

ROBOTIS CM9 공개소스활용하기

ROBOTIS CM-9은 H/W와 S/W가 모두 공개된 오픈 소스입니다. 모든 소스는 아래의 gitHub 사이트를 통해서 접근할 수 있습니다. 공개된 소스를 가지고 자신만의 로봇 개발 환경을 구축할 수 있고 전세계 모든 사람들과 공유할 수 있습니다.

여기서는 윈도우 OS를 기본으로 설명하지만 linux와 Mac OS X에도 활용할 수 있도록 이클립스를 이용한 개발 방법을 소개합니다.

1. ROBOTIS CM9 공개 소스 위치

https://github.com/robotis-pandora/ROBOTIS_CM9_Series.git

ROBOTIS_CM9_Series / +

Change title name from "ROBOTIS" to "ROBOTIS CM9"		
robotis-pandora authored 2 hours ago		
Firmware	3 days ago	fix file name in makefile [robotis-pandora]
cm-9_ide	2 hours ago	Change title name from "ROBOTIS" to "ROBOTIS CM9" [robotis-pandora]
Notice_en.txt	4 days ago	First commit [robotis-pandora]
Notice_ko.txt	4 days ago	First commit [robotis-pandora]
README.md	4 days ago	Update README.md [robotis-pandora]

- ① Firmware : bootloader와 ROBOTIS CM9 IDE에서 사용되는 Core-library로 구성되어 있습니다. ROBOTIS CM9에서 사용되는 Wiring API는 모두 Core-library를 기반으로 하고 우리가 CM-9 IDE에서 컴파일하고 다운로드 되는 펌웨어는 모두 core-library_0x08003000 디렉토리에 포함되어 있습니다.




bootloader는 flash memory의 0x08000000의 위치에서 시작되고 IDE에서 다운로드 되는 사용자 펌웨어는 0x08003000에서 시작됩니다.

ROBOTIS_CM9_Series / Firmware / +

fix file name in makefile		
robotis-pandora authored 3 days ago		
..		
bootloader_0x08000000	4 days ago	First commit [robotis-pandora]
core-library_0x08003000	3 days ago	fix file name in makefile [robotis-pandora]

- ② CM-9_ide : ROBOTIS CM9은 아두이노 1.0.1을 기반으로 제작되었으며 아두이노는 Processing을 기반으로 만들어졌기 때문에 Processing-core라는 프로젝트를 참조로 합니다. 실제로 ROBOTIS CM9 IDE가 구현된 부분은 Processing-head가 됩니다. 따라서 향상된 IDE를 개발하기 위해서는 processing-head 부분을 수정해야 합니다.

ROBOTIS_CM9_Series / cm-9_ide /

Change title name from "ROBOTIS" to "ROBOTIS CM9"		
 robotis-pandora authored 2 hours ago		
..		
 processing-core	3 days ago	more cleanup [tician]
 processing-head	2 hours ago	Change title name from "ROBOTIS" to "ROBOTIS CM9"
















2. Eclipse 실행을 위한 Java Development kit를 다운로드하고 설치합니다.

<http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html>

Java SE Development Kit 7u17

You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.

☐ Accept License Agreement
 ☒ Decline License Agreement

Product / File Description	File Size	Download
Linux x86	106.65 MB	 jdk-7u17-linux-i586.rpm
Linux x86	92.97 MB	 jdk-7u17-linux-i586.tar.gz
Linux x64	104.78 MB	 jdk-7u17-linux-x64.rpm
Linux x64	91.71 MB	 jdk-7u17-linux-x64.tar.gz
Mac OS X x64	143.78 MB	 jdk-7u17-macosx-x64.dmg
Solaris x86 (SVR4 package)	135.39 MB	 jdk-7u17-solaris-i586.tar.Z
Solaris x86	91.67 MB	 jdk-7u17-solaris-i586.tar.gz
Solaris SPARC (SVR4 package)	135.92 MB	 jdk-7u17-solaris-sparc.tar.Z
Solaris SPARC	95.32 MB	 jdk-7u17-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	22.97 MB	 jdk-7u17-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	17.59 MB	 jdk-7u17-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	22.61 MB	 jdk-7u17-solaris-x64.tar.Z
Solaris x64	15.02 MB	 jdk-7u17-solaris-x64.tar.gz
Windows x86	88.75 MB	 jdk-7u17-windows-i586.exe
Windows x64	90.42 MB	 jdk-7u17-windows-x64.exe

Accept 버튼을 클릭하고 사용하고 있는 PC의 OS에 맞게 JDK 패키지를 다운로드 받고 설치합니다.

성공적으로 JDK가 설치되었다면 아래와 같이 명령창에서 확인할 수 있습니다.

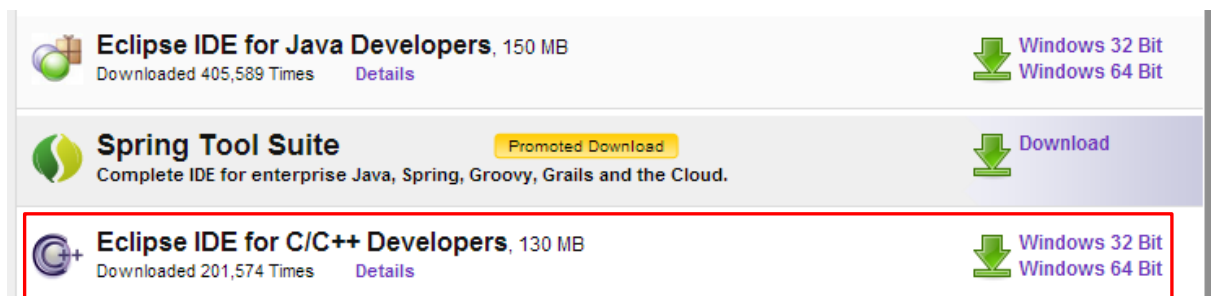
```
C:\Users\win2storm>java -version
java version "1.7.0_17"
Java(TM) SE Runtime Environment (build 1.7.0_17-b02)
Java HotSpot(TM) Client VM (build 23.7-b01, mixed mode, sharing)
```

3. Eclipse를 이용한 공개 소스 받기 및 프로젝트 Import

이클립스의 git plug-in을 이용하면 gitHub에 공개된 공개소스를 쉽게 받을 수 있습니다. 이클립스는 사용되는 언어에 따라 다양한 패키지가 있으므로 Firmware 개발을 위해서는 C/C++ 패키지를 받고 IDE개발을 위해서는 Java 패키지를 받습니다. 호환을 위해서 최신 버전을 받습니다.

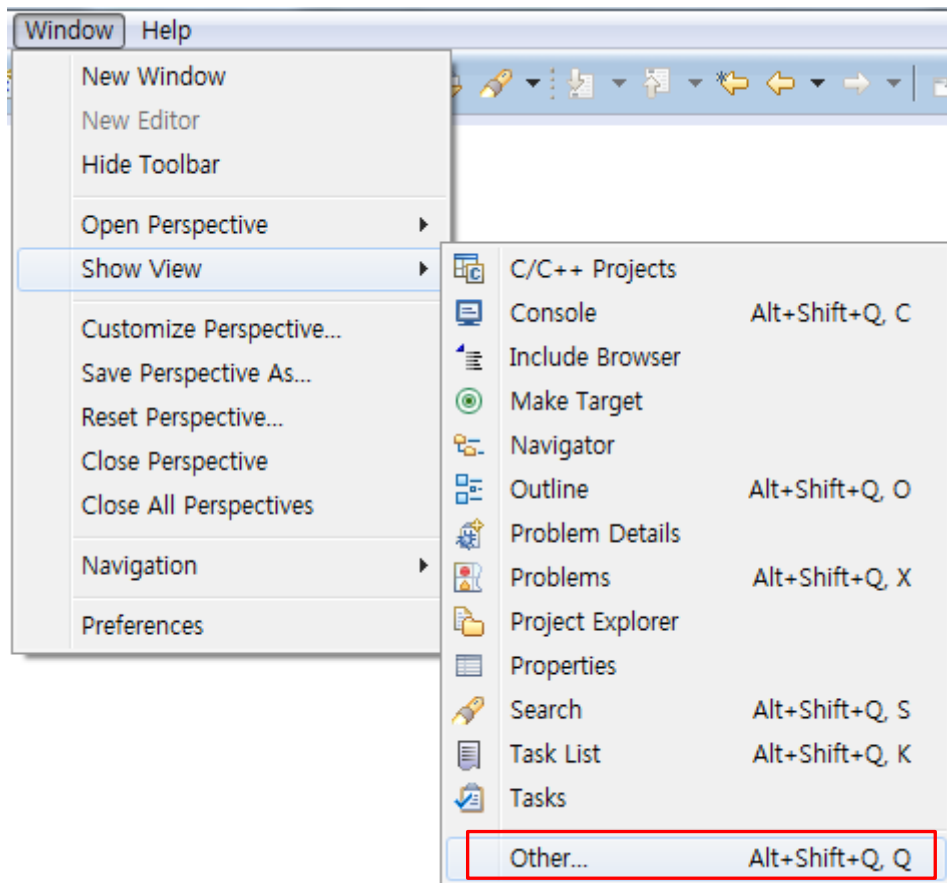
Eclipse download : <http://www.eclipse.org/>

여기서는 C/C++ 패키지를 받습니다.

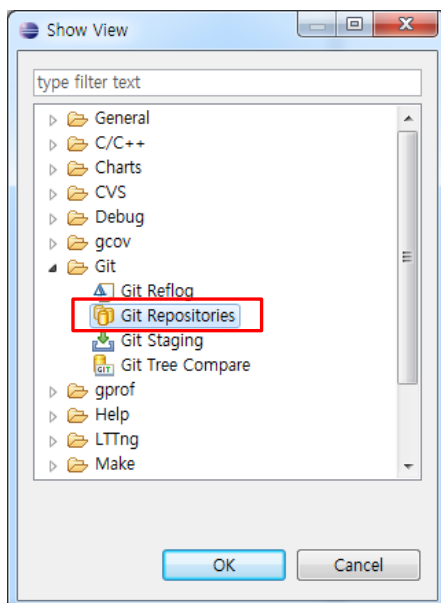


자신의 PC에 설치된 자바 버전에 따라 32bit 또는 64bit를 받습니다. 자바 버전은 커맨드 창에서 `java -version`을 통해 확인합니다. 만약 java 환경이 32bit인데 64bit 패키지를 받아서 실행한다면 실행이 되지 않습니다. 그 반대의 경우도 마찬가지입니다.

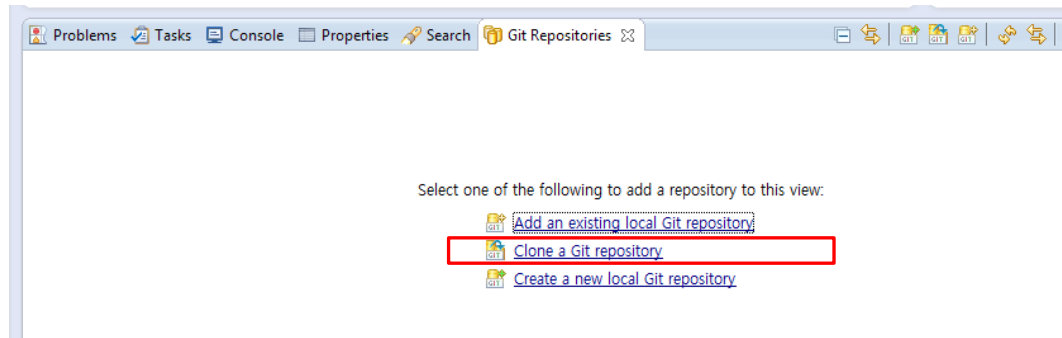
① 이클립스를 실행하고 git plug-in을 실행합니다.



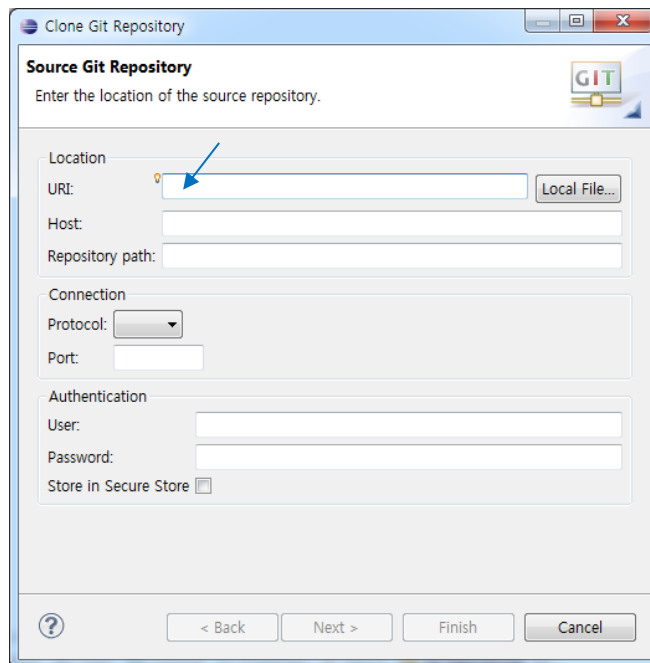
Window -> Show View -> Other...를 선택합니다.



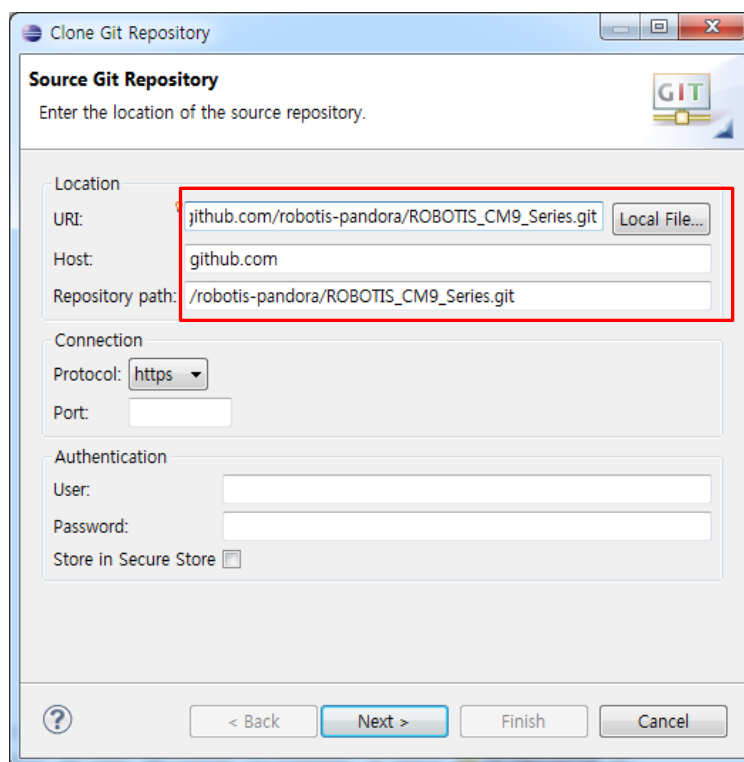
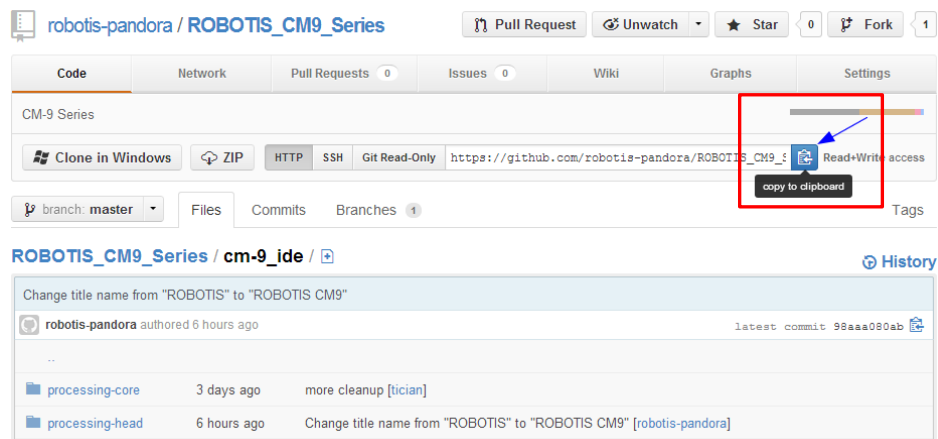
Git Repositories를 선택하면 하단 View 윈도우에 아래와 같이 Git 플러그인이 나타납니다.



