판매하는 회사에서 여러 가지 종류의 실습 장비를 판매하고 있다. 이런 실습 장비들은 대량 제조를 하지 않으므로 비교적 고가이며, 또한, 실습 장비와 함께 제공되는 전용 소프트웨어를 사용하도록 하고 있다. 이런 종류의 장비들은 지속적인 업그레이드와 관리가 필요하며, 이로 인해서 초기 도입 비용과 함께 유지 보수 비용도 비교적 높은 편이다. 따라서 제어 시스템의 실험을 지속 가능하게 유지하는 것은 쉽지 않다.

이 책에서 실시하는 실습에 사용되는 실습 장비는 어떤 특정 업체의 장비에 구속되지 않고, 현재 시중에서 쉽게 구입할 수 있는 부품들을 이용해서 어렵지 않게 구성이 가능하다. 또한 동일한 부품을 사용하지 않는 경우에도 이 책의 실습 내용을 응용해서 적용하는 것도 가능하다. 이 실습에서는 학생들이 C 언어를 이용해서 프로그램을 하게 되므로 다소간의 진입 장벽이 있을 수 있지만, 전기, 전자 공학을 전공하는 대부분의 학생들은 저학년에 필수적으로 C 언어를 학습하는 경우가 많으므로 C 언어의 사용이 큰 장벽이라고 생각되지 않는다. 또한, 부수적으로, 이러한 실습을 통해서 학생들은 프로그래밍 언어에 대한 응용 능력을 향상할 수 있게 되고 실무 능력 향상에 도움이 될 수 있다.

실습에 대한 문의 사항이 있을 경우 저자 이메일(limdj@hanyang.ac.kr)로 문의할 수 있다.

실험 실습 목차

1. 제어 시스템의 개념

1.5 실험 실습: Lab 1

1.4.1 디지털 제어용 마이크로컨트롤러

1.4.2 첫 프로젝트: Hello World

1.4.3 타이머 인터럽트와 D/A 변환

2. 제어 시스템의 기초 수학

2.7 실험 실습: Lab 2

2.7.1 A/D 변환

2.7.2 1차 동적 시스템의 실시간 시뮬레이션

3. 제어 시스템의 모델링

3.9 실험 실습: Lab 3

3.9.1 모터 제어를 위한 엔코더

3.9.2 DC 모터 모델의 계수 측정

4. 제어 시스템의 성능

4.7 실험 실습: Lab 4

4.7.1 2차 동적 시스템의 실시간 시뮬레이션

4.7.2 아날로그 다이내믹 시뮬레이터의 피드백 제어

5. 근 궤적 법

5.5 실험 실습: Lab 5

5.5.1 아날로그 다이내믹 시뮬레이터에 대한 PD 제어기

5.5.2 DC 모터의 위치 제어기

6. 주파수 응답을 이용한 해석법

6.8 실험 실습: Lab 6

7. 주파수 영역에서의 제어 시스템 설계

7.7 실험 실습: Lab 7

7.7.1 아날로그 다이내믹 시뮬레이터에 대한 진상 제어기

7.7.2 자기 부상 장치에 대한 진상 제어기

8. 상태 변수 방정식을 이용한 제어 시스템 해석

8.8 실험 실습: Lab 8

9. 상태 변수 방정식을 이용한 제어 시스템 설계

9.5 실험 실습: Lab 9

9.5.1 아날로그 다이내믹 시뮬레이터에 대한 상태 변수 피드백 제어기

9.5.2 아날로그 다이내믹 시뮬레이터에 대한 상태 변수 추정기 기반 제어기

9.5.3 자기 부상 장치에 대한 상태 변수 추정기 기반 제어기

1. 제어 시스템의 개념

1.4 실험 실습: Lab 1

1.4.1 디지털 제어용 마이크로컨트롤러

디지털 제어 시스템은 마이크로컨트롤러를 이용해서 구성할 수 있다. 이 책의 실습에서는 디지털 제어 시스템을 구성하기 위해서 아래 그림과 같은 STMicroelectronics사의 STM32F419 Discovery 보드를 사용한다. Discovery 보드는 STMicroelectronics사에서 자사의 마이크로컨트롤러를 시험해 볼 수 있는 시험용 보드이며 비교적 저렴한 가격에 판매하고 있다. 특히 이와 같은 마이크로컨트롤러는 프로그래밍을 위해서 디버깅 장비가 필요한 경우가 많이 있지만, 디스커버리 보드는 이러한 디버깅 장치가 내장되어 있으므로 별도의 디버깅 장치가 필요하지 않고 개발용 컴퓨터만 준비되면 즉시프로그래밍이 가능하다.



그림 1-13 STM32F429 Discovery 보드

이 보드에는 STM32F429ZIT6 마이크로컨트롤러가 장착되어 있으며, 이 마이크로컨트롤러는 2MB의 플래시 메모리와 256KB의 SRAM을 가지고 있다. 이 마이크로컨트롤러는 RISC 구조 기반의 Arm Cortex-M4 코어를 가지고 있다. Arm Cortex-M4 코어는 부동 소수점(floating point) 연산 명령어를 실행할 수 있어서 복잡한 수식의 신호 처리 연산이나 제어 알고리즘 연산에 적합하다. 또한, 다양한 주변 장치를 갖추고 있으며, 특히 3개의 A/D 컨버터와 2개

의 D/A 컨버터를 가지고 있다. 디지털 제어 시스템을 구성할 때 정확한 타이밍의 구현은 중요하다. 이 마이크로컨트롤러에는 11개의 타이머가 있으며, 그 중에 일부는 인코더 신호를 읽는 기능을 가지고 있다. 이 책에서는 Cortex-M 마이크로컨트롤러의 내부 구조에 대한 상세한 내용을 다루지 않고, 마이크로컨트롤러를 이용한 제어 알고리즘의 구현에 대한 내용을 다룬다.

1.4.2 첫 프로젝트: Hello World

실습을 위해서 가장 먼저 준비해야 하는 것은 컴퓨터(PC, Personal Computer)와 STM32F429 Discovery 보드이다. 일반적으로, 마이크로소프트의 윈도우가 설치된 컴퓨터가 많이 사용되지만, 다른 운영 체제를 사용하는 컴퓨터의 사용도 가능할 수 있다. 실습용 보드는 “STM32F429I-DISC1”으로 검색하거나, 다음과 같은 링크의 쇼핑몰에서 구매가 가능하다.

<https://www.eleparts.co.kr>

<https://www.devicemart.co.kr/>

<https://www.digikey.kr/>

<https://www.mouser.kr/>

다음으로, 실습용 보드를 프로그래밍 하기 위해서, STMicroelectronics에서 배포하는 STM32CubeIDE 소프트웨어를 다운로드 받은 후 컴퓨터에 설치한다. 아래는 다운로드를 위한 링크이며, 다운로드를 위해서 회원 가입이 필요할 수 있다.

<https://www.st.com/en/development-tools/stm32cubeide.html>

STM32CubeIDE는 편집기, 컴파일러, 링커, 디버거 등이 모두 포함된 통합 프로그램이며, 각종 디바이스의 설정에 필요한 소스 코드를 자동 생성하는 기능을 가지고 있다. 일반적으로 마이크로컨트롤러를 프로그래밍 하기 위해서는 많은 레지스터들의 설정이 필요하며, 이를 위해서는 마이크로컨트롤러의 기능을 파악하고 숙지하는 과정이 필요하다. 그러나, STM32CubeIDE를 사용할 경우, 라이브러리의 사용에 의해서 어렵지 않게 마이크로컨트롤러의 프로그래밍이 가능하므로 많은 시간을 절약할 수 있는 장점이 있다. STMicroelectronics에서는 이러한 용도로 사용되는 HAL 라이브러리를 제공하며, 이 책에서는 HAL 라이브러리를 이용한 코드 예제를 제공한다.

실습용 보드의 프로그래밍 환경에 익숙해 지기 위해서, 가장 먼저 “Hello World” 메시지를 화면에 프린트하는 프로젝트를 실행해 보기로 한다. 이를 위해서 컴퓨터에 터미널 프로그램의 설치가 필요하다. 터미널 프로그램은 여러 가지가 있으며, 이 책에서는 SysProgs의 SmarTTY를 사용하기로 한다. SmarTTY는 검색을 통하거나 아래 링크를 통해서 다운로드 받을 수 있다.

<https://sysprogs.com/SmarTTY/>

STM32F429 Discovery 보드에는 미니 USB 커넥터와 마이크로 USB 커넥터가 있으며, 이 실습에서는 미니 USB 커넥터만 사용한다. 실습용 보드의 미니 USB 커넥터는 3가지 기능을 제공한다. 먼저, 실습 보드에 전원을 공급 하며, USB-to-serial 컨버터의 기능을 통해서 컴퓨터의 USB 포트와 통신하는 기능을 제공한다. 또한, 실습용 보드에 내장하고 있는 디버거(ST-Link)와 연결하는 기능을 제공한다. 컴퓨터와 실습용 보드의 미니 USB 커넥터를 케이블로 연결한 후, SmarTTY 프로그램을 열면 다음 그림과 같은 화면을 볼 수 있다. 아래의 그림에서 COM 포트의 번호는 사용하는 컴퓨터의 구성에 따라서 다를 수 있다. 또한, 사용하는 컴퓨터에서 사용하는 주변 장치에 따라서 다른 COM 포트 장치가 나타날 수 있다.

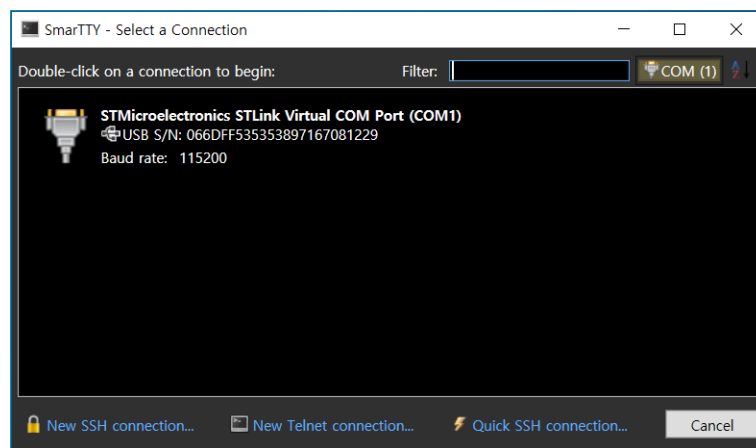


그림 1-14. SmarTTY 터미널 프로그램의 시작 화면

위의 그림의 아이콘을 더블 클릭하면 다음과 같이 터미널이 열리면서 터미널 화면을 볼 수 있다. 이제, 컴퓨터와 실습용 보드는 시리얼 포트를 통해서 통신할 수 있는 준비가 된 상태이다.

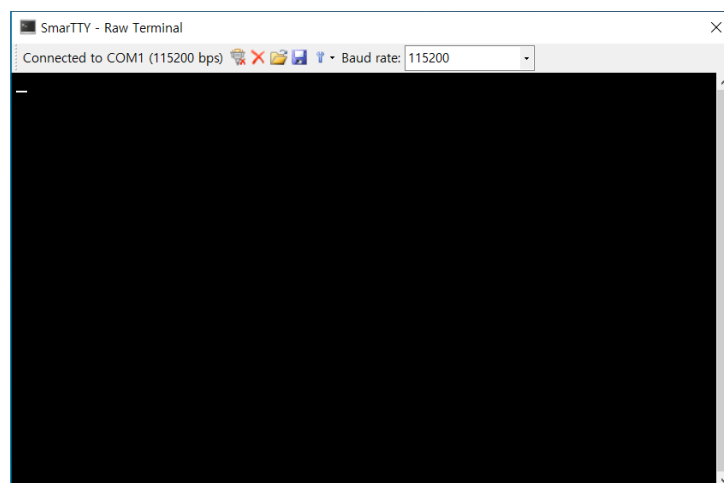


그림 1-15. 시리얼 터미널 화면

이제 Discovery 보드를 프로그램하기 위해서, STM32CubeIDE 프로그램을 시작하면 아래 그림과 같은 시작 화면을 볼 수 있다. 아래의 설명과 같이 프로젝트를 진행하는 과정에 필요한 파일들의 다운로드가 필요하게 되는데, 이를 위해서 자신의 계정에 로그인을 요구한다. 따라서, 아래에 설명한 단계를 진행하기 전에 **자신의 계정에 로그인** 할 필요가 있다. STM32CubeIDE 프로그램에서 자신의 계정에 로그인하는 방법은 이 파일의 뒷부분에 첨부되어 있다.

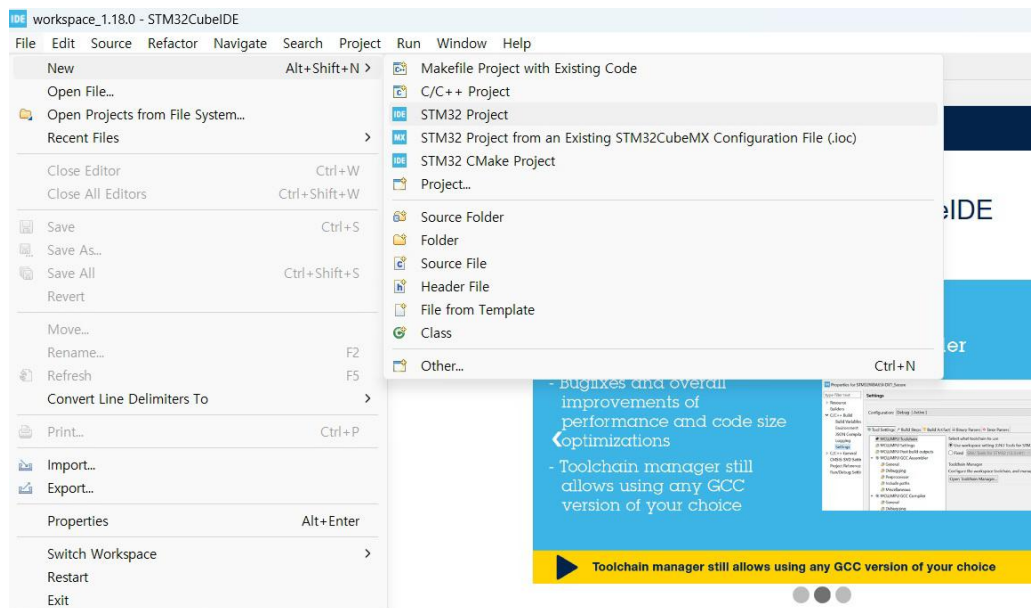


그림 1-16. STM32CubeIDE 시작 화면

프로젝트를 생성하기 위해서, 위의 화면에서, File 메뉴에서 New를 선택한 후 STM32 Project 항목을 선택하면, 다음과 같은 보드 선택 화면을 볼 수 있다.

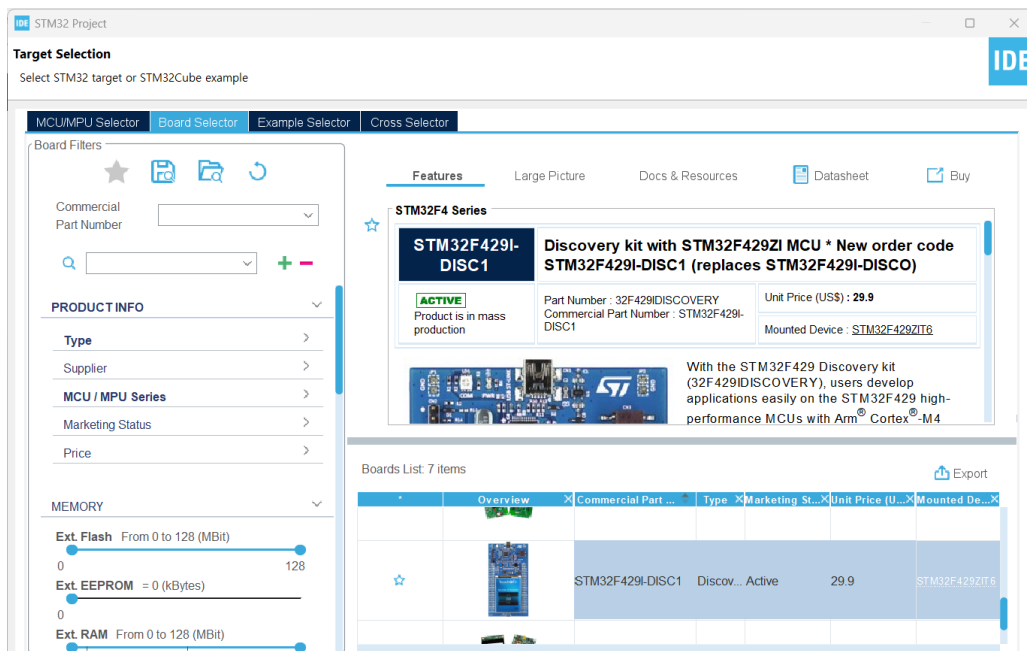


그림 1-17. 보드 선택 화면

위의 화면에서 Board Selector 탭을 선택한 후, STM32F429I-DISC1 보드를 선택한다. 보드의 종류가 많으므로 Type은 Discovery Kit, MCU/MPU Series는 STM32F4를 선택하면 목록이 짧아지므로 보드를 찾는 것이 쉬워진다. 다음 NEXT 버튼을 클릭하면, 다음과 같은 프로젝트 설정 화면을 볼 수 있다.

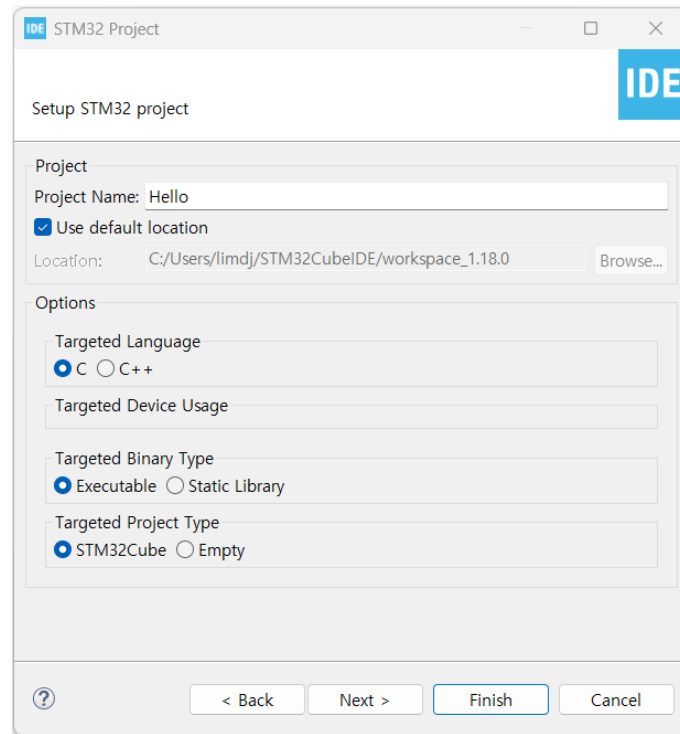


그림 1-18. 프로젝트 설정 화면

프로젝트의 이름을 입력한 후 Finish 버튼을 클릭한다. 이때, 이 화면에서 프로젝트 파일들이 저장되는 폴더의 위치를 기억할 필요가 있다. 또한, 이를 변경하기를 원할 경우 원하는 폴더에 저장하는 것도 가능하다. 위의 화면에서 Finish를 클릭하지 않고 Next를 클릭할 경우 보드 라이브러리의 버전을 선택할 수 있는 화면이 나오며, 이 화면에서 라이브러리의 버전을 선택할 수 있다. Finish 버튼을 클릭할 경우 최신 버전의 라이브러리가 자동으로 선택되며, 최신 버전의 라이브러리가 설치되어 있지 않을 경우에는 최신 버전의 라이브러리의 다운로드와 설치가 자동으로 진행될 수 있다.