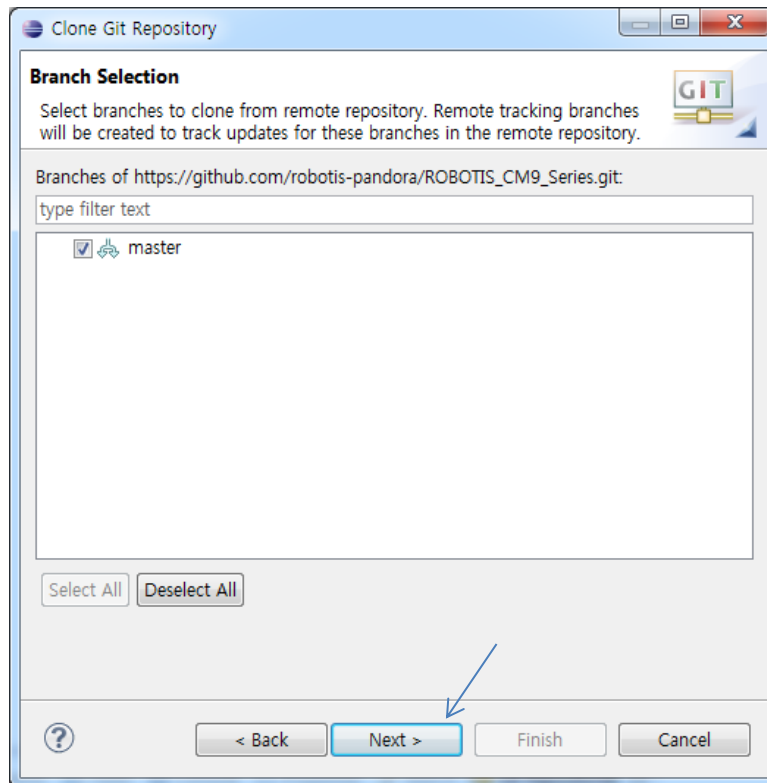
Clone a Git repository를 선택합니다.



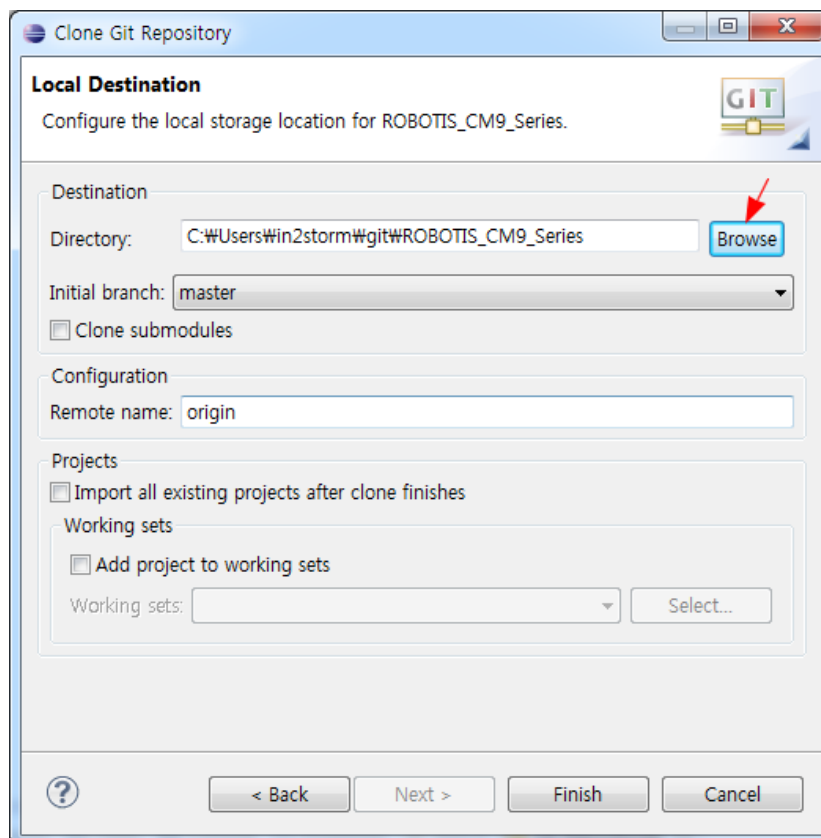
https://github.com/robotis-pandora/ROBOTIS_CM9_Series.git를 URI항목에 복사해서 붙여넣기 합니다. 자동으로 완성되는 부분이 생깁니다. 혹은 아래와 같이 gitHub 사이트에서 ROBOTIS_CM9_Series를 찾으신 후 아래 버튼을 눌러 클립보드에 복사해서 붙여넣기 해도 무방합니다.



그대로 Next를 클릭합니다.

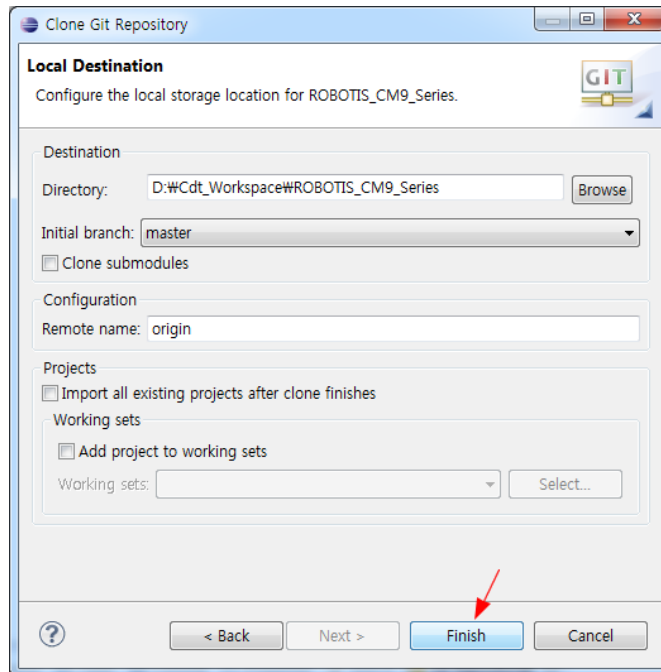


마찬가지로 Next 합니다.

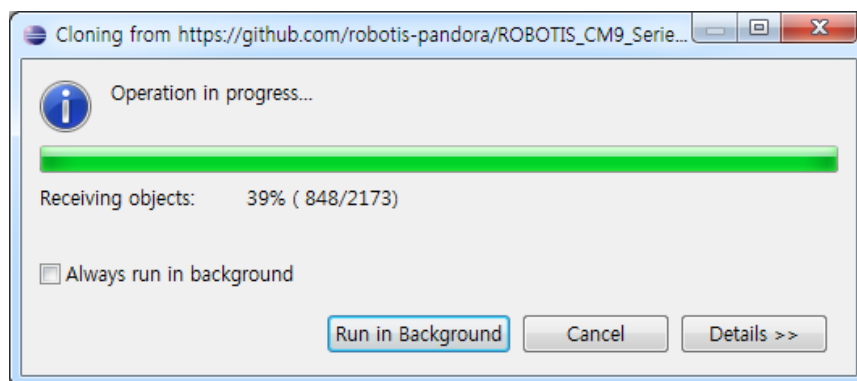


Destination 항목에서 Repository가 저장될 디렉토리를 지정해야 합니다. Browse 버튼

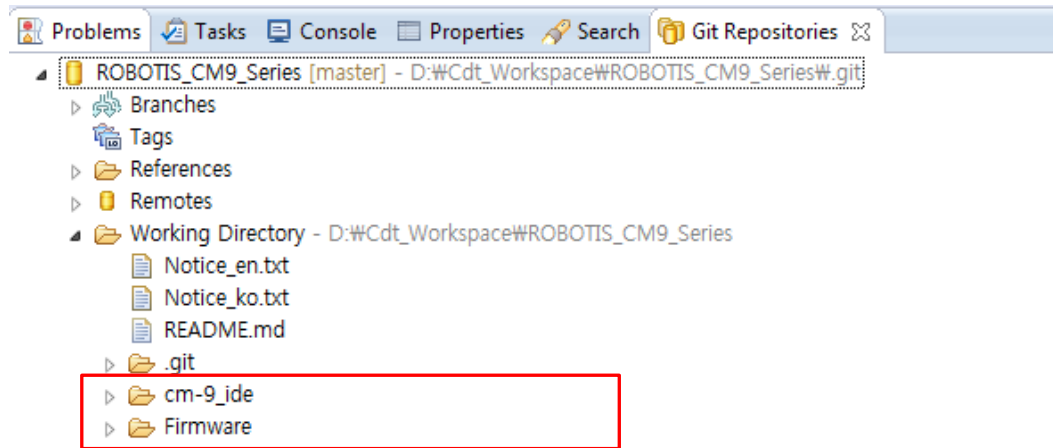
을 눌러서 작업할 디렉토리를 지정합니다.



Finish를 눌러서 본격적으로 소스를 다운받습니다. 이 부분은 시간이 조금 걸립니다.



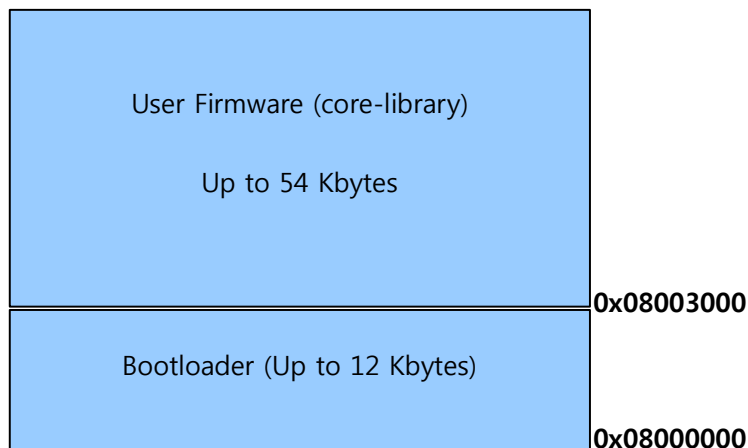
받기가 끝나면 아래와 같이 Master branch가 나타나고 하위 단계로 내려가면 ROBOTIS CM9 풀 소스가 나타납니다.



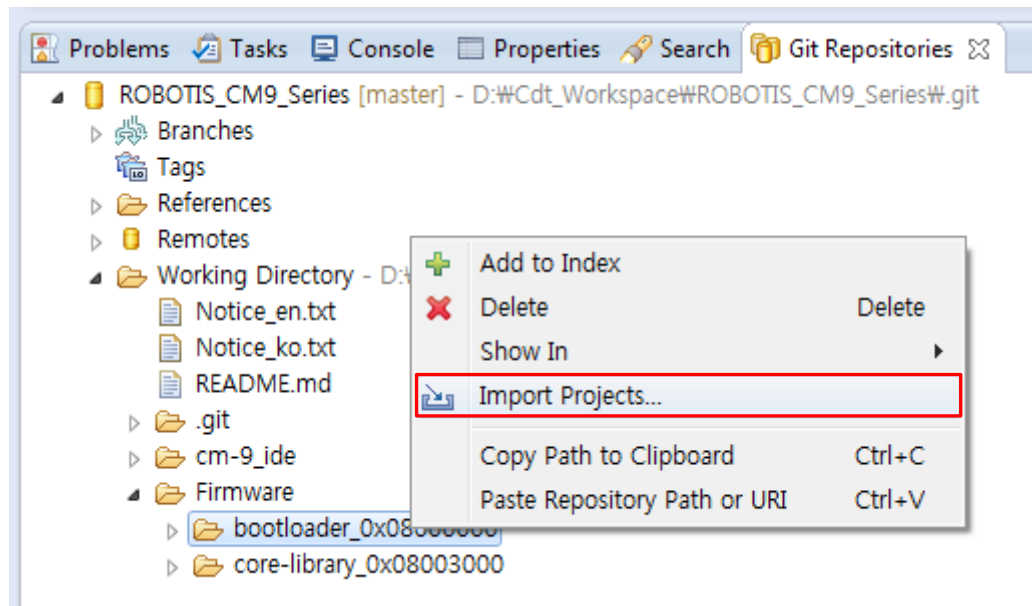
여기서 이클립스 C/C++ 패키지로 Firmware 디렉토리까지만 개발할 수 있습니다. ROBOTIS CM-9 IDE는 이클립스 Java 패키지를 통해 프로젝트를 등록해야 합니다.

② Bootloader 프로젝트 Import 하기

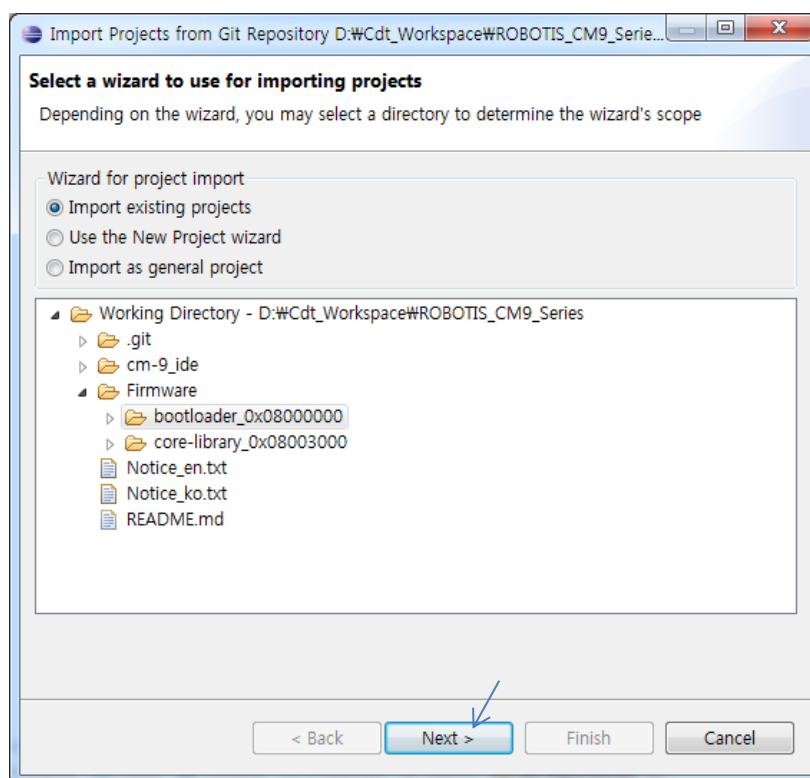
CM-900의 bootloader는 CPU인 STM32F103C8의 내부 Flash memory의 0x08000000 번지에서 시작하고 사이즈는 최대 12kbytes 내에서 빌드 되어야 합니다. 12 kbytes가 넘어가면 core-library 기반의 사용자 펌웨어 시작 위치 0x08003000을 넘어 버리기 때문에 사용자 펌웨어를 실행할 수 없습니다.



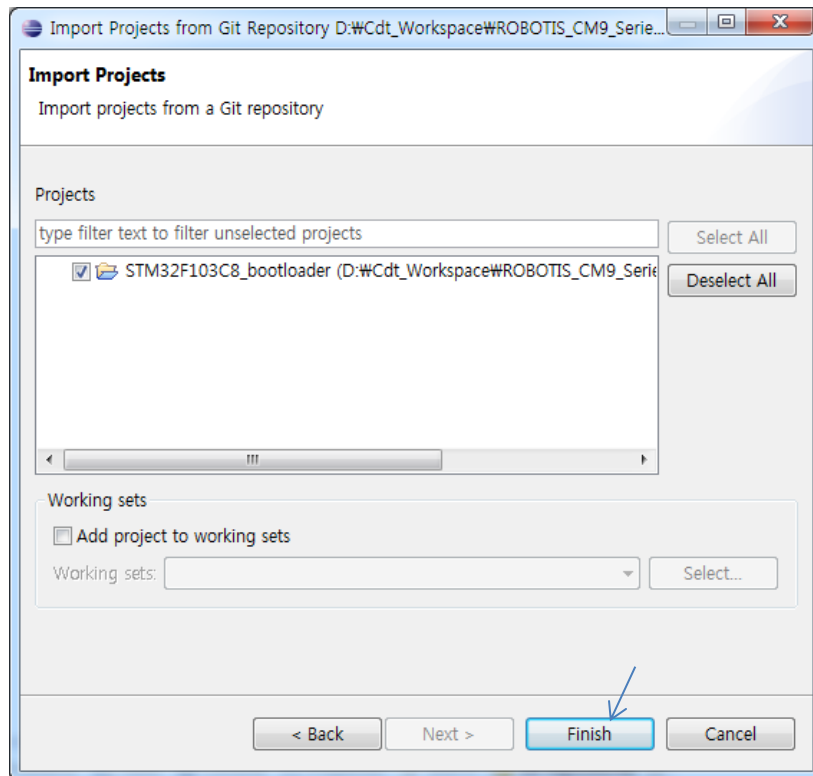
위 단계에서 받은 Repository에서 Working Directory 아래의 Firmware -> bootloader_0x08000000 항목을 선택하고 마우스 오른쪽 버튼을 클릭해서 Import Projects...를 선택합니다.



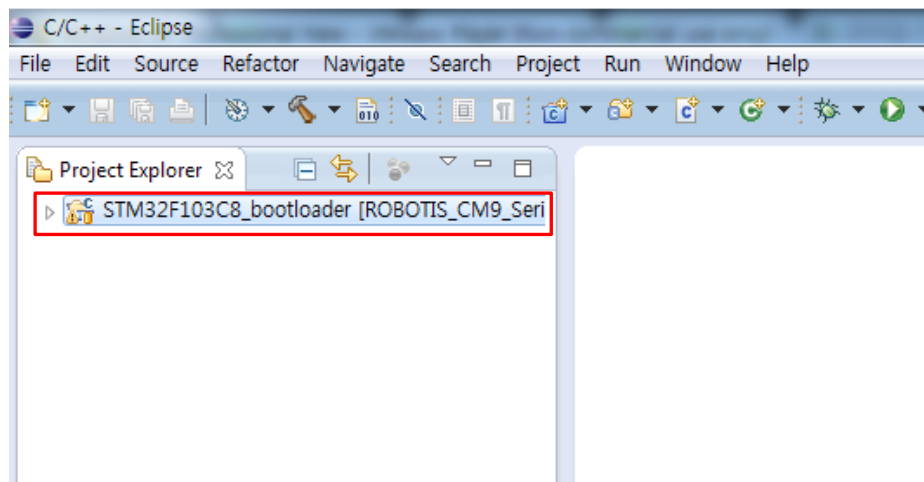
Wizard for project import에서 Import existing projects 항목을 선택하고 그대로 Next를 클릭합니다.



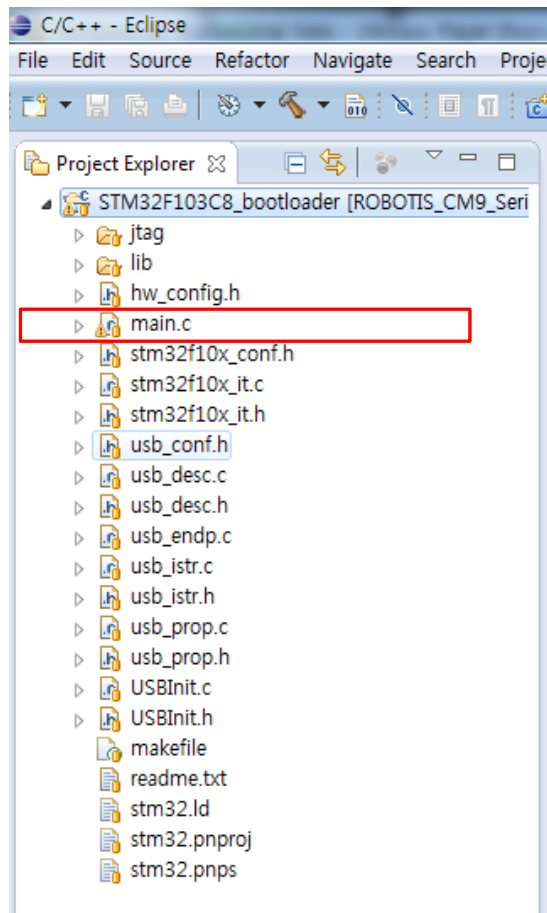
아래의 화면도 그대로 Finish하면 프로젝트가 이클립스에 포함됩니다.



Project Explorer에 포함되었습니다.



한 단계 내려보면 bootloader를 구성하는 C파일들이 나타납니다.



main.c 파일을 열어보면 아래와 같이 나타납니다.

```
main.c
+ /***** (C) COPYRIGHT 2008 STMicroelectronics *****/
- /*
 * @File: main.c
 * @Brief : main function for the bootloader of cm-9 series board.
 * changed by ROBOTIS,.LTD.
 */
/* Includes -----*/
#include "stm32f10x_lib.h"
#include "USBInit.h"

/* Private typedef -----*/
typedef enum {FAILED = 0, PASSED = !FAILED} TestStatus;

typedef void (*pFunction)(void);

/*
 * CM-900 Compile Option
 * 2012-08-29 ROBOTIS,.LTD. sm6787@robotis.com
 */
- //#define DEBUG_ENABLE_BY_USART2
- //#define POWER_SOURCE_DETECT
- //#define USE_USB_POWER_MANAGEMENT
- //#define USE_EEPROM_EMULATOR

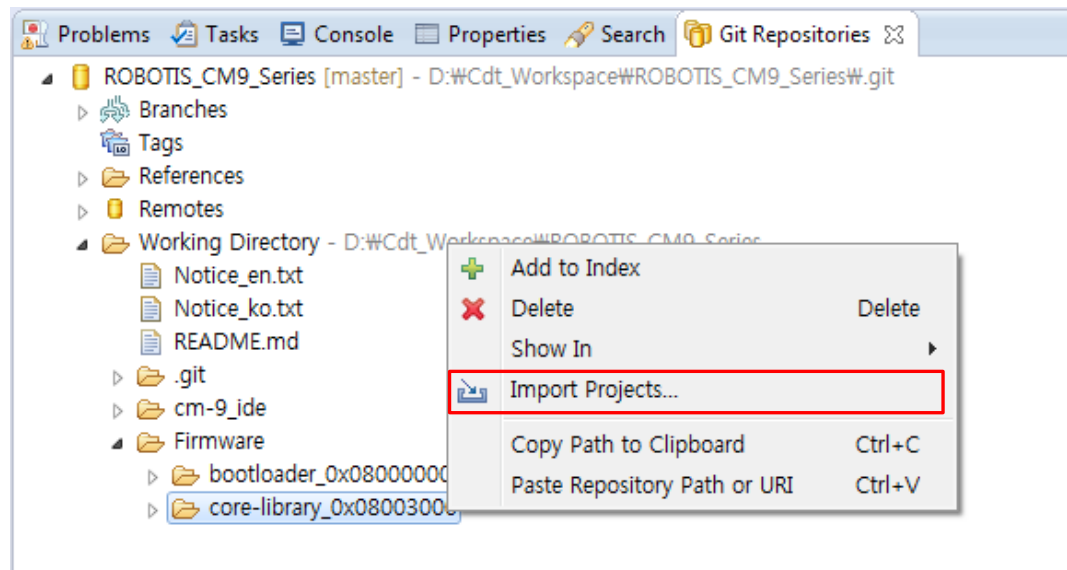
#define COMMAND_LENGTH 16
#define COMMAND_BUFFER_SIZE 80
#define PARA_NUM 10

#define P_OPERATING_MODE 19
```

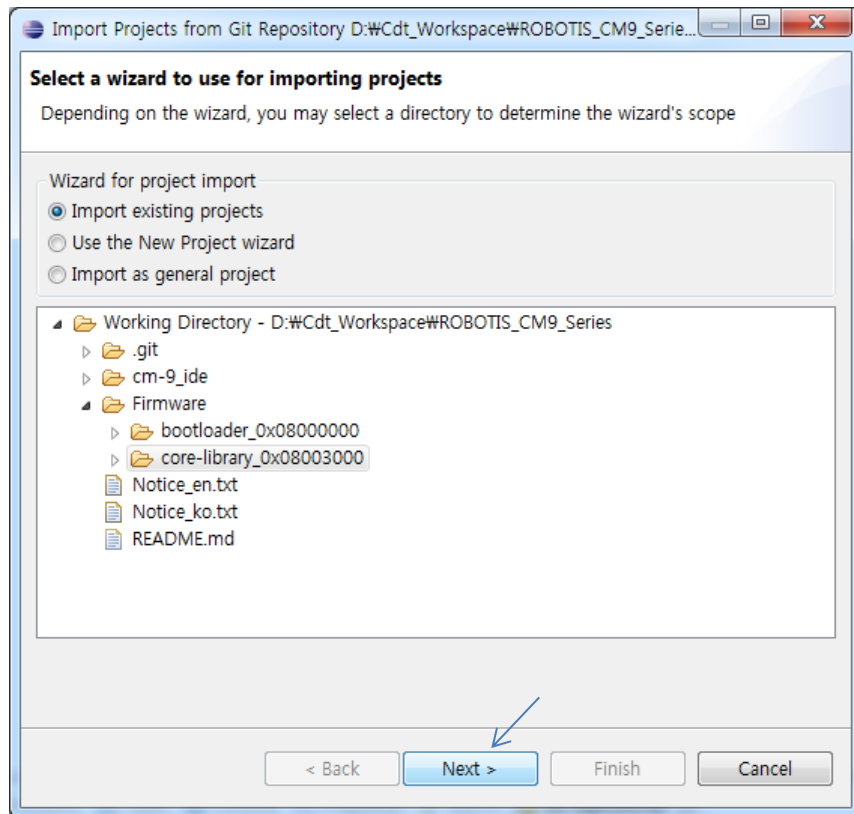
③ core-library 프로젝트 Import하기

core-library 프로젝트는 bootloader 프로젝트와는 다르게 Wiring 언어로 된 C++ 프로젝트로 이루어져 있습니다. 이제 core-library 프로젝트를 Import 합니다.

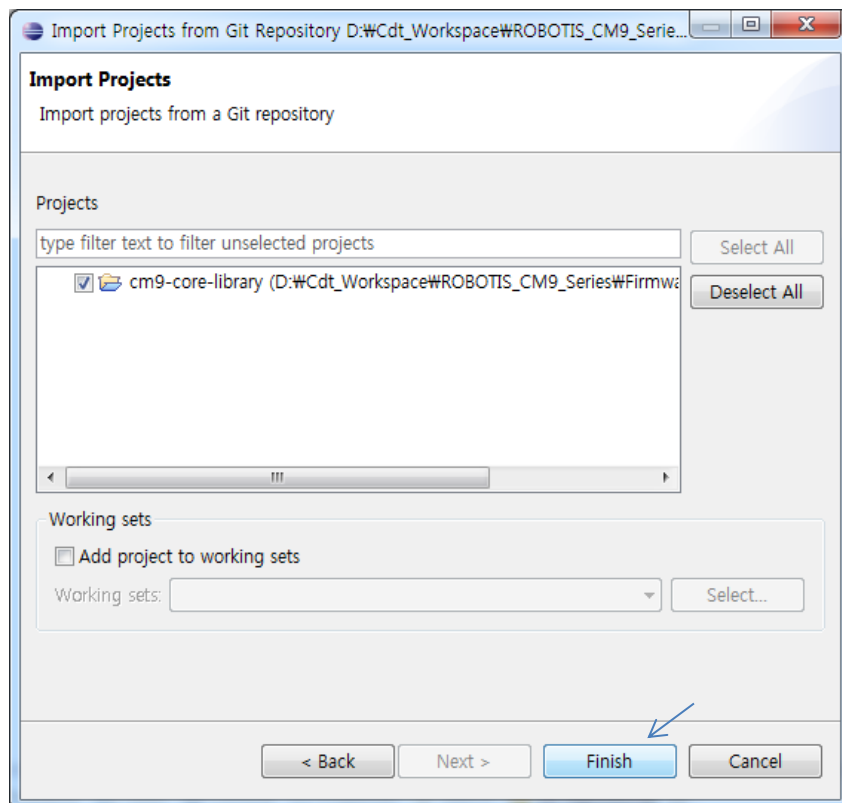
이클립스의 Git Repository View에서 core-library_0x08003000을 선택하고 팝업 메뉴의 Import Projects...를 선택합니다.



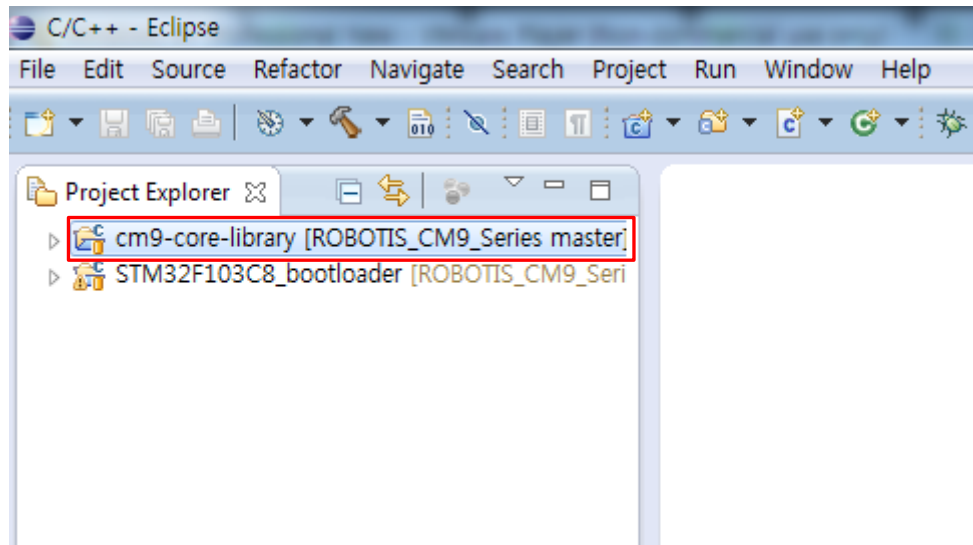
Select a wizard... 선택창에서 그냥 Next를 합니다.



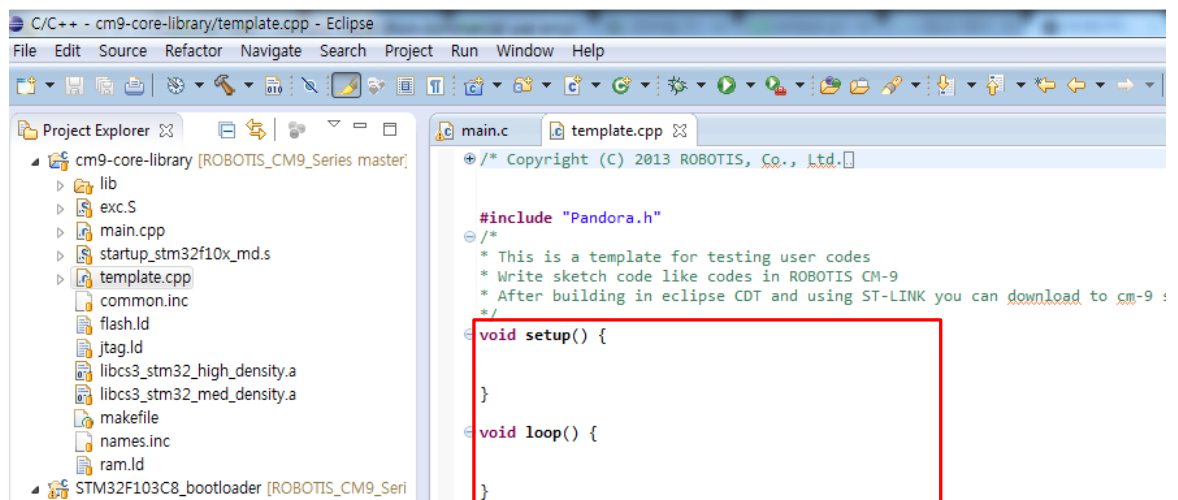
Import Projects 에서도 cm9-core-library가 나타나는 것을 확인하고 바로 Finish하면 Import가 끝납니다.



bootloader 프로젝트와 마찬가지로 cm9-core-library 프로젝트가 나타났습니다.



cm9-core-library 프로젝트를 아래의 template.cpp파일을 열어보면 ROBOTIS CM-9에서 보았던 익숙한 코드가 나타납니다.