다음으로, 다음과 같이 보드 초기화에 대한 설정을 물어보는 화면이 나오면 Yes 버튼을 클릭한다. 보드의 소프트웨어의 초기화 설정을 기본 설정으로 선택한다는 의미이다.

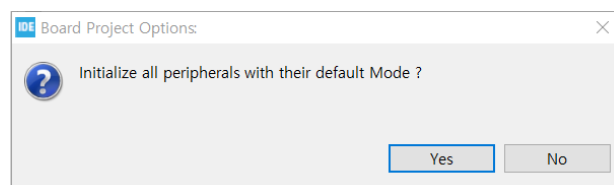


그림 1-19. 보드 초기화 설정 화면

보드 초기화 설정 화면 이후에도 사용하는 프로그램의 버전에 따라서 선택하는 화면이 추가

로 나올 수 있으며, 대체로 본 프로젝트와 크게 상관이 없으므로 YES를 선택해서 넘어간다.

위의 과정 이후에 필요한 파일의 다운로드가 진행될 수 있다. 프로그램 설치 후 처음으로 프로젝트를 시작할 때 다운로드하는 파일들이 많으므로 다운로드가 종료되기 까지 다소 오래 걸릴 수 있다. 파일의 다운로드는 대체로 처음으로 실행할 때 진행되며, 이후에는 업데이트가 있을 경우에만 다운로드가 실행된다. 다운로드의 진행 과정에 동의하는 화면이 나올 경우에는 동의를 체크하고 진행한다. 사용하는 컴퓨터에 바이러스 방지 프로그램이 설치되어 있는 경우, **바이러스 실시간 감시 기능을 끈 상태에서 진행**하는 것을 권고한다. 바이러스 방지 프로그램의 종류에 따라서 STM32CubeIDE 프로그램을 바이러스 프로그램으로 인식해서 정상적인 작동을 방해하는 경우가 있을 수 있다. 바이러스 방지 프로그램으로 인해서 정상적인 다운로드가 진행되지 않을 경우의 해결 방법에 대한 설명은 Lab1 파일의 마지막에 첨부한다.

위의 과정이 모두 끝나면, 다음 그림과 같은 프로젝트 화면을 볼 수 있다. 이때, 화면 좌측의 Project Explorer 창에서 새로 생긴 프로젝트를 확인한다.

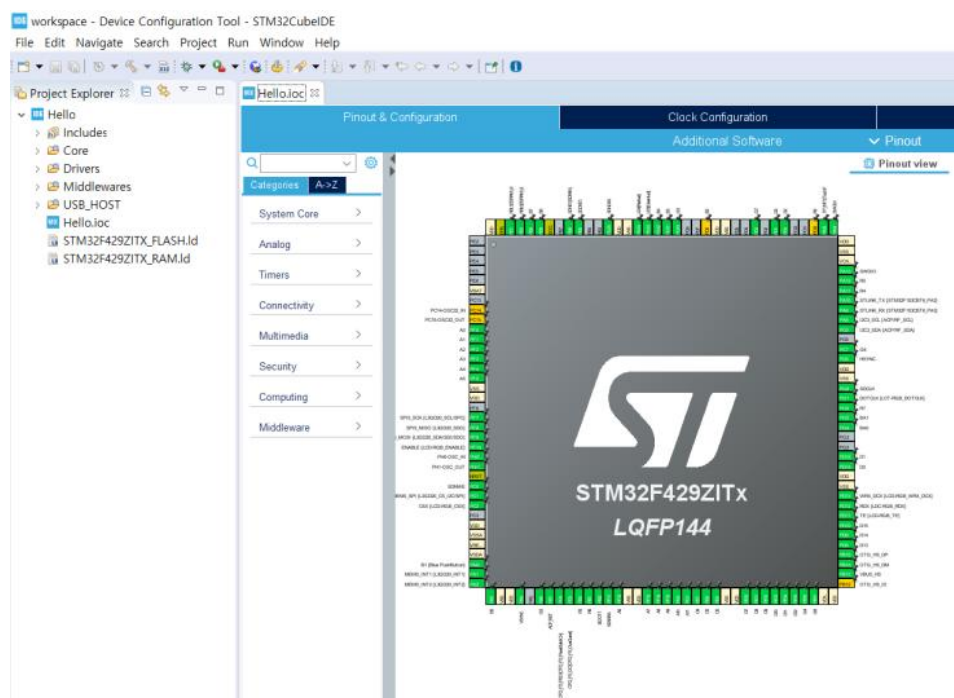


그림 1-20 Project 화면

보드의 각종 디바이스에 대한 초기화는 기본 모드를 선택 했으므로, 설정에 대한 변경은 필요하지 않다. 그러나, 필요한 주변 장치들에 대한 설정이 제대로 이루어 졌는지 확인해 보는 것도 필요할 수 있다. Pinout & Configuration 탭에서 Connectivity 항목을 클릭하면 아래의 그림과 같이 통신 주변 장치의 리스트를 볼 수 있다. 이중에서 USART1이 컴퓨터와 통신을

하는 시리얼 포트이다. USART1 버튼을 클릭하면 시리얼 포트의 설정을 볼 수 있다. 기본 모드 설정에서 USART1은 이미 선택되어 있으므로 설정을 바꿀 필요가 없으며 확인만 한다.

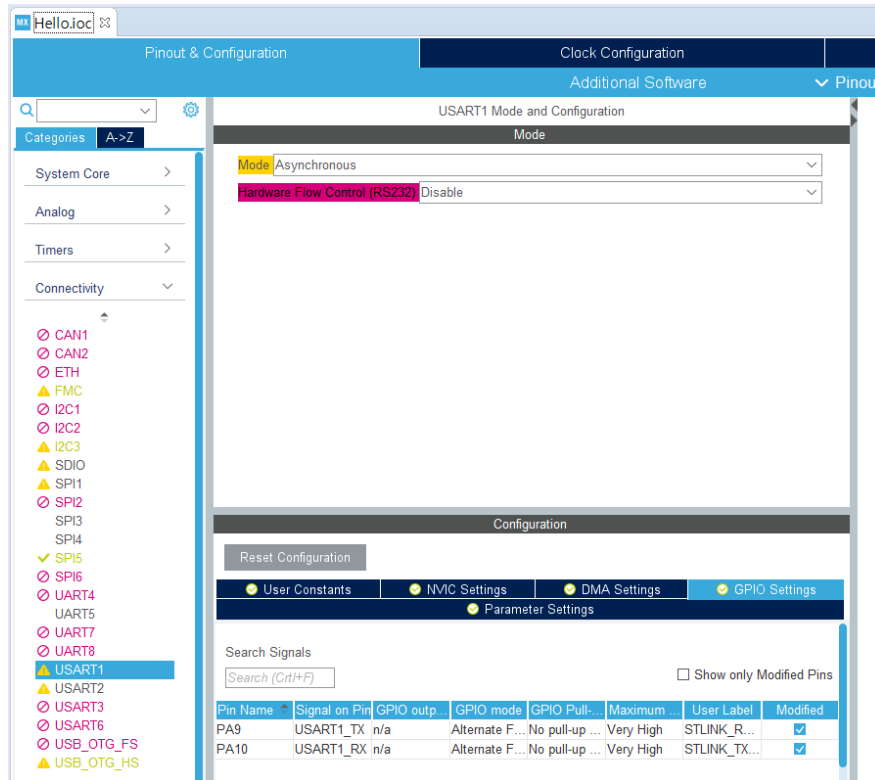


그림 1-21 USART 설정

코드 생성을 진행하기 전에 FREERTOS를 비활성화(disable) 할 필요가 있다. FREERTOS는 아래 그림과 같이 Middleware 항목을 클릭해서 나오는 목록에서 FREERTOS를 클릭한 후, Disable을 선택하면 비활성화 된다. 물론, 이 책에서 실행하는 모든 프로젝트들은 FREERTOS를 이용해서 수행할 수 있으나, 프로젝트들을 가급적 간단하게 실행할 수 있도록 FREERTOS는 사용하지 않기로 한다.

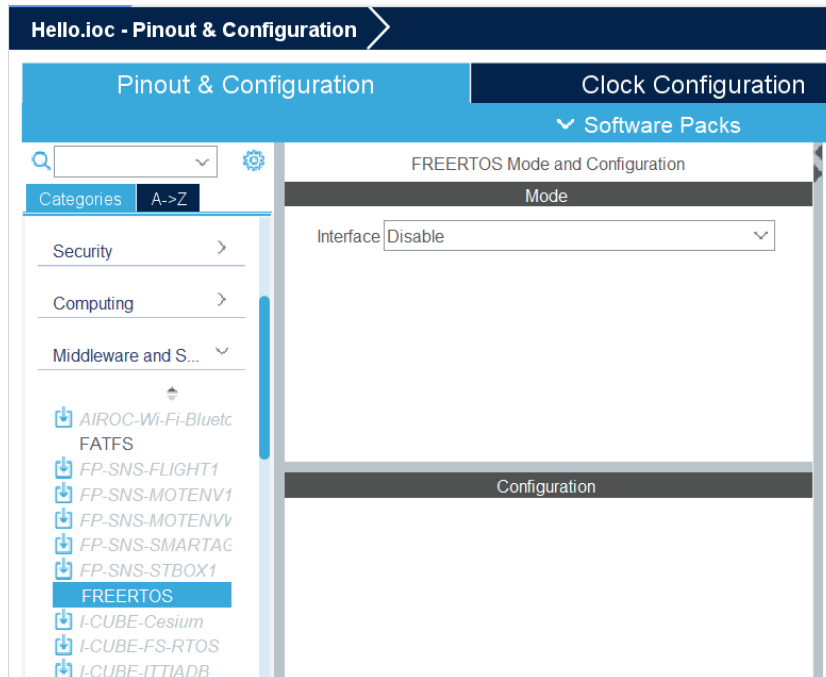


그림 1-22 FREERTOS 비활성화

아래 그림과 같이 Project 메뉴에서 Generate Code를 선택하면 코드 생성이 진행된다.