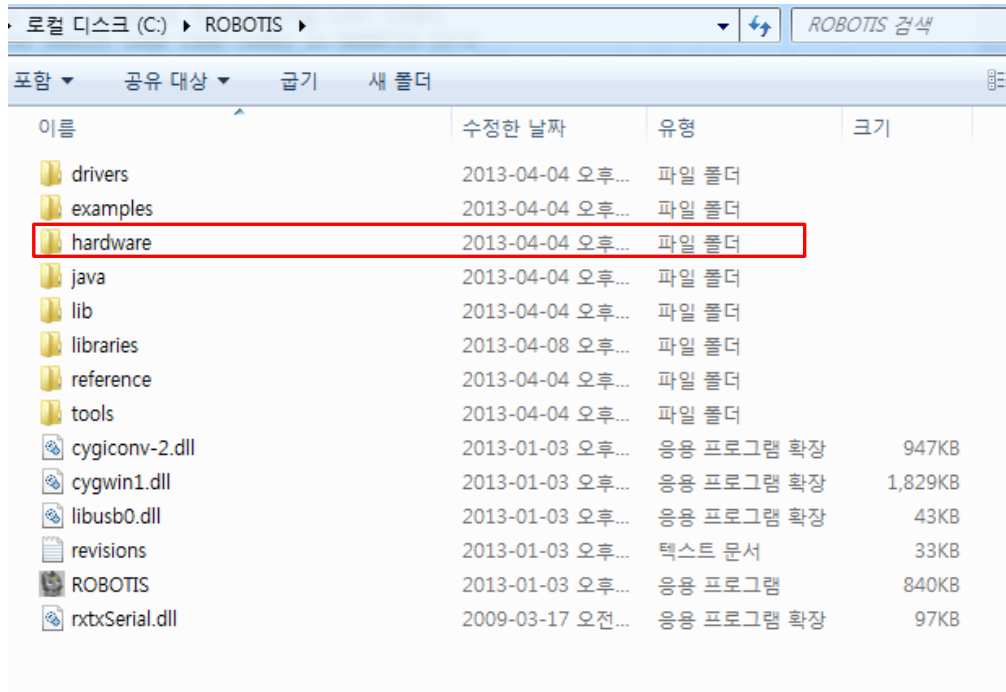


template.cpp의 setup()과 loop()함수를 채워서 ROBOTIS CM9을 이용하듯이 스케치 코드를 점검해볼 수 있습니다. 이렇게 이클립스의 뛰어난 코드 편집/추적 기능을 이용해서 강력한 로봇 개발 환경을 만들 수 있습니다.

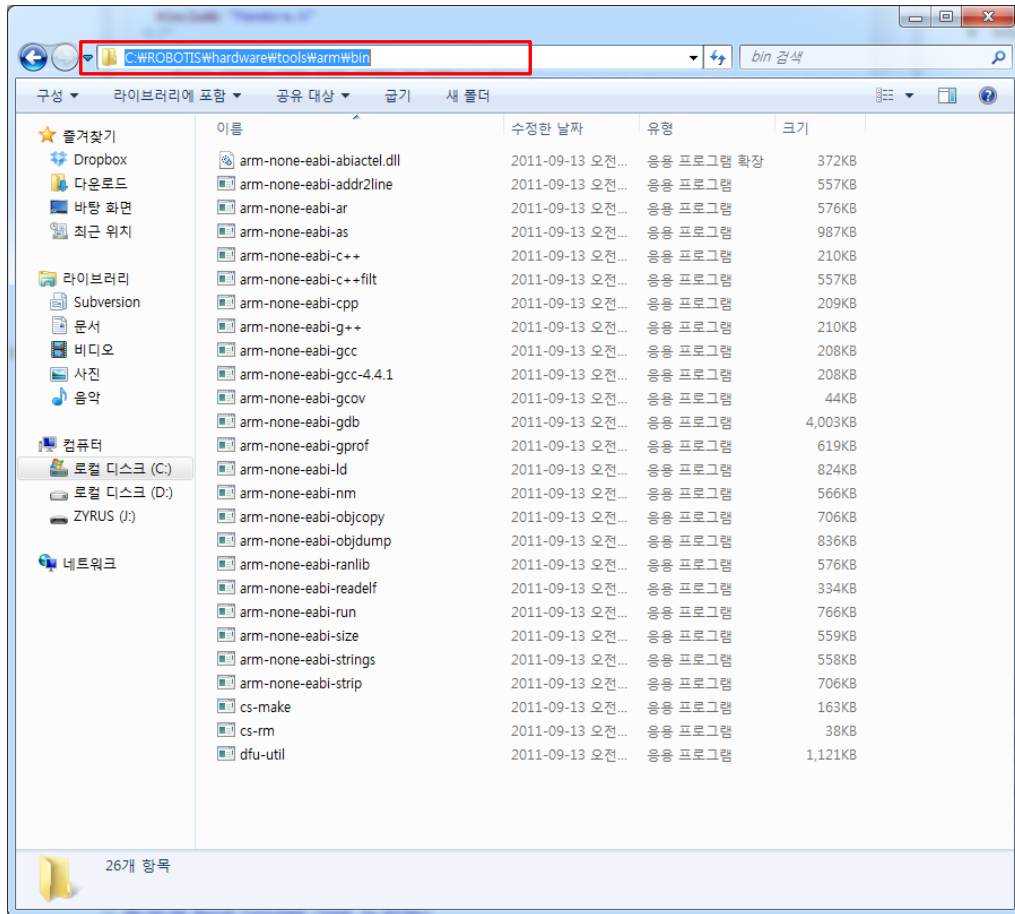
4. Code Sourcery G++ Lite 환경 변수 등록하기

CM-900의 부트로더를 포함한 모든 펌웨어는 ROBOTIS CM9 IDE에 포함된 Code Sourcery G++ Lite 툴체인을 이용해 빌드를 수행합니다.

ROBOTIS CM9을 다운로드 받았다면 Code Sourcery G++ Lite를 별도로 다운로드 받지 않고도 활용할 수 있습니다. ROBOTIS CM9 프로그램의 실행폴더에서 아래 경로에 툴체인이 있음을 확인합니다.

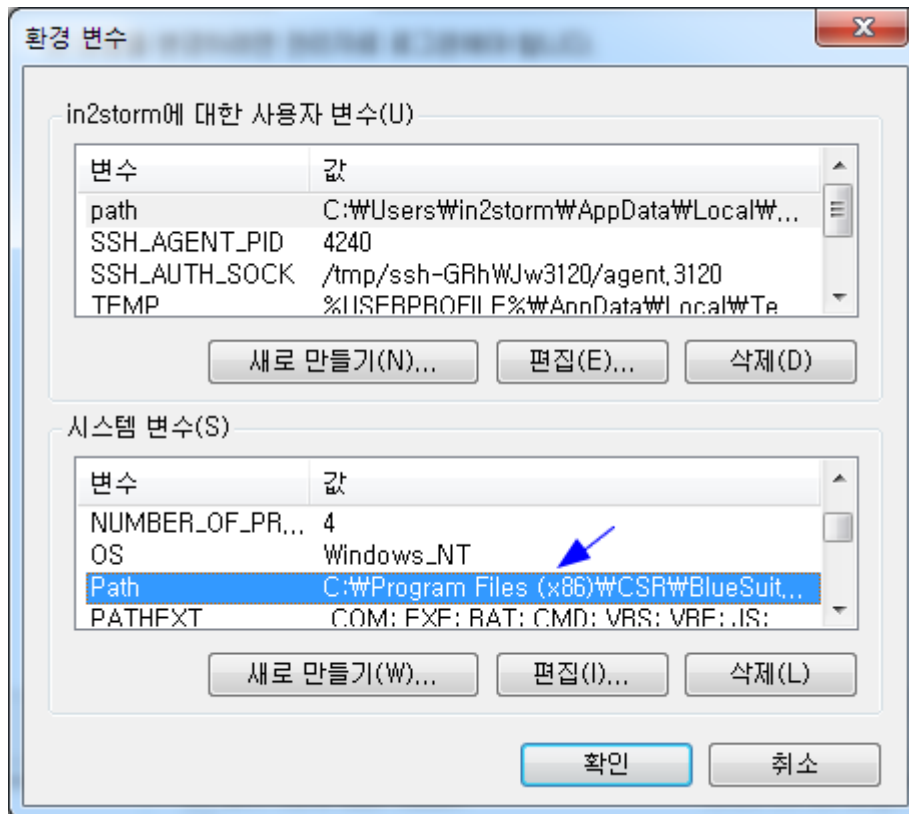


아래와 같이 ROBOTIS\hardware\tools\arm\bin 디렉토리에 arm-none-eabi-XXX 실행파일들이 있습니다.

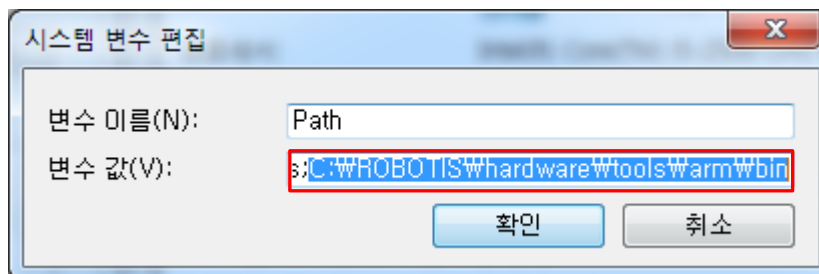


이 경로를 윈도우의 환경 변수에 등록합니다. 이렇게 등록한 디렉토리는 차후에 삭제하시면 이클립스에서 펌웨어 빌드가 되지 않으니 주의하세요.

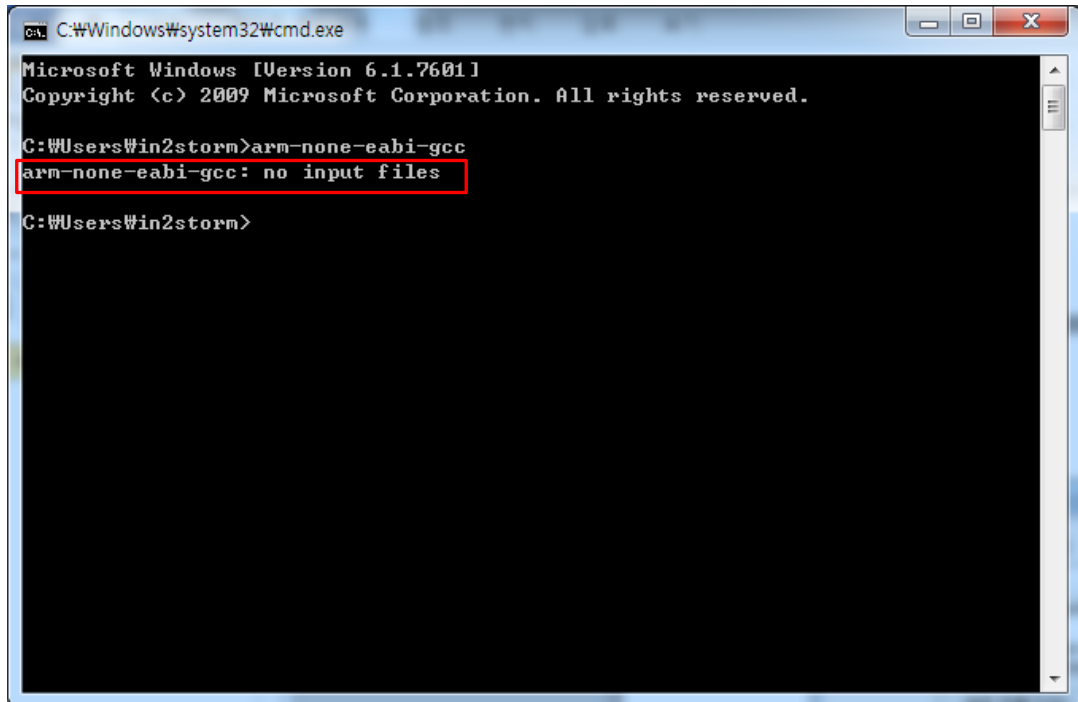
시스템 속성 -> 고급 -> 환경변수 를 클릭합니다.



아래와 같이 Path 항목의 맨 끝에 톨체인 경로를 입력하고 확인합니다.



이제 cmd창을 열어서 arm-none-eabi-gcc를 입력하면 아래와 같이 응답이 오면 정상적으로 환경변수가 등록되었습니다.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\win2storm>arm-none-eabi-gcc
arm-none-eabi-gcc: no input files