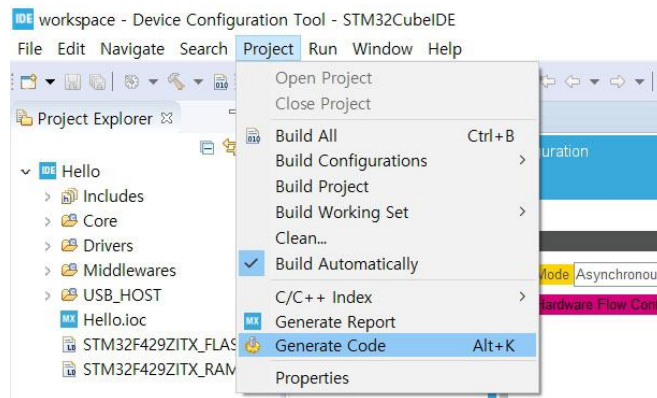


그림 1-23 코드 생성

코드 생성이 완료된 후, Project Explorer에서 생성된 코드 파일의 이름을 더블 클릭하면 파일을 열 수 있다. 아래의 그림은 main.c를 더블 클릭해서 열린 화면이다.

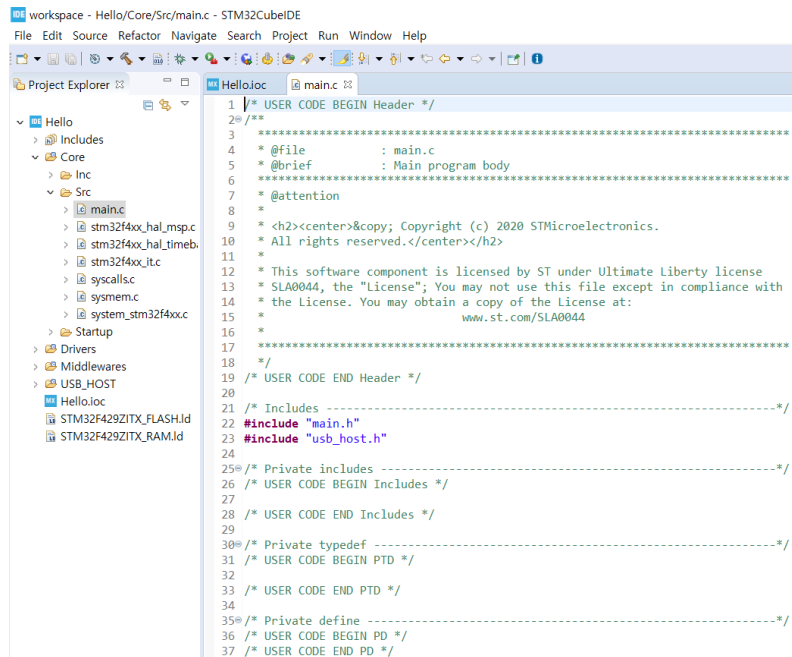


그림 1-24 main.c 파일

위의 화면에서 열린 main.c에는 주변 기기를 설정하는 코드 이외에는 아무 동작도 하지 않는다. 필요한 동작을 하기 위해서는 main.c의 파일에 필요한 코드를 입력해야 한다. 생성된 소스 코드 파일을 보면 /* USER CODE BEGIN ... */ 과 /* USER CODE END ... */의 코멘트 문이 있는 것을 볼 수 있다. 생성된 소스 코드 파일의 /* USER CODE BEGIN ... */ 과 /* USER CODE END ... */의 코멘트 문 사이에는 사용자가 코드를 입력할 수 있는 부분이다. 이후에 코드 생성을 다시 실행했을 때, 이 영역의 코드는 사용자 코드로 인식해서 삭제하지 않는다. 만약, 이와 같은 사용자 코드 영역 이외의 영역에 코드를 입력한 후 코드 생성을 다시 실행하면 입력한 코드는 삭제되므로 주의가 필요하다.

임베디드 시스템 프로그래밍에서 printf의 기능은 매우 유용하다. 일반적으로 컴퓨터에서 C 언어 프로그램을 작성할 때 printf는 콘솔 화면에 프린트 하는 기능이다. 일반 컴퓨터에서 콘솔 화면은 디스플레이 화면에 나타나며, 사용자가 이 함수의 기능의 구현 코드를 작성할 필요가 없다. 그러나, 마이크로컨트롤러의 프로그램을 할 때 printf 기능을 사용하기 위해서는 이 함수를 구현하는 코드가 있어야 한다. 아래에 주어진 코드는 시리얼 포트를 이용한 printf 문을 구현하는 함수이다. 아래에 주어진 코드를 main.c 소스 파일에 추가한다. 반드시 /* USER CODE BEGIN ... */ 과 /* USER CODE END ... */의 코멘트 문 사이에 입력한다. 코드 1.2는 printf 함수의 구현 함수이다. 이렇게 구현된 printf 문을 사용할 때는 모든 초기화가 끝난 이후에 사용해야 한다.

소스 파일 사용에 관한 주의: PDF 파일에서 복사해서 편집기에 붙이기를 할 경우 정상적인

복사와 붙이기(copy and paste)가 되지 않을 수 있다. 따라서 배포된 소스 파일의 main.c에서 아래의 코드를 찾아서 복사한 후 붙이기를 한다. 배포된 main.c에서 아래에 주어진 코드를 제외한 나머지 부분의 코드는 자동 생성된 코드로서 사용하는 프로그램의 버전에 따라 달라질 수 있다. 따라서 main.c 파일 전체를 그대로 사용할 경우 정상적으로 실행되지 않을 수 있으므로 아래에 주어진 코드만 복사를 해서 편집기에서 붙이기를 한다. 앞으로 진행되는 모든 실습에서 사용하는 소스 코드에 대해서도 동일하게 적용이 된다.

코드 1.1

```
/* USER CODE BEGIN Includes */
#include "stdio.h"
/* USER CODE END Includes */
```

코드 1.2

```
/* USER CODE BEGIN 0 */
#ifdef __GNUC__
#define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
#else
#define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
#endif /* __GNUC__ */
PUTCHAR_PROTOTYPE
{
    HAL_UART_Transmit(&huart1, (uint8_t *)&ch, 1, 0xFFFF);
    return ch;
}
/* USER CODE END 0 */
```

코드 1.3

```
/* USER CODE BEGIN 2 */
printf("Hello World\r\n");
/* USER CODE END 2 */
```

소스 코드 입력이 완료되면 파일을 저장한 후, 아래 그림과 같이 Project 메뉴의 Build Project 항목을 선택해서 빌드를 실행한다.

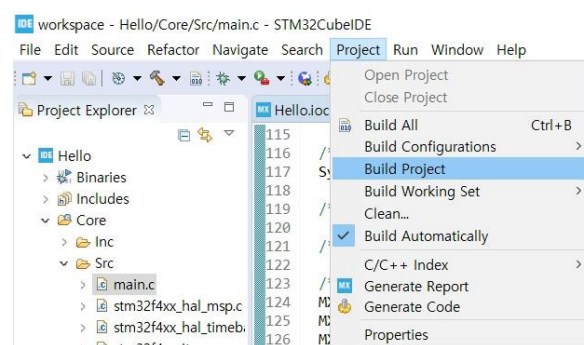


그림 1-25 프로젝트의 빌드

빌드 작업이 종료된 후, 아래 그림과 같이 오류 없이 완료된 것을 확인한다.

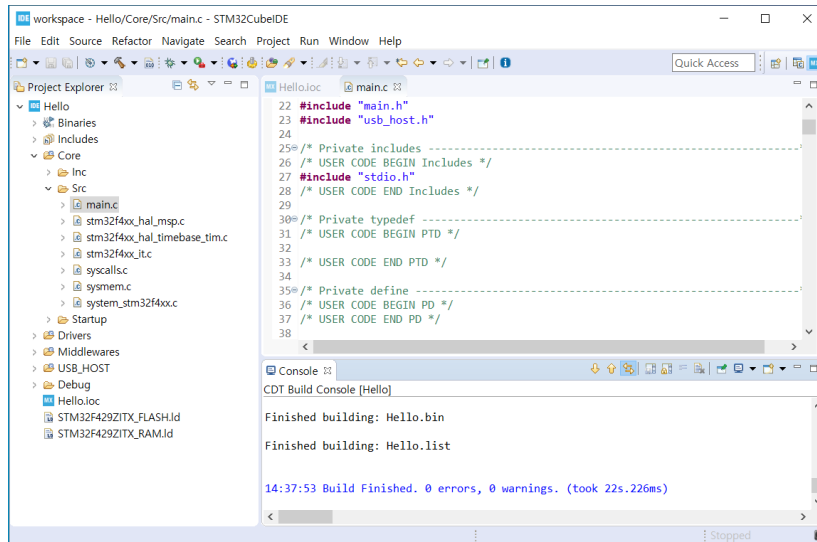


그림 1-26 빌드 작업 종료

빌드가 성공적으로 완료되면, Discovery 보드에서 실행할 준비가 된 것이다. 다음 그림과 같이 Run 메뉴에서 Debug를 선택한다.

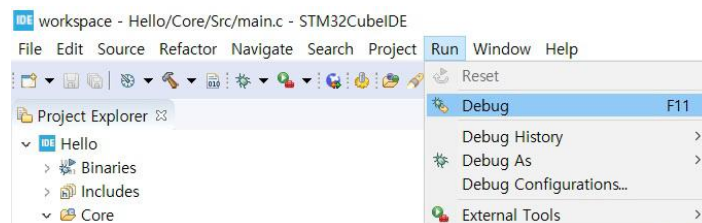


그림 1-27 디버깅 시작

다음과 같은 옵션 선택 화면이 나오면 OK 버튼을 클릭한다.