C:\Users\win2storm>
```

5. Bootloader 빌드 및 다운로드

전 단계에서는 gitHub를 통해 ROBOTIS CM-9 Series의 모든 소스를 받는 방법과 Code Sourcery G++ Lite 툴체인 환경변수 등록하는 방법을 배웠습니다.

이제는 ST-LINK장비를 이용해 이클립스에서 소스 빌드와 함께 다운로드까지 한 번에 수행하는 방법을 배워 보겠습니다.

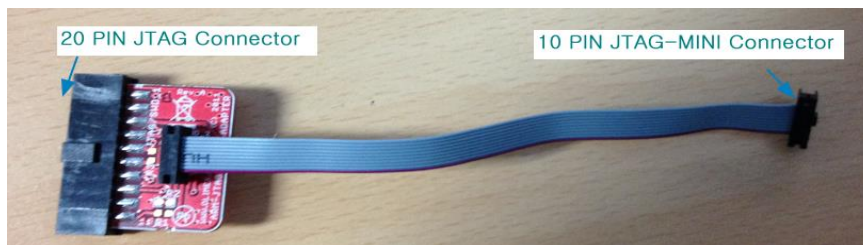
ST-LINK와 같은 JTAG 장비를 이용해서 다운로드하는 방법은 OpenOCD와같은 무료 디버깅 툴을 이용하거나 IAR이나 Keil같은 상용 툴을 사용하는 방법과 같이 여러 가지가 있지만 여기서는 STMicroelectronics에서 제공하는 ST-LINK 유틸리티를 활용하는 방법을 소개합니다.

단, **Appendix 1 부트로더 다운로드 방법과 같이 ST-LINK/V2와 20Pin to 10 Pin Converter는 CM-900의 구성품이 아니므로 반드시 별매로 구매하셔야 합니다.**

- ① ST-LINK와 CM-900을 연결합니다.



<ST-LINK/V2>

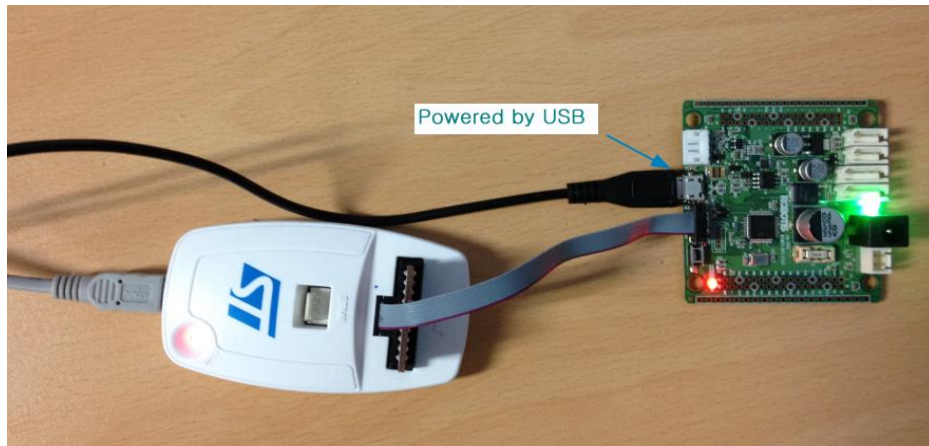


< 20 PIN to 10 PIN Converter >

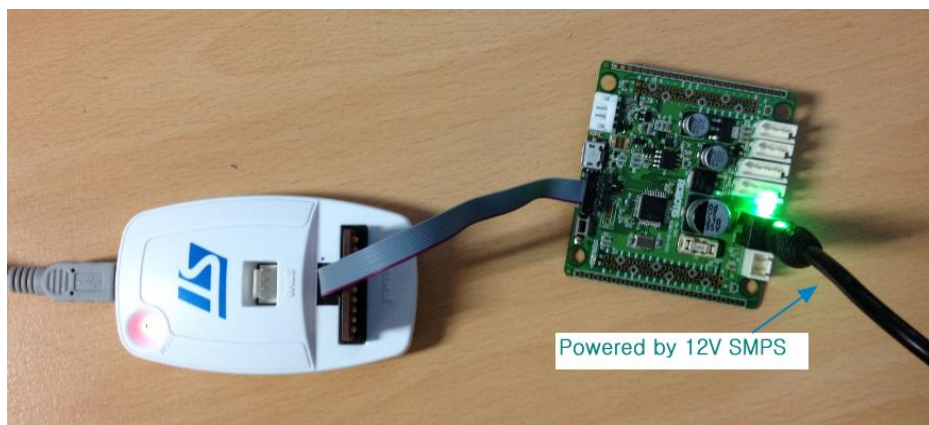
Converter의 20 PIN을 ST-LINK/V2에 연결하고 10 PIN을 CM-900의 아래 포트에 연결합니다.



- ② ST-LINK 연결을 위해서는 아래와 같이 CM-900에 USB 혹은 SMPS/Battery 등 별도의 전원이 있어야 합니다.



<USB 전원 별도 입력>




<SMPS 전원 별도 입력>

- ③ ST-LINK Utility를 다운로드 받습니다. (드라이버가 포함되어 있습니다.)

<http://www.st.com/web/en/catalog/tools/PF258168>

압축 파일을 풀고 STM32 ST-LINK Utility_v2.x.x.exe를 실행 시켜서 ST-LINK Driver까지 모두 설치 하면 아래와 같이 실행화면을 확인 합니다

다운로드 ▸ stsw-link004			
stsw-link004 검색			
포함 ▾	공유 대상 ▾	닫기	새 폴더
이름	수정한 날짜	유형	크기
 STM32 ST-LINK Utility_v2.5.0	2013-03-11 오후...	응용 프로그램	24,130KB

모두 설치 하였다면 ST-LINK를 USB 케이블을 이용해 PC와 연결합니다..

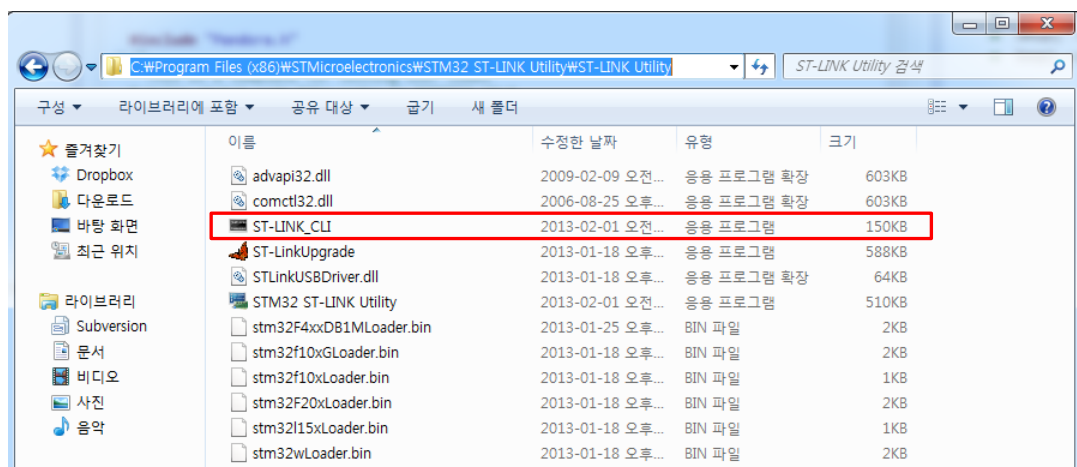


윈도우의 장치관리자 -> 범용 직렬 버스 컨트롤러 항목에서 STMicroelectronics STLink dongle 장치를 확인합니다.

여기까지는 Appendix 1과 같습니다. 이제 ST-LINK_CLI.exe라는 커맨드 모드 유틸리티를 이용해 이클립스의 Externl Tool로 등록하는 과정을 따라합니다.

- ④ ST-LINK_CLI.exe 위치를 확인합니다. 아래의 위치를 열기 합니다.

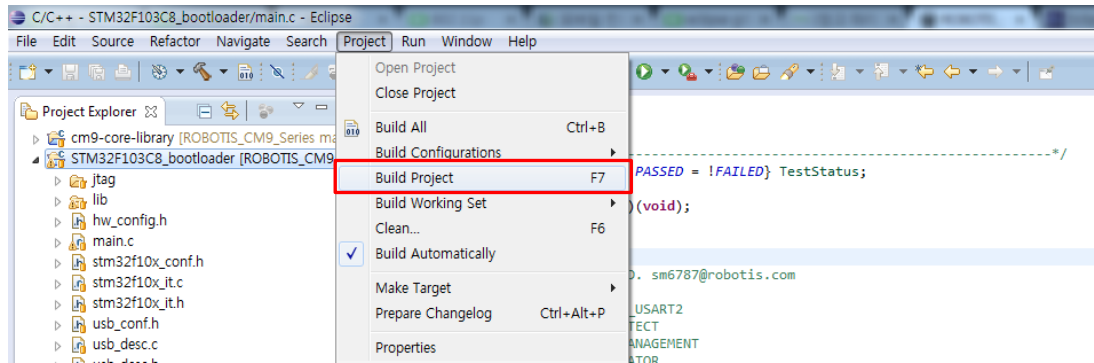
C:\Program Files (x86)\STMicroelectronics\STM32 ST-LINK Utility\ST-LINK Utility



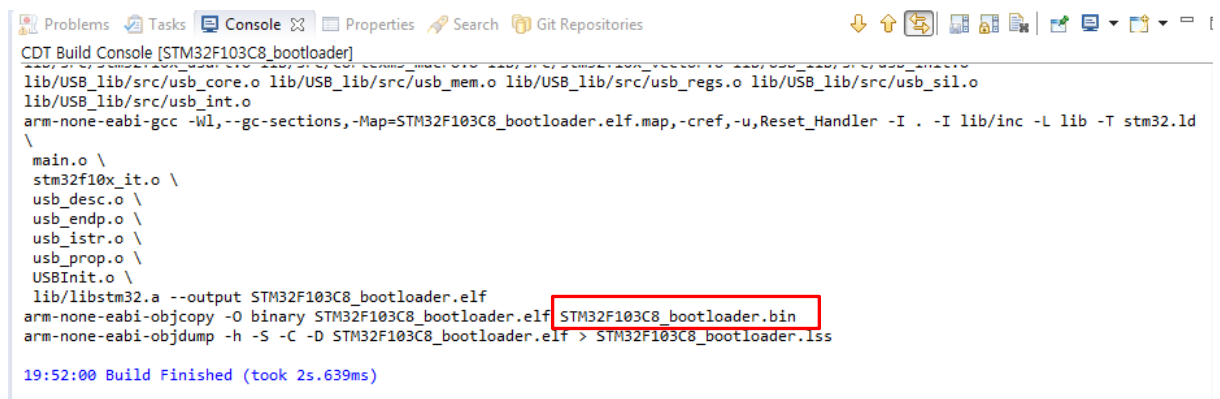
ST-LINK_CLI.exe 파일을 확인합니다. 이 경로를 이클립스에 등록해야 합니다.

- ⑤ 부트로더를 빌드합니다.

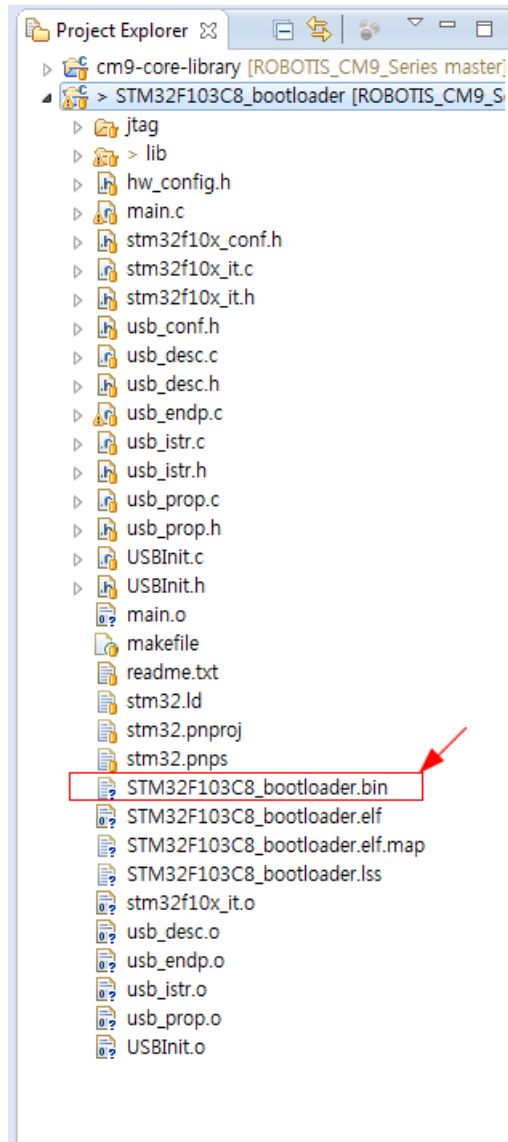
STM32F103C8_bootloader 프로젝트 내의 C파일 아무거나 열기합니다. 여기서는 main.c를 열기합니다. 활성화 된 C파일이 포함된 프로젝트가 빌드 됩니다.



Project -> Build Project 를 클릭하면 빌드가 시작되고 최종으로 아래와 같은 결과가 Console창에 나타난다면 성공적으로 빌드가 되어서 바이너리가 생성되었습니다.

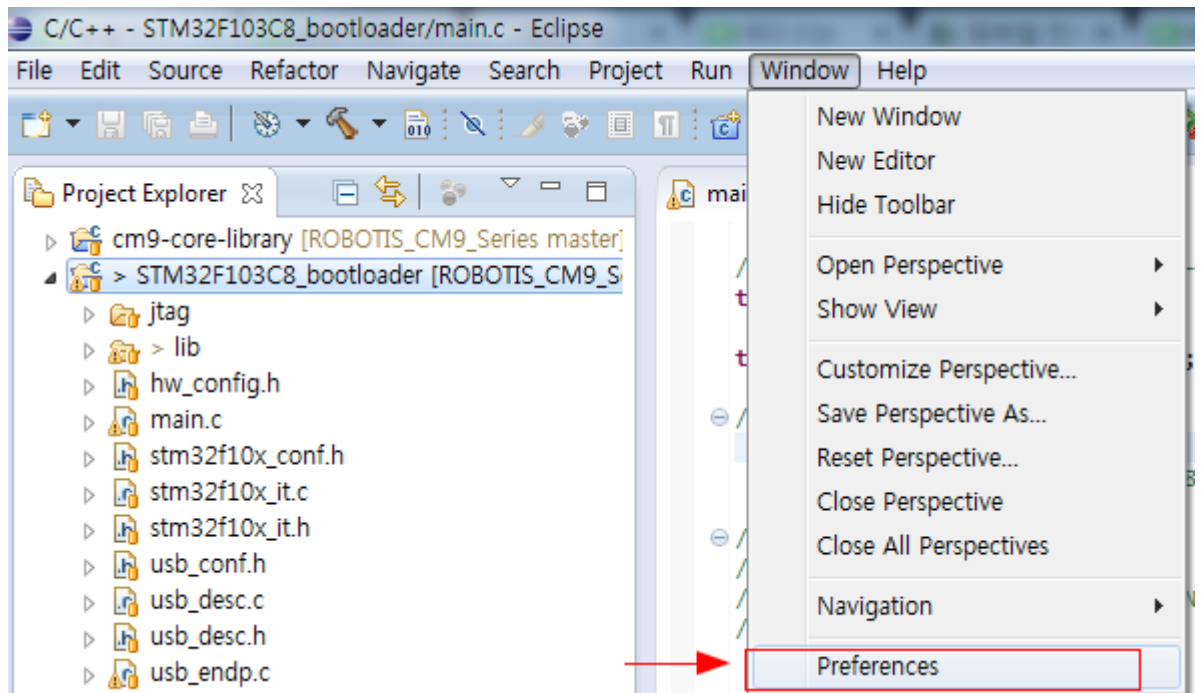


Project Explorer를 활성화한 뒤에 F5번 키로 Refresh하면 아래와 같이 STM32F103C8_bootloader.bin 파일이 생성되었음을 확인할 수 있습니다.

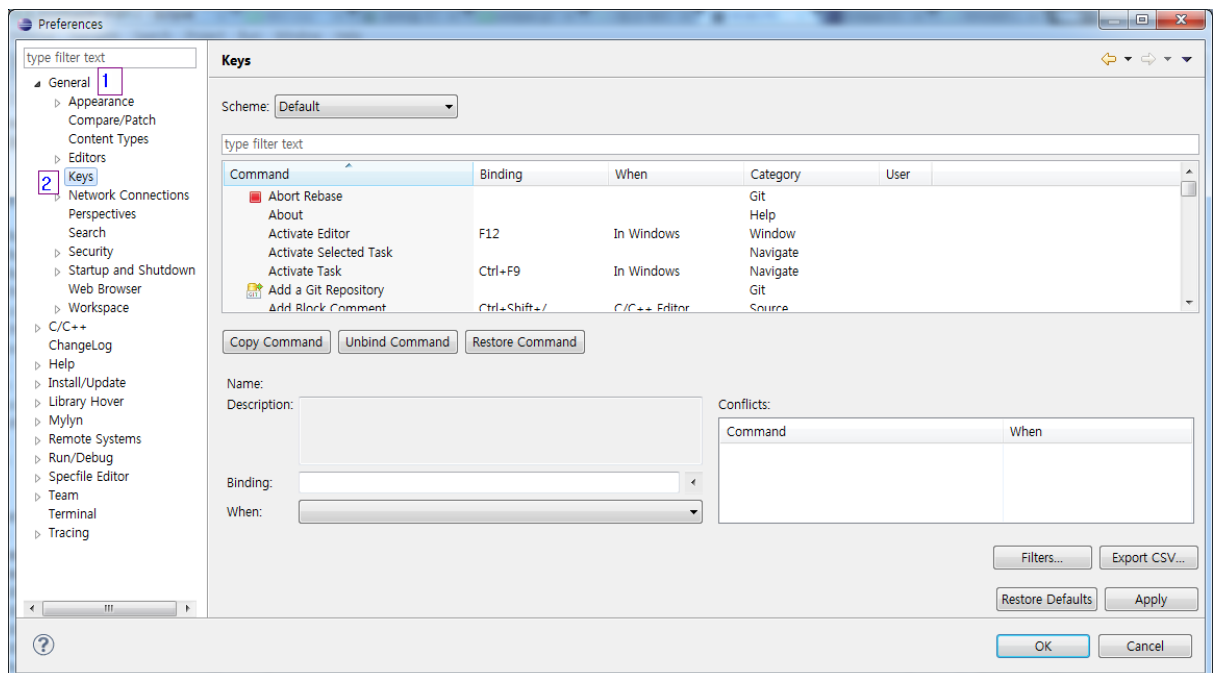


바이너리 생성 완료!

여기서 이클립스를 편리하게 사용할 수 있게 단축키를 설정하는 방법을 소개합니다.
반드시 해야되는 것은 아니므로 건너뛰셔도 무방합니다.

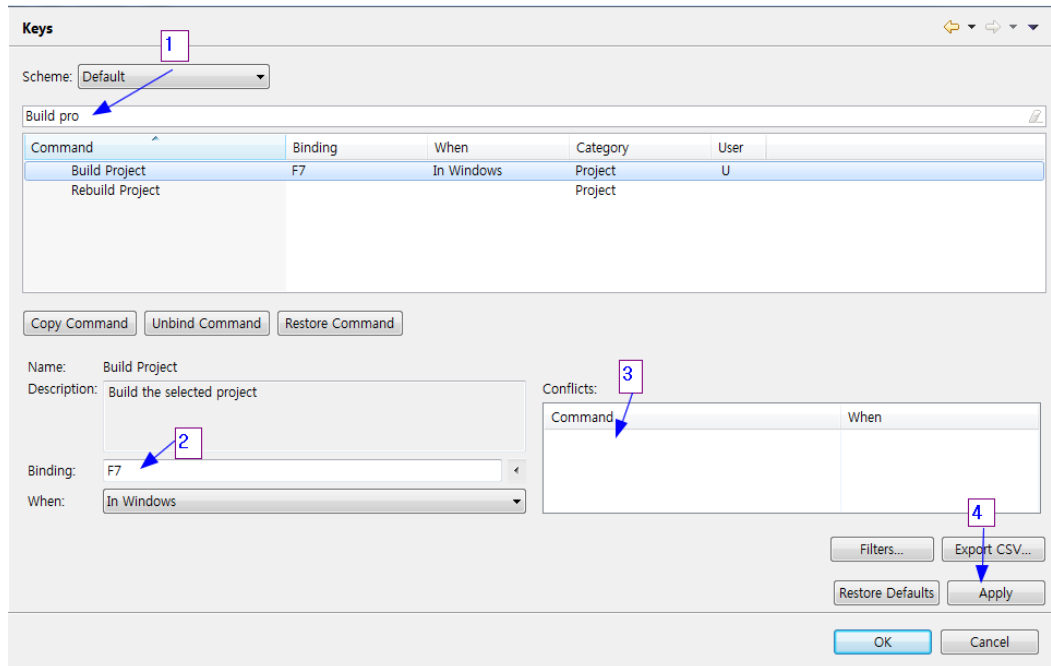


Window -> Preferences를 선택합니다. 이클립스의 일반적인 환경설정을 할 수 있습니다.

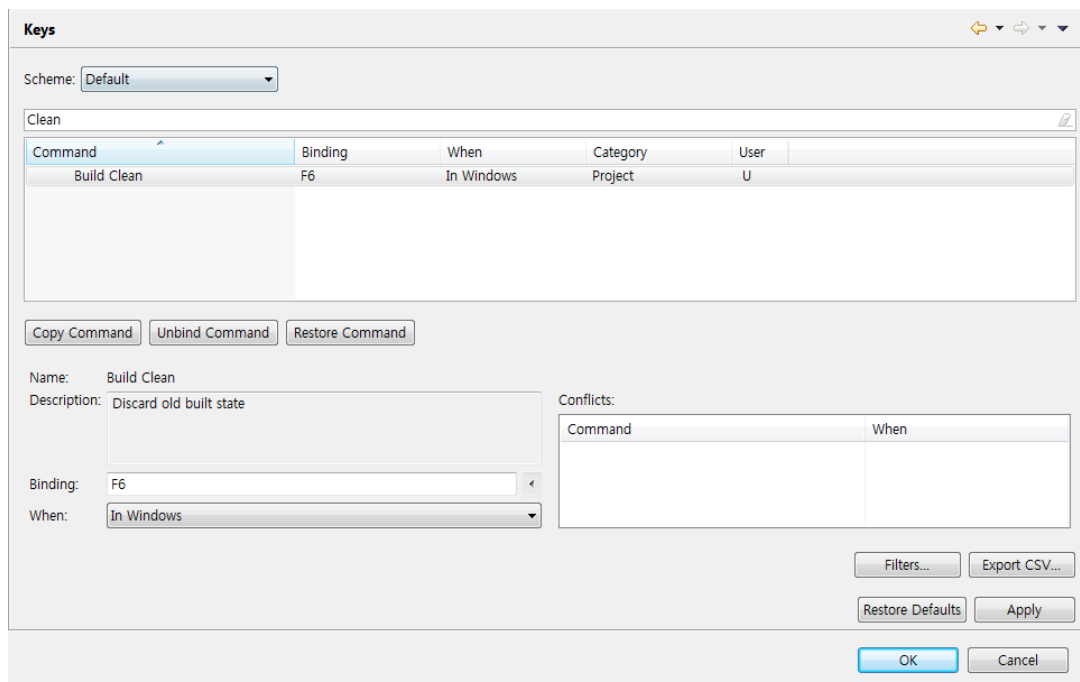


Preferences -> General -> Key 를 클릭합니다.

그리고 아래의 Search란에 Build project를 입력하면 Build Project라는 Command가 나타나는데 아래와 같이 Binding 항목을 클릭하고 F7을 지정합니다. 그리고 Conflicts 항목에서 충돌되는 단축키가 없는지 확인하고 없다면 Apply를 클릭합니다.

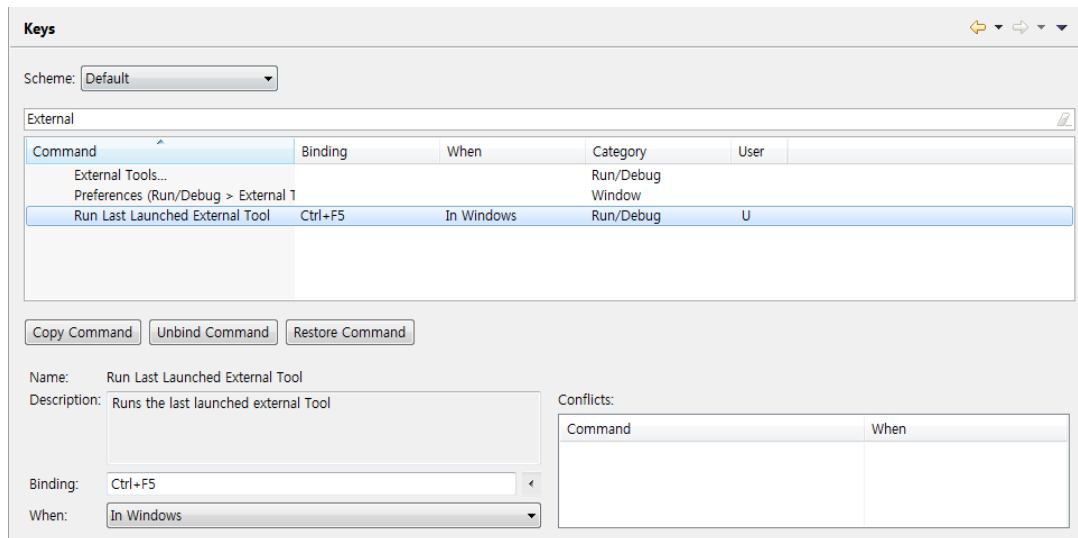


마찬가지로 Clean이라는 키워드로 커맨드를 검색한 다음 Build Clean 항목을 찾아서 F6 번으로 단축키를 Binding 합니다.



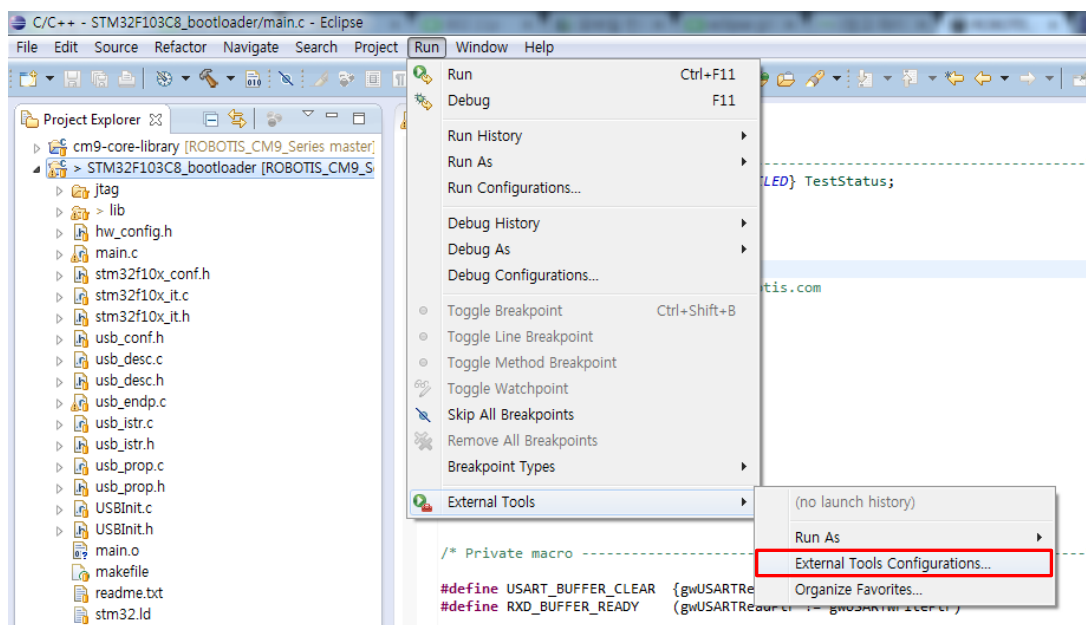
나중을 위해 External tools도 단축키를 등록합니다.

External이라는 키워드로 커맨드를 찾으면 Run Last Launched External Tool이라는 커맨드의 단축키를 Ctrl+F5번으로 Binding합니다.

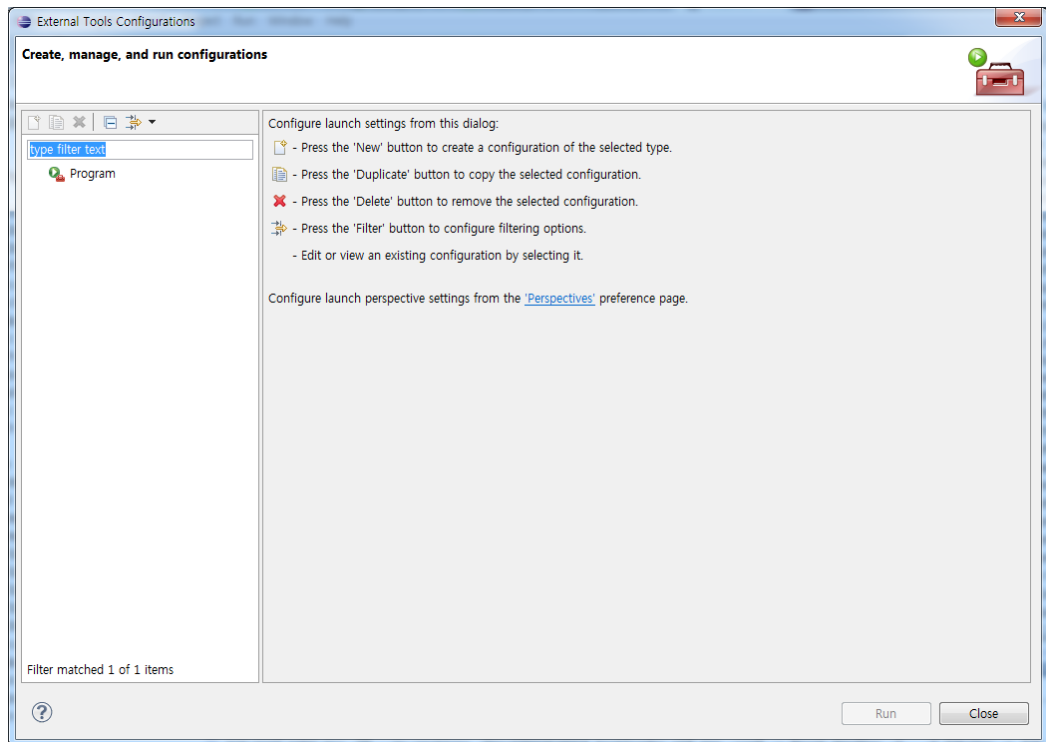


⑥ ST-LINK_CLI.exe를 External Tools로 등록하고 다운로드 합니다.

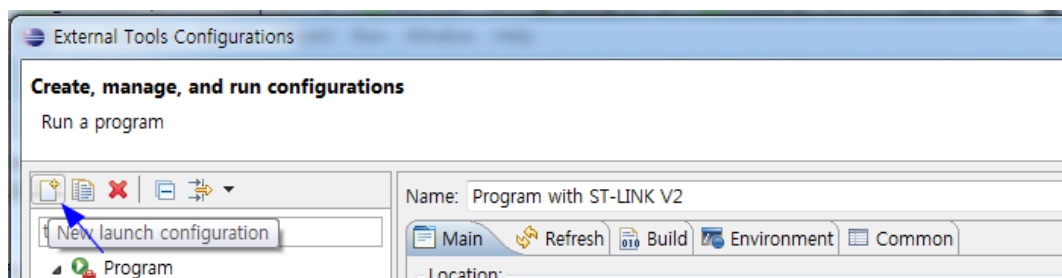
Run -> External Tools -> External Tools Configurations... 항목을 클릭합니다.



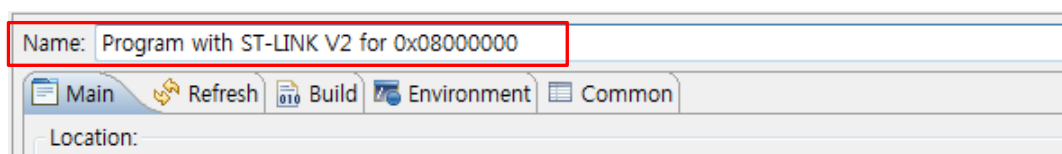
아래와 같은 화면이 나타나면.



New launch configuration 아이콘을 클릭합니다

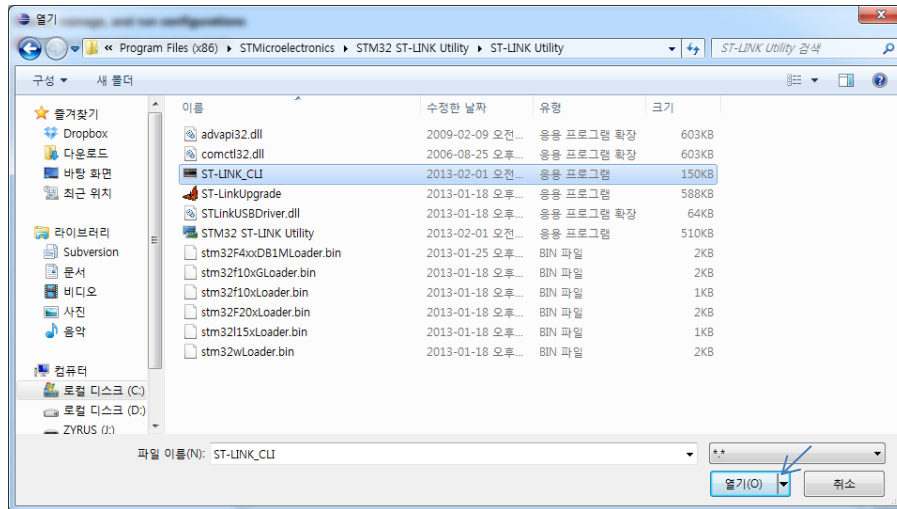


Name : Program with ST-LINK V2 for 0x08000000 을 입력합니다.

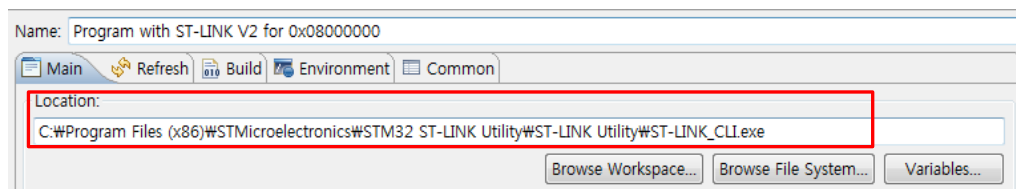


Main 탭의 Location : 항목의 Browse File System 버튼을 클릭해서 ST-LINK_CLI.exe가 있는 경로를 선택합니다. 여기서는 아래의 경로를 예로 들겠습니다.

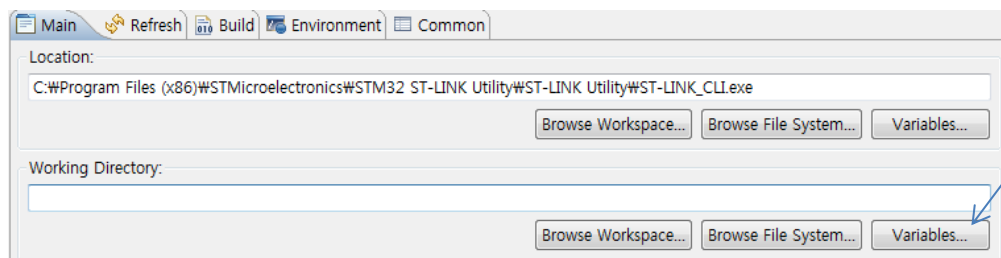
C:\Program Files (x86)\STMicroelectronics\STM32 ST-LINK Utility\ST-LINK Utility\



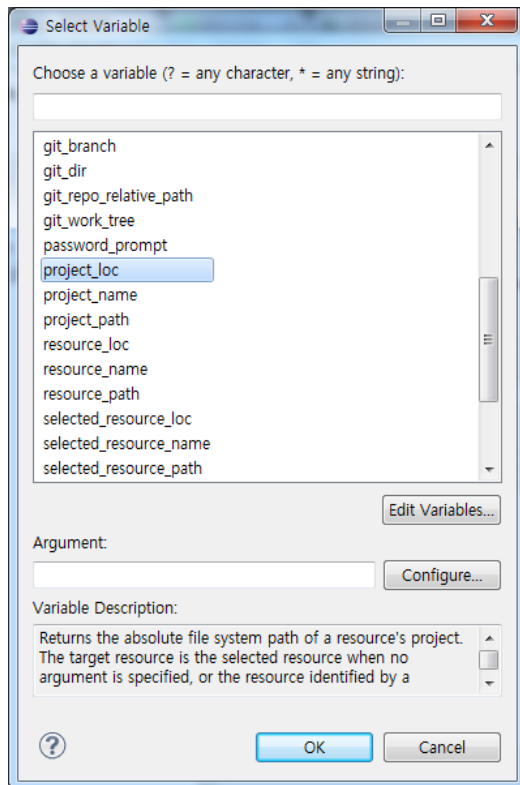
ST-LINK_CLI.exe를 선택하고 열기 합니다.



이번에는 Working Directory를 지정합니다. 이클립스에 등록된 공통 환경 변수를 이용합니다. Variables... 버튼을 클릭합니다.



현재의 프로젝트 디렉토리에 해당하는 \${project_loc}을 선택합니다.



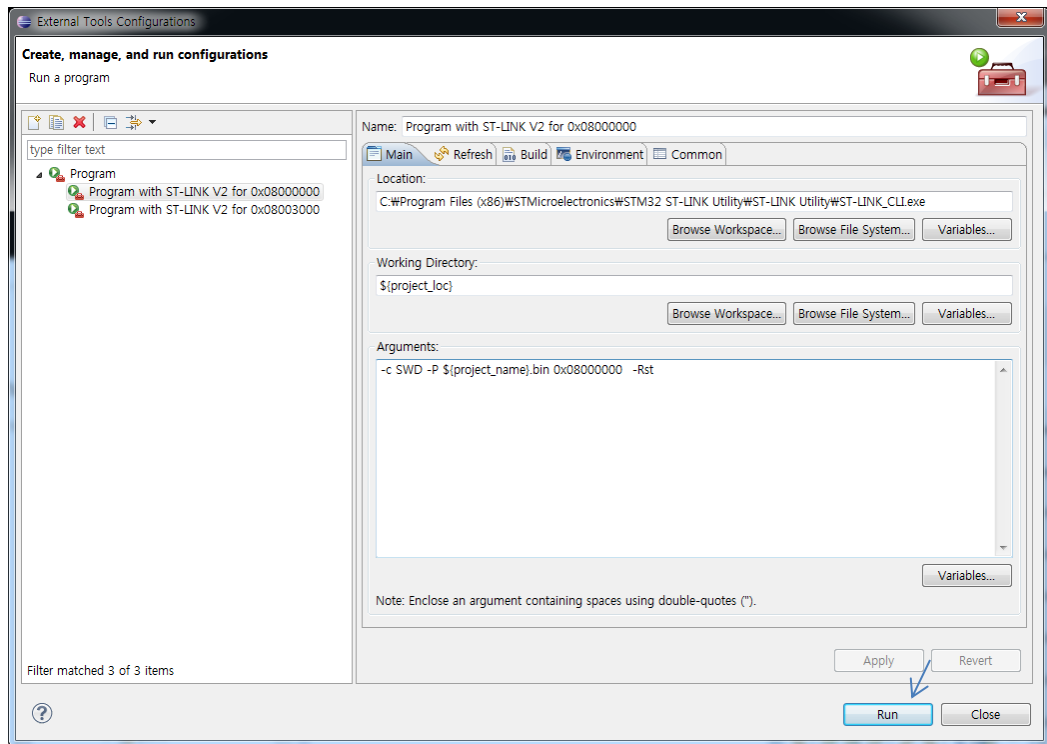
이제 마지막으로 Argument를 등록합니다.

Arguments는 ST-LINK_CLI.exe에 넘겨주는 인자입니다.

-c SWD -P \${project_name}.bin 0x08000000 -Rst

입력합니다. 여기서 0x08000000의 자리수에 주의 하세요

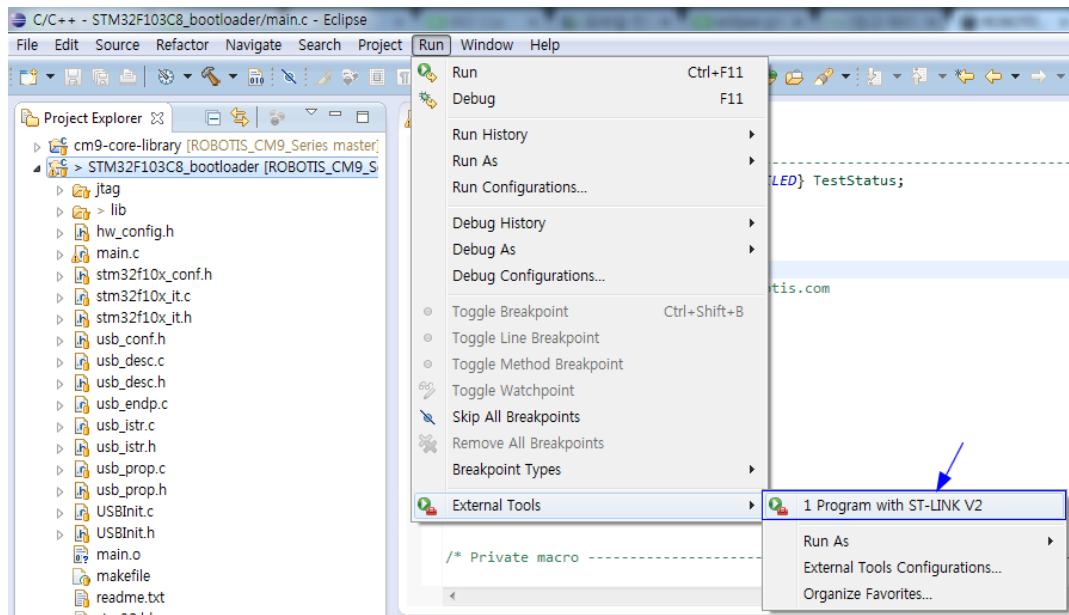
이제 Apply 버튼을 클릭하고 Run 버튼을 눌러서 다운로드를 시도합니다.



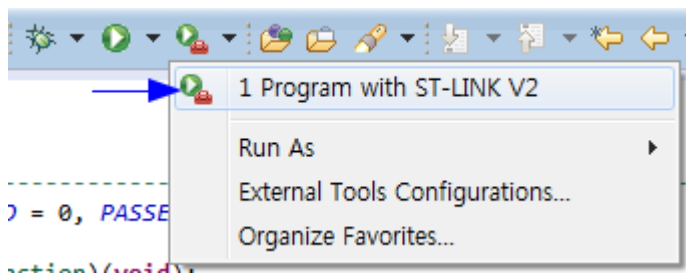
콘솔창에 아래와 같은 결과가 나타났다면 다운로드가 성공적으로 끝났습니다.



- ⑦ External Tool로 등록한 ST-LINK_CLI.exe는 이전 단계에서 설정한 단축키 또는 단축아이콘을 통해서도 바로 실행할 수 있습니다.



또는 Ctrl + F5를 누르거나 아래 단축 아이콘을 통해서도 수행됩니다.



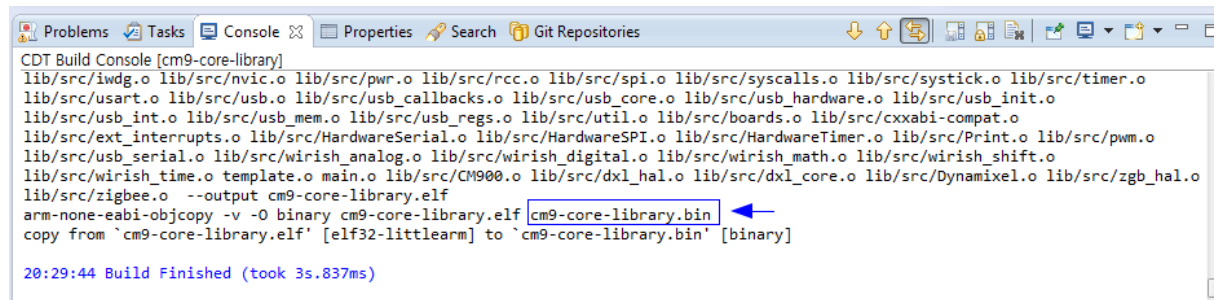
6. core-library 프로젝트 빌드 및 다운로드

- ① cm9-core-library 프로젝트를 빌드 합니다.

template.cpp파일을 클릭해서 열기합니다. 또는 cm9-core-library 프로젝트내에 어떤 파일이든 열기 합니다.



위에서 설정한 단축키 F7를 누르거나 Project -> Project Build 를 클릭합니다.

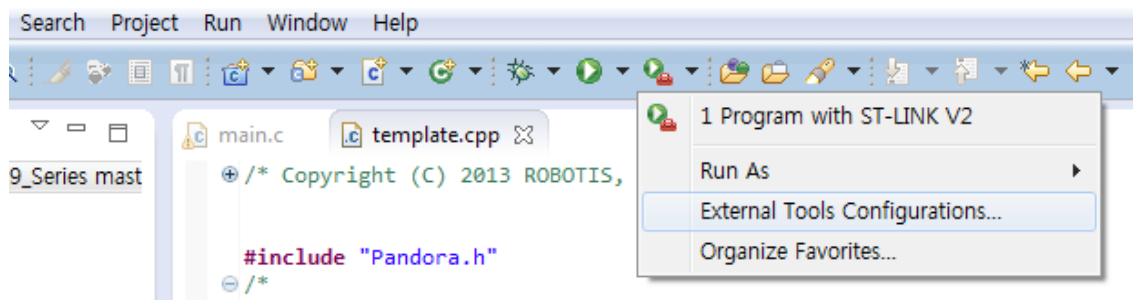