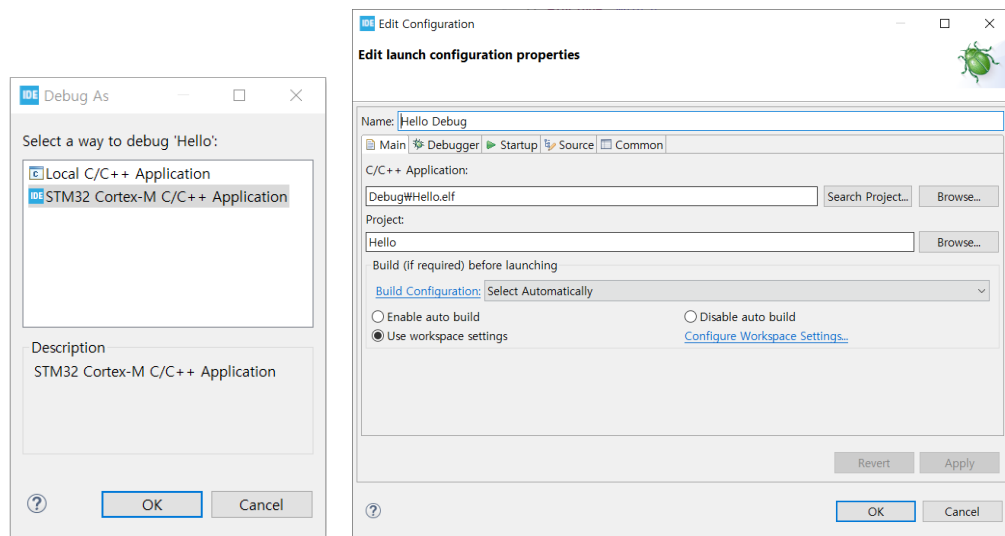


그림 1-28 디버깅 옵션

실행 파일의 다운로드가 완료되는 것을 기다린다. 화면의 윈도우 아래쪽의 상태 바(status bar)에서 진행 상태를 볼 수 있다. “Download verified successfully”라는 메시지가 나올 때까지 기다린후, 다운로드가 완료되면 다음과 같은 디버깅 화면이 나온다.

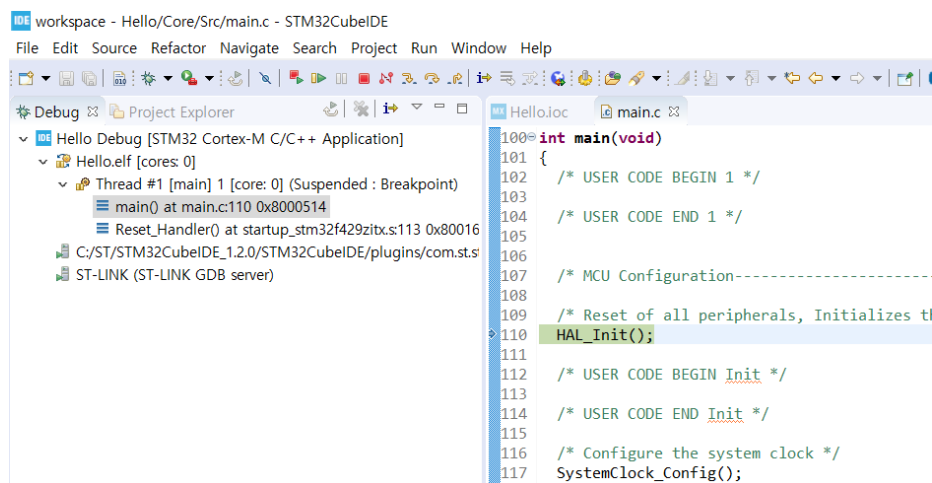


그림 1-29 디버깅 창

아직 프로그램이 실행되는 상태는 아니며 대기 상태이다. 프로그램을 실행하기 위해서는 메뉴 바의 녹색의 “Go” 버튼을 클릭한다. 다음, 아래 그림과 같이 “Go” 버튼이 회색으로 바뀌면서 프로그램이 실행된다. 실행 상태에서 프로그램을 중단하기 위해서는 빨간색 사각형의 정지 버튼을 클릭한다.

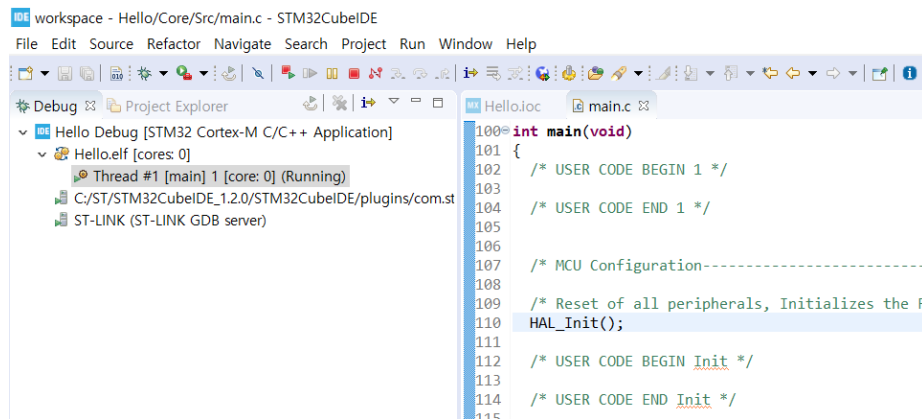


그림 1-30 디버깅 창

터미널 창에 아래와 같이 “Hello World” 메시지를 볼 수 있으면 성공적으로 실행된 것이다.

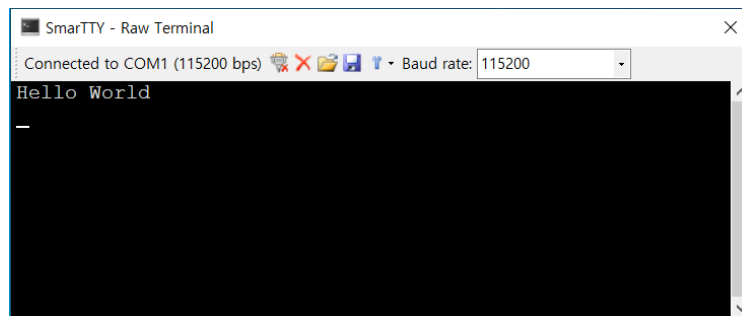


그림 1-31 시리얼 터미널 창의 “Hello World” 메시지

1.4.3 타이머 인터럽트와 D/A 변환

디지털 제어 시스템에서 제어 신호는 정확한 순간에 출력이 되어야 한다. 이를 위해서 제어 신호 발생의 간격을 정확하게 제어할 수 있는 타이밍 장치가 필요하다. 일반적으로, 디지털 제어 시스템의 제어 신호 발생 간격은 일정한 값으로 정해져 있다.

인터럽트는 마이크로컨트롤러를 이용한 실시간 디지털 제어 시스템에서 가장 중요한 기능이다. 인터럽트는 즉각적인 대응이 필요한 이벤트의 발생을 프로세서에 알리는 신호이다. 마이크로컨트롤러의 인터럽트에는 여러 가지 종류가 있으며, 각 인터럽트 종류마다 인터럽트 서비스 루틴(interrupt service routine) 혹은 인터럽트 핸들러(interrupt handler)라고 부르는 코드가 배정되어 있다. 인터럽트 서비스 루틴은 인터럽트가 발생했을 때 실행되는 일종의 함수이다. 인터럽트의 발생이 감지되면 프로세서는 현재 실행 중인 코드를 중단하고 인터럽트 서비스 루틴으로 점프한다. 이때, 프로세서는 현재 진행 중인 작업을 중단하게 되므로, 프로세서는 현재의 상태에 관한 데이터를 스택에 저장한 후, 인터럽트 서비스 루틴으로 점프한

다. 인터럽트 서비스 루틴의 실행이 종료되면 프로세서는 스택에 저장된 정보를 복구한 후 중단된 작업을 다시 계속한다. 디지털 제어 시스템에서는 타이머가 일정한 시간 간격으로 인터럽트를 발생하도록 설정한다. 그리고 타이머 인터럽트에 대한 인터럽트 서비스 루틴에서는 제어 신호를 발생하기 위한 계산을 실행한 후 제어 신호를 출력한다. 마이크로컨트롤러의 초기 상태는 인터럽트가 비 활성화 된 상태이므로, 인터럽트를 사용하기 위해서는 이를 위한 설정과 인터럽트 서비스 루틴이 준비되어야 한다.

타이머이외에, 디지털 제어 시스템에 필요한 중요한 장치로 D/A 컨버터가 있다. 제어 시스템을 구성할 때, 제어 대상 시스템의 입력 신호가 아날로그 신호인 경우가 있을 수 있다. 이러한 경우, 디지털 제어 시스템의 제어기는 디지털 시스템이므로 제어 알고리즘에 따라서 디지털 값으로 계산된 제어 신호를 아날로그 신호로 바꿀 필요가 있다. 이 책에서 실습에 사용하는 STM32F429 Discovery 보드의 마이크로컨트롤러에는 2개의 D/A 컨버터가 내장되어 있다. 이 D/A 컨버터의 레졸루션(resolution)은 12비트이며, $2^{12} = 4096$ 개의 전압 레벨을 나타낼 수 있다. 이 보드에 인가되는 DC 전압은 3볼트이므로, D/A 컨버터가 출력할 수 있는 최소 전압 단계의 값은 다음과 같다. 즉, 0 볼트에서 3볼트 사이의 전압을 4096개의 디지털 숫자로 나타낸다.

$$\frac{3}{2^{12}} = 0.0007324 \text{ Volt} \quad (1.2)$$

16진수로 나타냈을 때, D/A 컨버터의 데이터 값은 0x000에서 0xFFFF까지이다. 이 숫자를 10진수로 나타내면, 0에서 4095까지의 숫자이다. D/A 컨버터에 0의 값을 쓰면 0볼트가 출력된다. 최대값인 16진수 0xFFFF 혹은 십진수 4095를 D/A 컨버터에 쓰면 출력 전압의 값은 다음과 같다. 이 최대 값이 3볼트에 약간 못 미치는 값이지만, 그 차이가 크지 않으므로 문제가 되는 경우는 거의 없다.

$$4095 \times \frac{3}{2^{12}} = 4095 \times \frac{3}{4096} = 2.99927 \quad (1.3)$$

이 프로젝트에서는 타이머 TIM10을 이용해서 1 msec의 시간 간격으로 인터럽트를 발생시키고, 타이머 인터럽트가 발생할 때 마다 D/A 컨버터에서 전압을 출력해서 타이머 인터럽트의 발생 시간 간격을 확인해 본다.