```
CDT Build Console [cm9-core-library]
lib/src/iwdg.o lib/src/nvic.o lib/src/pwr.o lib/src/rcc.o lib/src/spi.o lib/src/syscalls.o lib/src/systick.o lib/src/timer.o
lib/src/usart.o lib/src/usb.o lib/src/usb_callbacks.o lib/src/usb_core.o lib/src/usb_hardware.o lib/src/usb_init.o
lib/src/usb_int.o lib/src/usb_mem.o lib/src/usb_regs.o lib/src/util.o lib/src/boards.o lib/src/cxxabi-compat.o
lib/src/ext_interrupts.o lib/src/HardwareSerial.o lib/src/HardwareSPI.o lib/src/HardwareTimer.o lib/src/Print.o lib/src/pwm.o
lib/src/usb_serial.o lib/src/wirish_analog.o lib/src/wirish_digital.o lib/src/wirish_math.o lib/src/wirish_shift.o
lib/src/wirish_time.o template.o main.o lib/src/CM900.o lib/src/dxl_hal.o lib/src/dxl_core.o lib/src/Dynamixel.o lib/src/zgb_hal.o
lib/src/zigbee.o --output cm9-core-library.elf
arm-none-eabi-objcopy -v -O binary cm9-core-library.elf cm9-core-library.bin
copy from 'cm9-core-library.elf' [elf32-littlearm] to 'cm9-core-library.bin' [binary]