앞의 예에서 설명한 방법으로 새로운 프로젝트를 시작하고 프로젝트 이름을 Timer10으로 정한다. 아래 그림과 같이 한 개의 워크스페이스(workspace)내에 두개의 프로젝트가 생성된 것을 확인할 수 있다. 이와 같이 한 개 이상의 프로젝트가 있을 경우, 빌드와 실행할 때 올바른 프로젝트가 선택되도록 주의한다. 프로젝트 이름을 클릭하면 프로젝트가 선택되고, 선택된 프로젝트 이름의 배경 색이 바뀐다. 아래 그림의 Project Explorer에서 볼 수 있듯이 선

택된 프로젝트 이름 Timer10의 배경 색이 바뀐 것을 볼 수 있다.

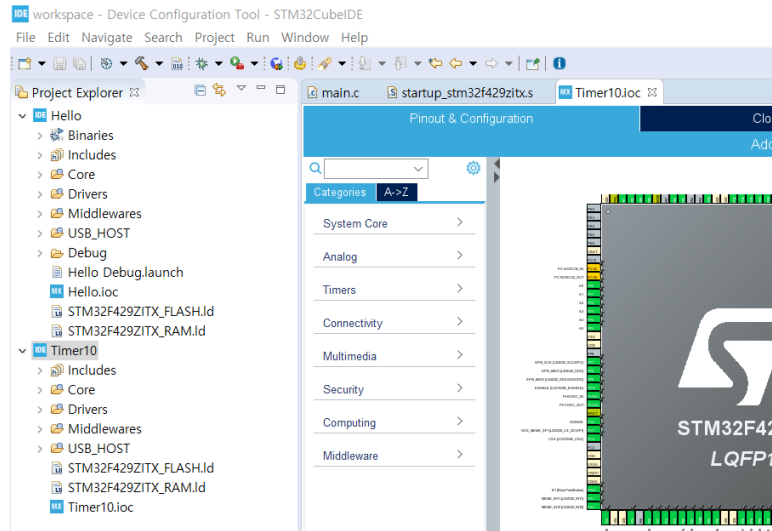


그림 1-32 Timer10 프로젝트

타이머에 대한 설정을 진행하기 전에 클럭에 대한 설정이 필요하다. Clock Configuration 탭을 클릭하면 다음 그림과 같은 클럭 설정 화면이 나온다. 이 화면에서 HCLK (MHz)라고 표시된 창을 찾아서 168을 입력하고 enter 키를 누르면 다음과 같이 클럭의 값이 변경된다.

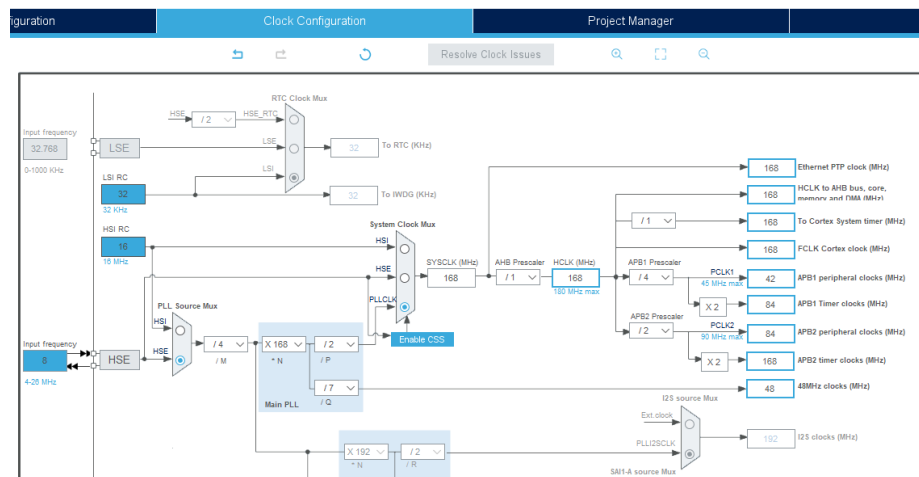


그림 1-33 클럭 설정

클럭에 대한 설정이 완료된 후, Pinout & Configuration 탭을 클릭한다. 다음으로, Timers 버튼을 클릭하면 타이머들의 목록이 나오며, 여기에서 TIM10을 선택한다. TIM10의 설정 창에서 Activated 옆의 작은 박스를 클릭하면 체크로 바뀌며, Prescaler 창에는 839, Interval Clock Division 창에는 199를 입력한다. 다음 그림과 같이 설정된 것을 확인한다. 타이머 TIM10은

168MHz ABP2 타이머 클럭을 사용하며, 위와 같이 설정하면 TIM10은 다음 식과 같이 1 msec의 인터벌을 가지게 된다.

$$\frac{168 \text{ MHz}}{840 \times 200} = 1000 \text{ Hz} \tag{1.4}$$

모든 레지스터들은 0에서 숫자를 시작하므로 840 대신에 839를, 200 대신에 199를 입력한 것에 주목한다.

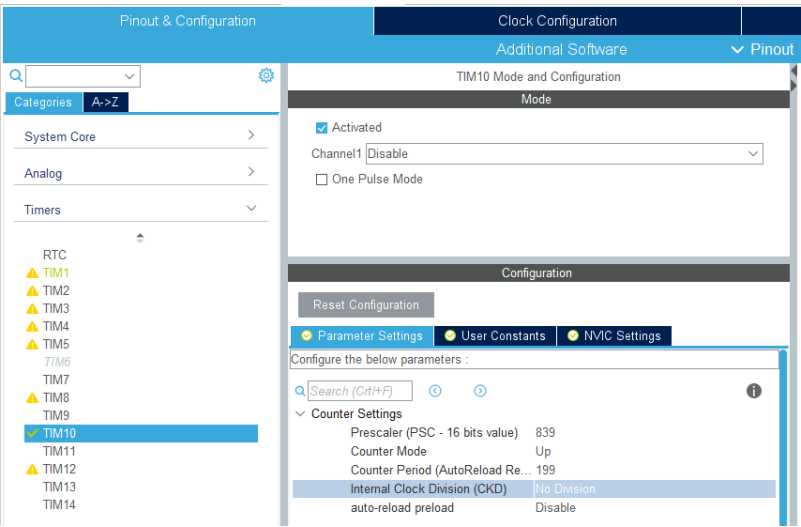


Figure 1-34 TIM10 configuration

타이머 인터벌에 대한 설정이 완료되면, 타이머의 인터럽트에 대한 설정이 필요하다. 타이머 TIM10의 설정 창에서 다음 그림과 같이 타이머 인터럽트 항목(TIM10 global interrupt)의 작은 상자를 클릭하면 체크로 변경되어 타이머 인터럽트가 활성화 된다.

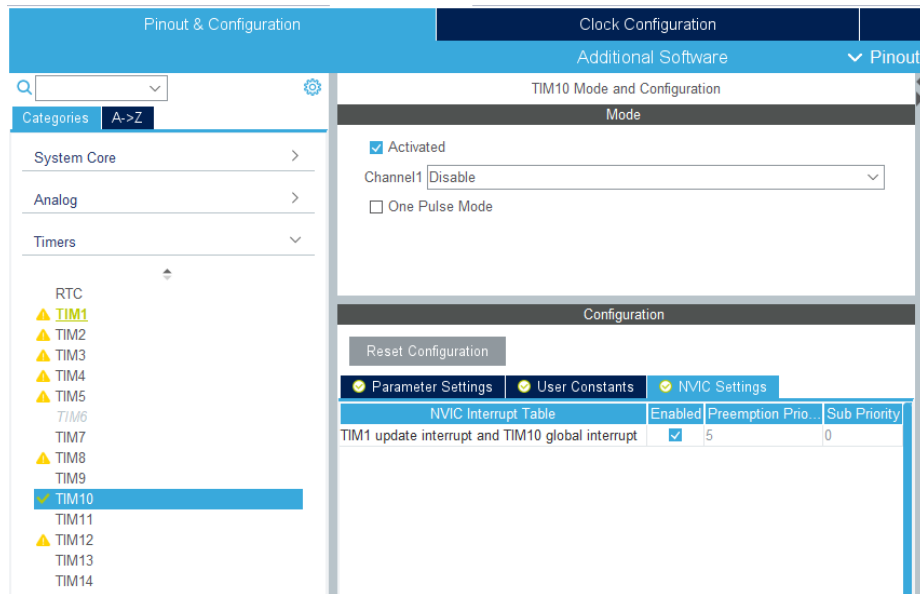


그림 1-35 TIM10 인터럽트 활성화

다음으로, D/A 컨버터를 설정한다. Analog 버튼을 클릭하면 다음 그림과 같이 아날로그 입출력 장치 목록이 나오며, DAC를 선택하면 D/A 컨버터 설정이 나온다. 여기에서 OUT1은 다른 장치가 사용 중이므로, OUT2를 선택한다. 아래 화면에서 볼 수 있듯이, DAC_OUT2 출력은 PA5 핀으로 출력되는 것을 볼 수 있다.

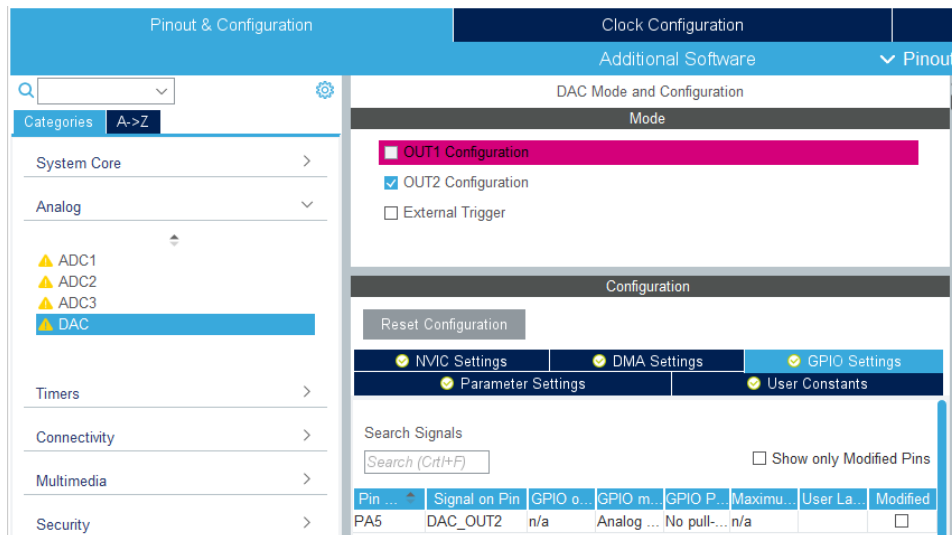


그림 1-36 The D/A 컨버터 설정

앞에서 언급한 바와 같이, 코드 생성을 실행하기 전에 FREERTOS를 비활성화(disable)한다. 이전 예제와 동일한 방법으로 코드 생성을 한 후, main.c 파일을 열어서 아래에 주어진 코드

1.4와 코드 1.5의 소스를 입력한다. 코드 1.4가 입력되는 부분은 모든 주변 장치의 초기화가 실행된 이후이다. 코드 1.4는 타이머와 D/A 컨버터의 동작을 시작하는 함수를 실행하는 코드이다.

코드 1.4

```
/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start_IT(&htim10);
HAL_DAC_Start(&hdac, DAC_CHANNEL_2);
/* USER CODE END 2 */
```

코드 1.5

```
/* USER CODE BEGIN Callback 0 */
uint32_t da_value;
if (htim->Instance == TIM10) {
    da_value=0xffff;
    HAL_DAC_SetValue(&hdac, DAC_CHANNEL_2, DAC_ALIGN_12B_R, (uint32_t)(da_value));
    for (uint32_t i=0;i<1000;i++);
    da_value=0x0;
    HAL_DAC_SetValue(&hdac, DAC_CHANNEL_2, DAC_ALIGN_12B_R, (uint32_t)(da_value));
}
/* USER CODE END Callback 0 */
```

코드 1.5는 main.c 파일의 끝 근처의 HAL_TIM_PeriodElapsedCallback 함수의 내부에 입력된다. HAL_TIM_PeriodElapsedCallback 함수는 타이머에 의한 인터럽트가 발생했을 때 불리는 함수(이러한 함수를 callback 함수라고 부른다.)이다. 즉, 인터럽트 서비스 루틴 내에서 호출되는 함수이다. 코드 1.5는 타이머에 의한 인터럽트가 1 msec 마다 발생했을 때 D/A 컨버터의 최대 전압 값(3볼트)을 출력하고 약간의 딜레이 후에 0볼트를 출력하는 기능을 수행하는 코드이다. 아래의 그림은 이 코드가 실행될 때, D/A 컨버터의 출력을 오실로스코프로 관찰한 화면이다. 예상한 대로, 1 msec 마다 3볼트 크기의 펄스 전압이 출력되는 것을 확인할 수 있다.

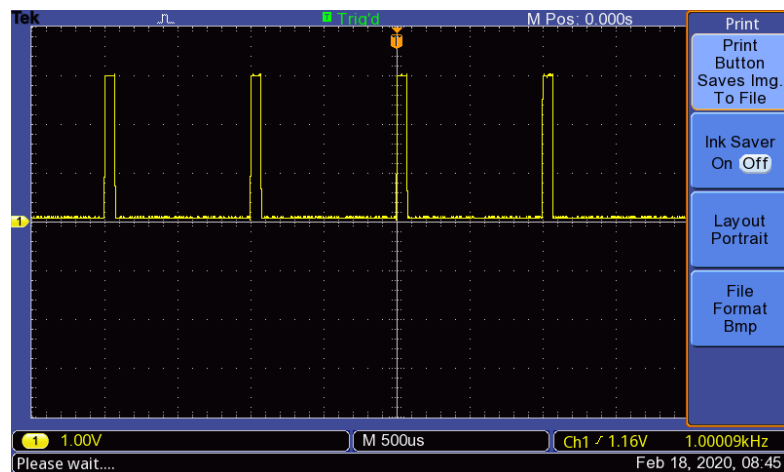


그림 1-37 오실로스코프 화면

실습 연습문제 1.1

아래 그림과 같은 톱니 파형의 전압을 D/A 컨버터에서 출력하는 코드를 앞에 주어진 코드를 수정해서 작성한다. 톱니 파형의 주기는 100 msec이며, 출력 전압의 최소 값은 0볼트, 최대 값은 3볼트이다.

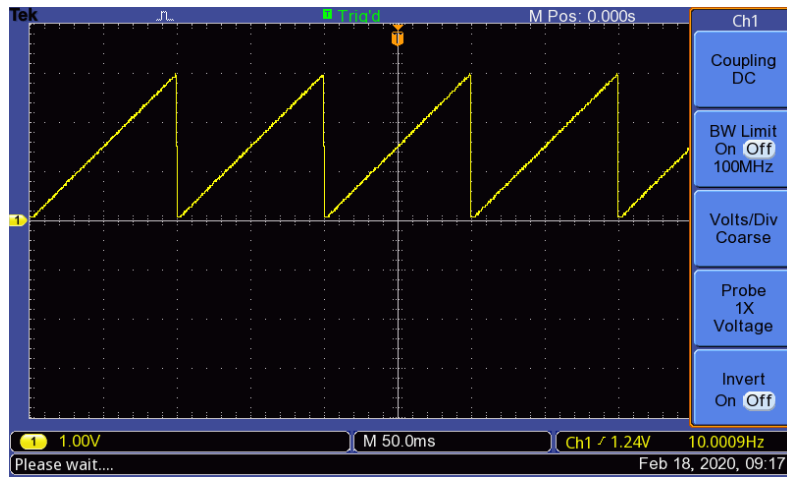
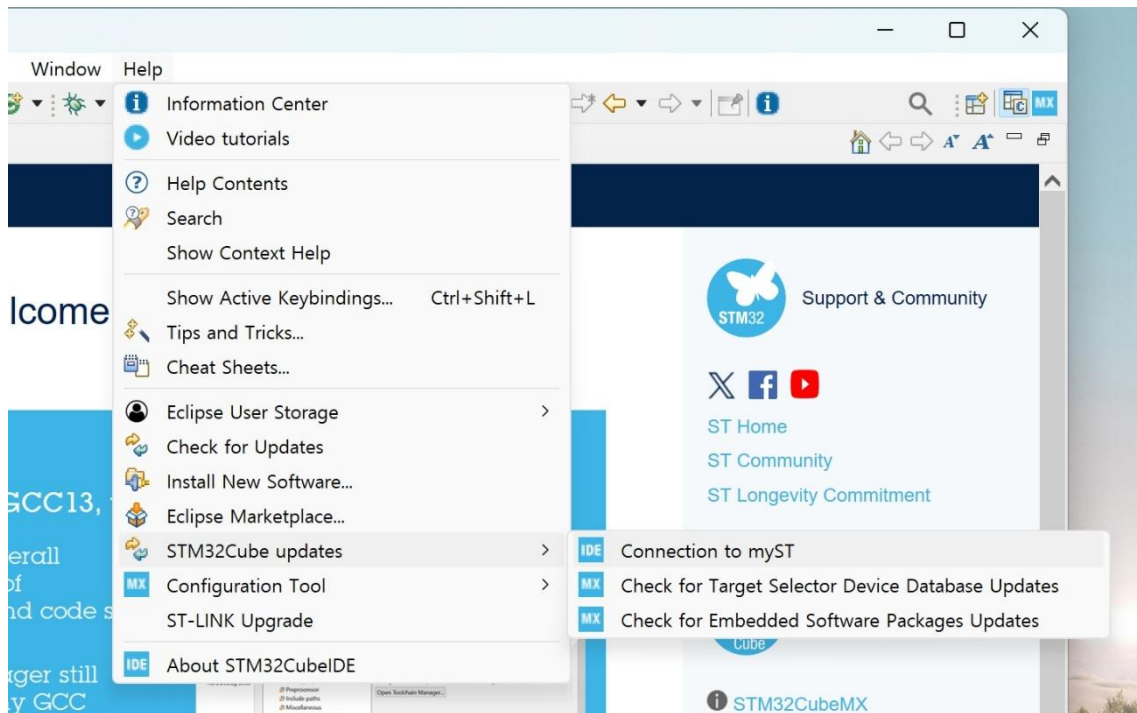


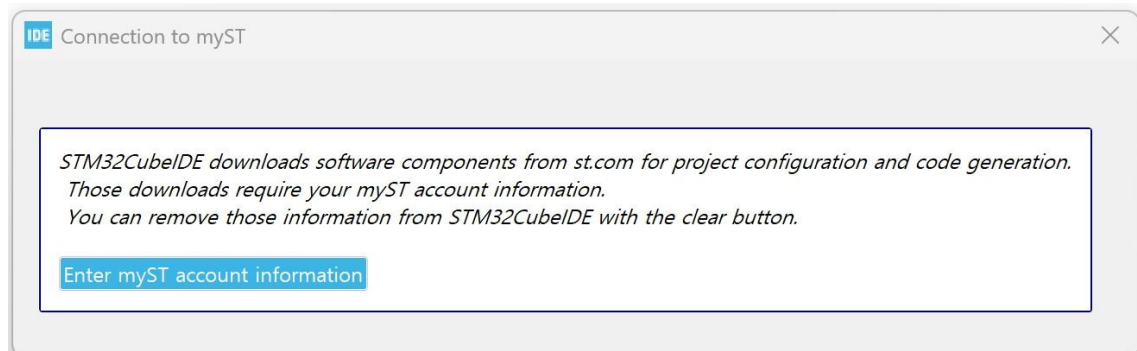
그림 1-38 톱니 파형의 오실로스코프 화면

STM32CubeIDE 에서 자신의 계정에 로그인하는 방법

STM32CubeIDE 를 시작한 후 아래 그림과 같이 Help 메뉴에서 STM32Cube updates 를 선택한 후, Connection to myST 를 선택한다.



다음으로 아래와 같은 안내 화면이 나오면 Enter myST account information 버튼을 클릭한다.



다음으로 아래의 화면에서 등록된 자신의 이메일 주소와 비밀번호를 입력한 후 Login 버튼을 클릭하면 자신의 계정에 로그인 된 상태에서 프로그램을 사용할 수 있게 되며, 필요한 파일들의 다운로드들이 진행될 수 있다.

MX

User Authentication Dialog

×

Already registered?

Enter your e-mail address and password to login your myST user.

E-mail address

Password

☐ Remember me on this device.

i

Login

New user?

myST brings you a set of personalized features:

○ Participate to ST Events

○ Stay informed with ST eNewsletters

○ Get help with ST Online Support

○ Discuss on the ST Community

○ Benefit from our Online Design Tools

○ Download Software

○ Order free samples

○ Manage your weekly product updates

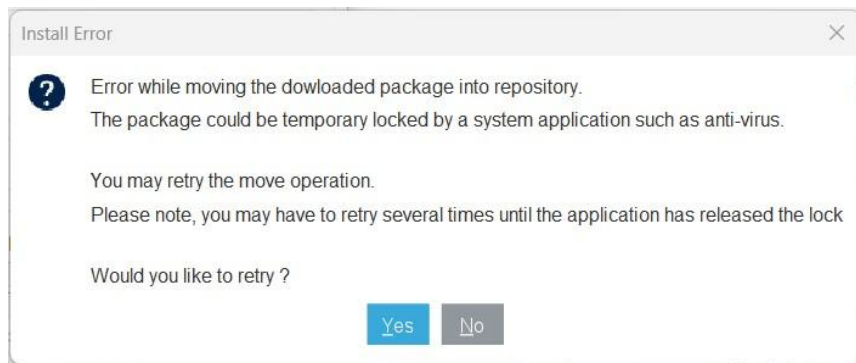
○ Buy ST Products & Tools

Create Account

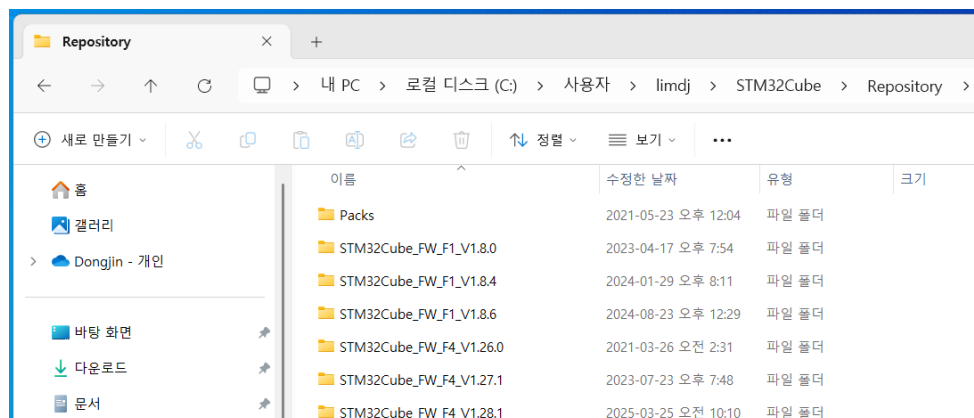
Feedback

참고 사항: 보드 라이브러리 설치가 정상적으로 진행되지 않는 경우의 해결 방법

STM32CubeIDE에서 프로그램 가능한 마이크로컨트롤러와 실습용 보드는 종류가 대단히 많으므로 프로그램을 처음 설치했을 때 필요한 모든 파일들이 설치되지 않는다. 사용자가 보드를 선택하고 프로젝트를 생성했을 때 추가 설치가 필요한 파일들은 자동으로 다운로드되고 설치가 진행된다. 이때 바이러스 방지 프로그램의 실시간 감시 기능이 동작하고 있을 경우 이러한 자동 설치 과정을 위험 요소로 판단해서 중단시킨다. 아래의 그림은 이와 같은 상황의 오류 메시지 화면이다.

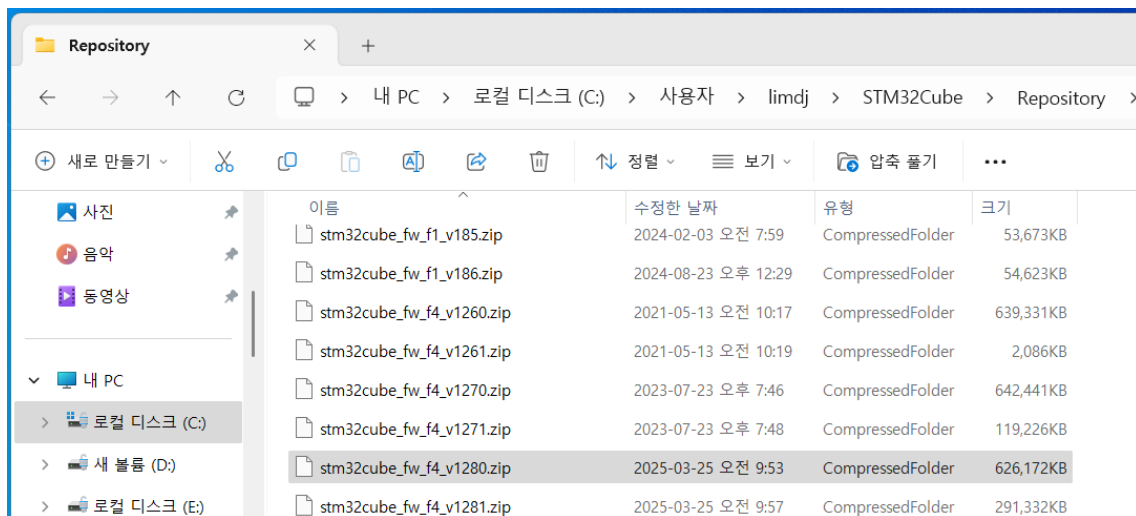


이와 같은 상황에 대해서 알아보기 위해서 라이브러리 파일이 설치되는 폴더를 열어본다. 프로젝트를 생성했을 때 설치가 필요한 파일들은 보드를 사용하기 위한 라이브러리 파일들이다. 이 파일들의 위치는 다음과 같다.

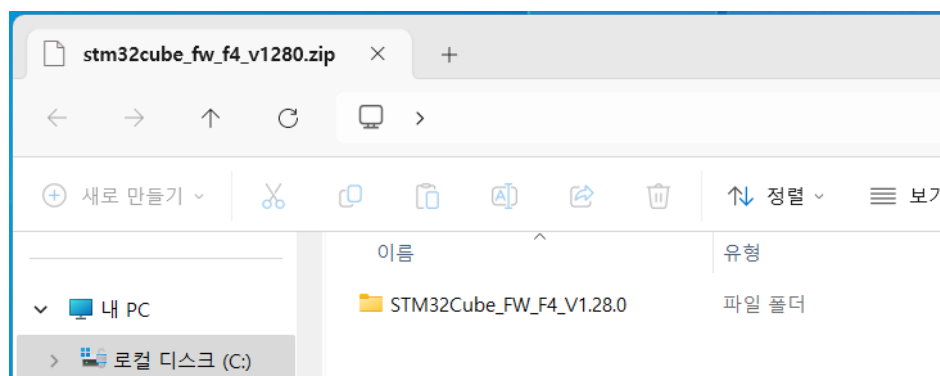


C 드라이브의 사용자 폴더에서 사용자 이름(사용하는 컴퓨터의 사용자 이름은 위의 그림과 다를 수 있음)의 폴더 내에 STM32Cube라는 폴더가 있으며, 이 폴더 내의 Repository 폴더에 보드를 위한 라이브러리 파일들이 설치된다. 각 파일들은 위의 그림과 같이 라이브러리의 버전 숫자를 포함하는 이름의 폴더에 설치된다. 설치 과정은 zip 압축 파일을 다운로드

한 후 압축을 풀어서 위의 그림과 같이 Repository 폴더에 복사를 한다. 정상적으로 설치가 진행이 된다면 이 과정은 사용자의 개입없이 자동으로 진행이 되지만, 바이러스 방지 프로그램은 이와 같은 과정에 위험 요소가 있다고 판단해서 중단 시킨다. 따라서 설치가 정상 진행되지 않은 경우에 Repository 폴더를 열어보면 압축 파일의 다운로드를 정상적으로 진행되어 저장된 것을 볼 수 있다. 따라서 이런 경우에는 압축 파일을 풀어서 압축 파일내의 위의 그림의 폴더 이름과 유사한 이름을 가지는 폴더를 복사한다. 예를 들어서 아래의 그림은 Repository 폴더에 저장된 압축 파일들이다.



위의 파일들 중에서 stm32cube_fw_f4_v1280.zip 파일을 열어보면 아래 그림과 같이 이름이 STM32Cube_FW_F4_V1.28.0인 폴더가 포함된 것을 볼 수 있다. 이 폴더가 Repository 폴더에 저장되어야 하는 폴더이다. 압축 파일을 풀어서 아래 그림과 같은 이름의 폴더를 Repository 폴더에 복사를 하면 설치가 완료된다. 파일의 개수가 많으므로 압축을 풀고 저장하는데 시간이 오래 걸릴 수 있다.



윈도우의 파일 탐색기에는 압축을 푸는 기능이 내장되어 있지만 압축을 푸는 과정이 오래 걸리는 등의 불편함이 있을 수 있다. 이때 사용할 수 있는 압축 프로그램으로 7-Zip 프로그램

램이 있다. 7-Zip으로 검색해서 찾거나, 다음의 링크에서 다운로드 받을 수 있다.

<https://www.7-zip.org/>

7-Zip 프로그램이 설치된 상태에서 압축 파일을 선택한 후, 마우스의 오른쪽 버튼을 클릭하면 나오는 아래 그림과 같은 메뉴에서 7-Zip을 선택해서 **여기에 압축 풀기**를 선택하면 압축 풀기가 진행된다.