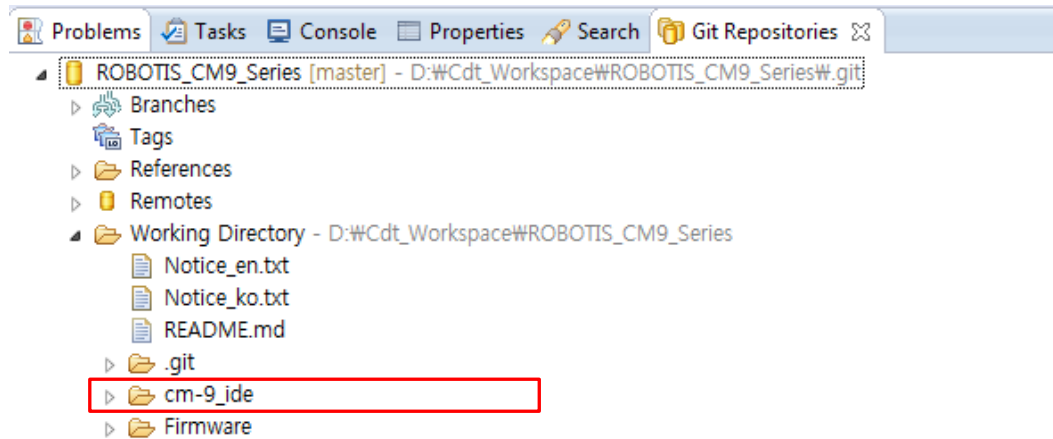
20:29:44 Build Finished (took 3s.837ms)
```

② External Tool Configuration for cm9-core-library

cm9-core-library.bin 프로젝트를 다운로드하기 위해서는 새롭게 External Tool Configuration을 설정해야 합니다. 왜냐하면 bootloader와 사용자 펌웨어 시작 주소가 다르기 때문입니다. cm9-core-library는 CM-900의 Flash memory 0x08003000번지에서 시작합니다.

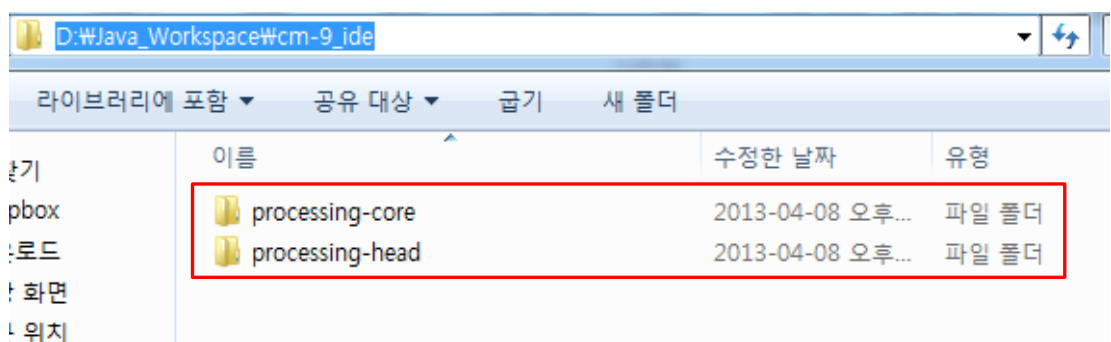
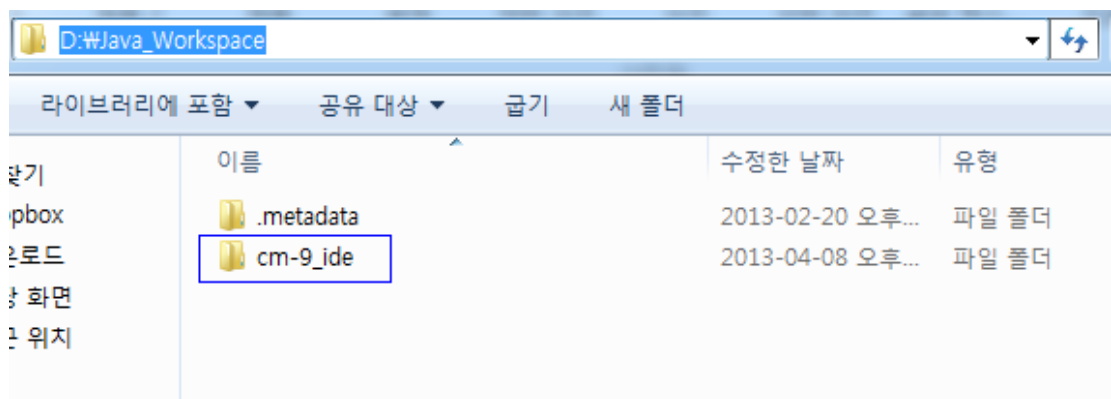


아래와 같이 새로운 External tool을 만듭니다.

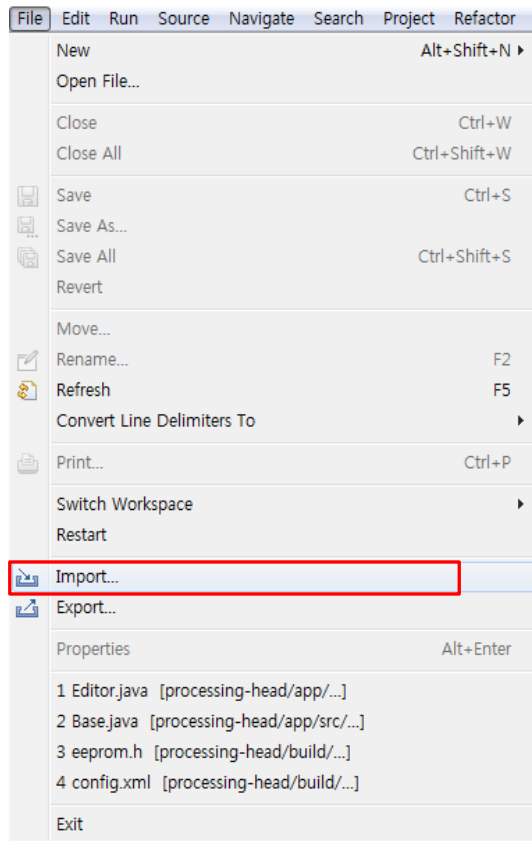


여기서는 D:\Cdt_Workspace\ROBOTIS_CM9_Series를 기준으로 설명합니다. 해당 폴더의 cm-9_ide 폴더 전체를 다른 디렉토리에 복사해서 Import해도 무방합니다.

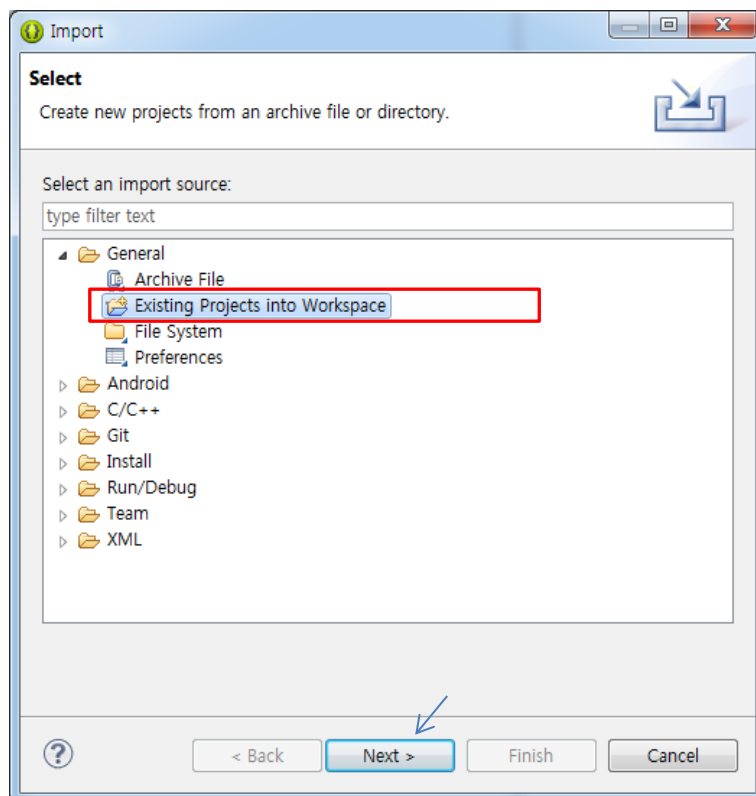
cm-9_ide 폴더를 java의 Workspace 아래로 복사합니다.



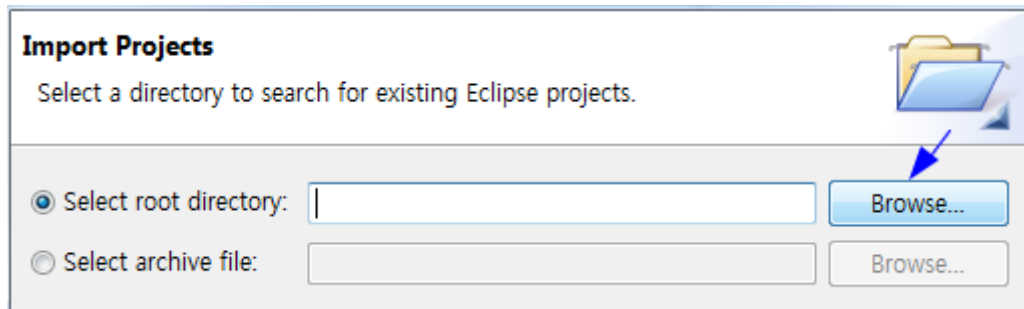
이클립스의 File -> Import를 선택합니다.



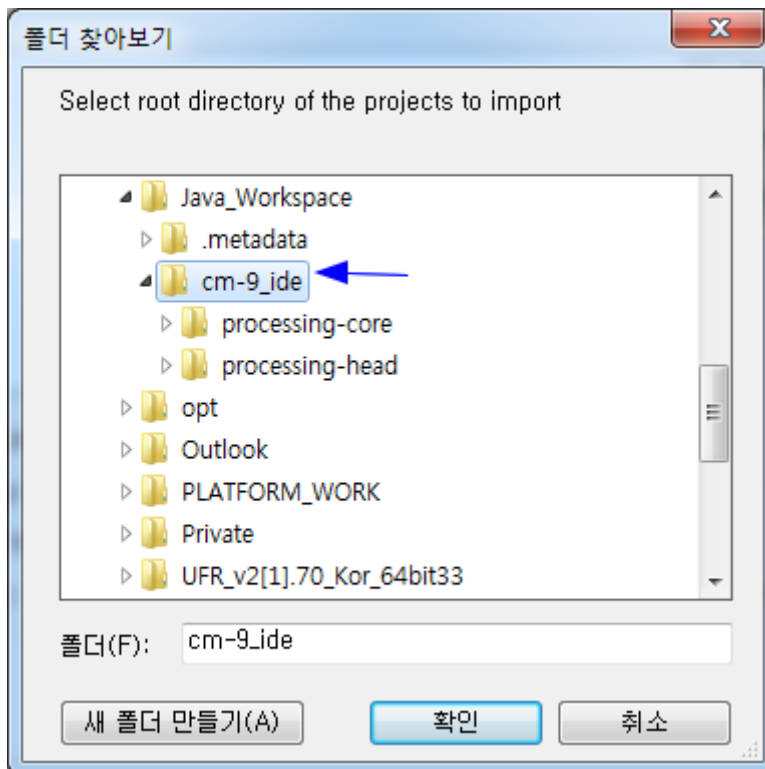
General -> Existing Projects into Workspace를 선택하고 Next합니다.



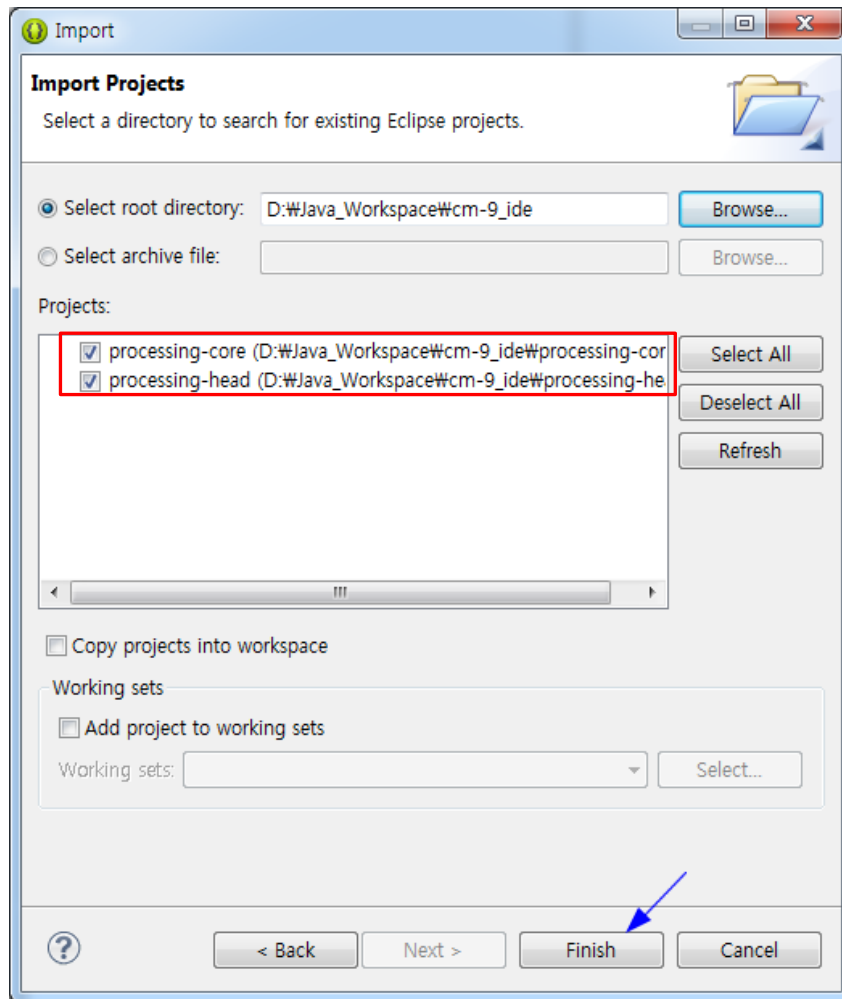
Browse... 버튼을 누릅니다.



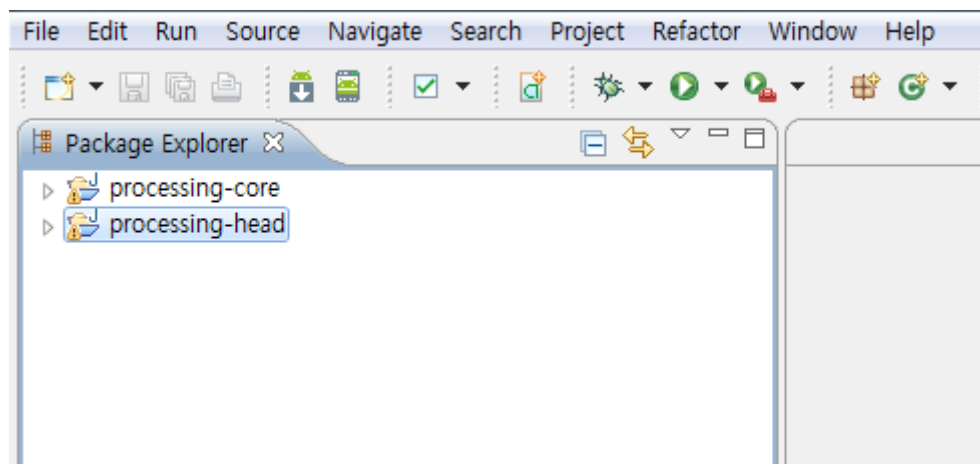
이전 단계에서 복사했던 cm-9_ide 폴더를 지정합니다.



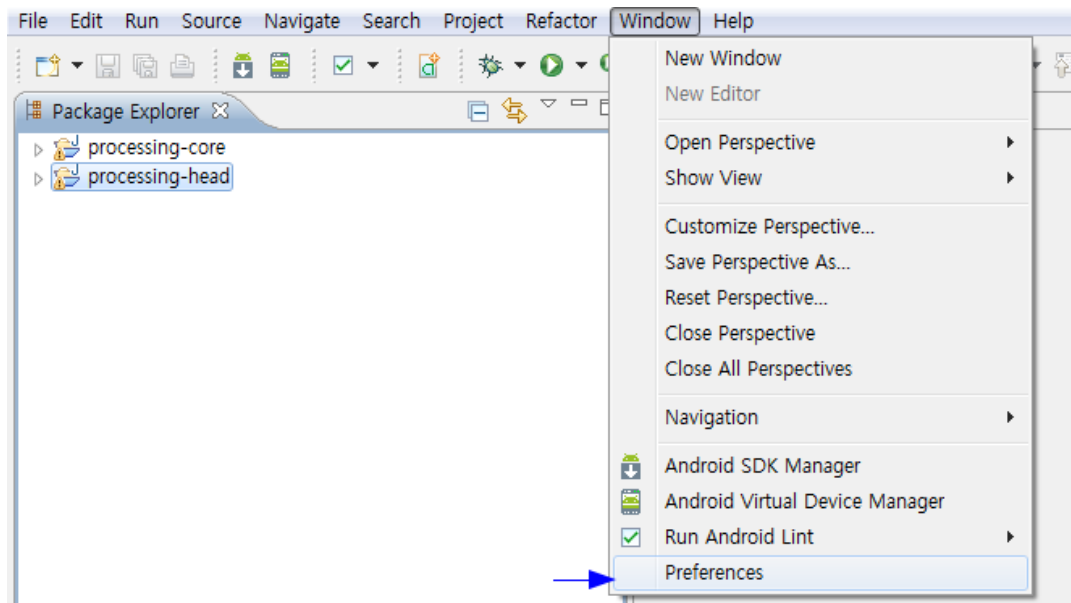
확인을 누르면 다음과 같이 두 개의 프로젝트가 인지되었습니다.



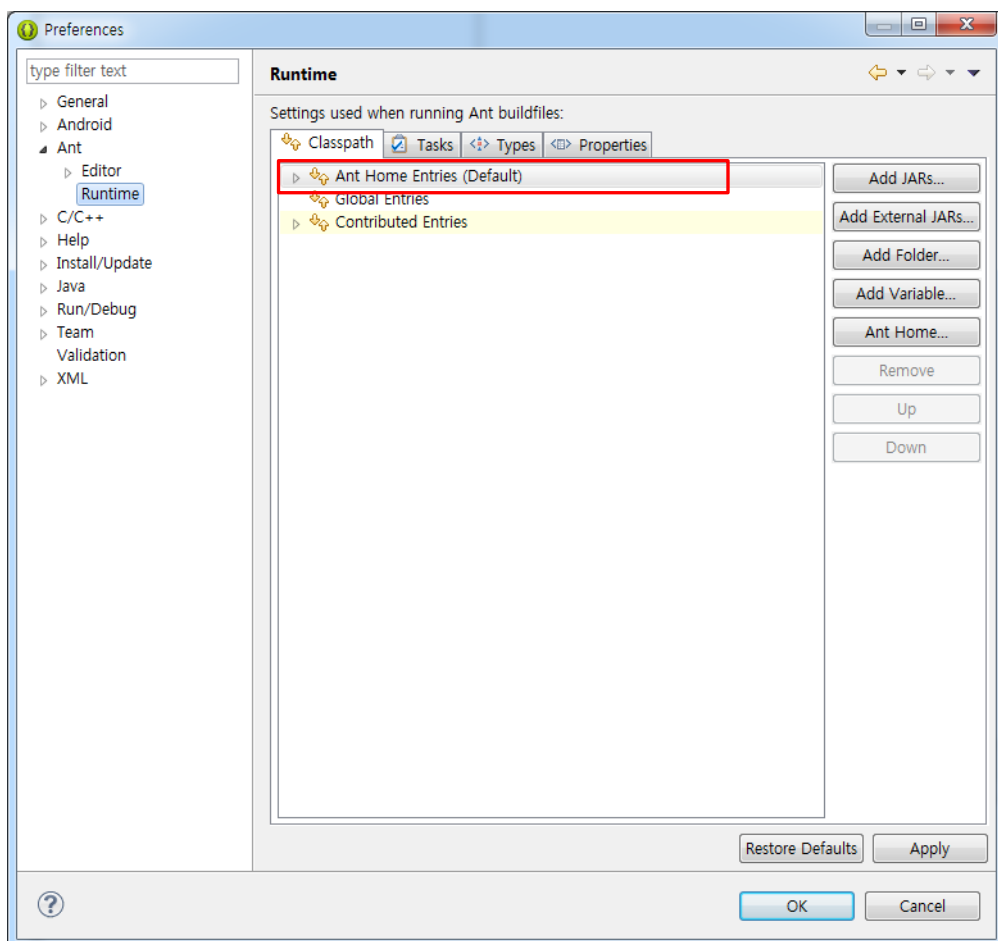
Finish를 누르면 Package Explorer에 두 개의 프로젝트가 등록되었습니다.



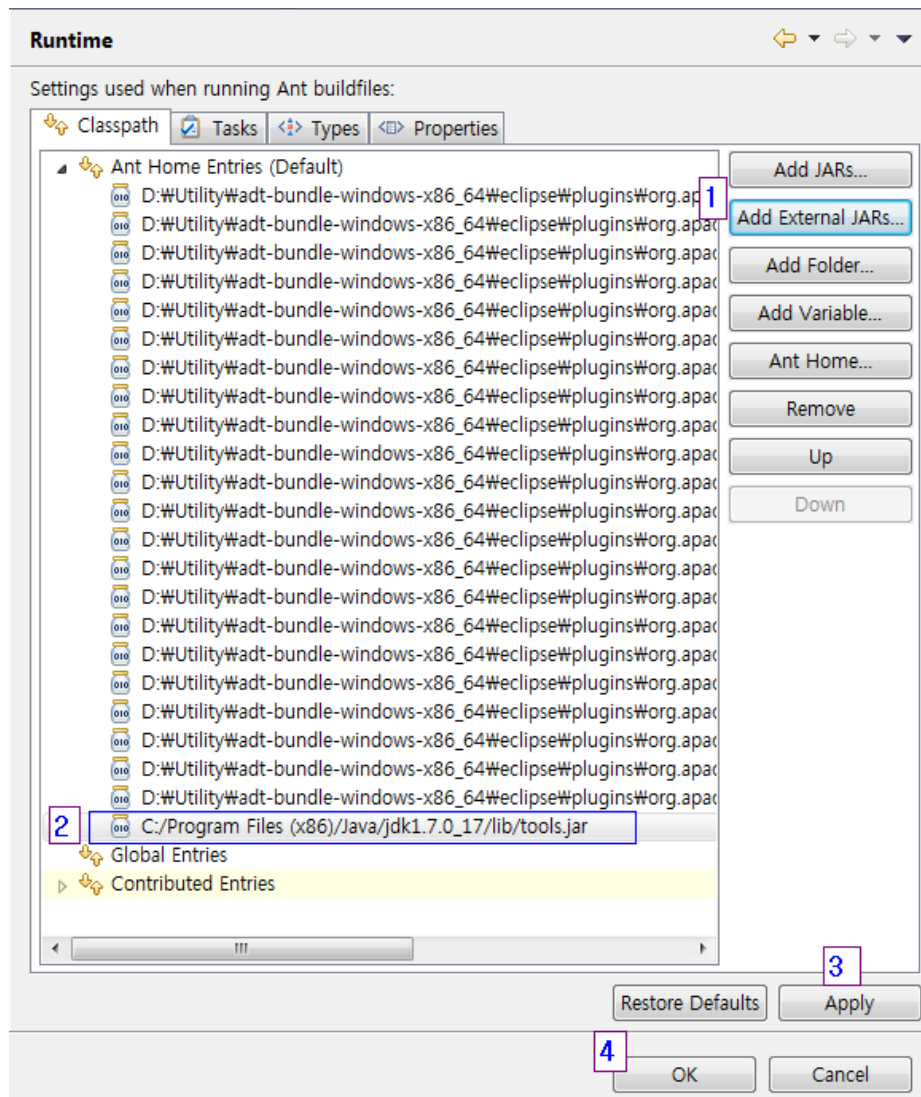
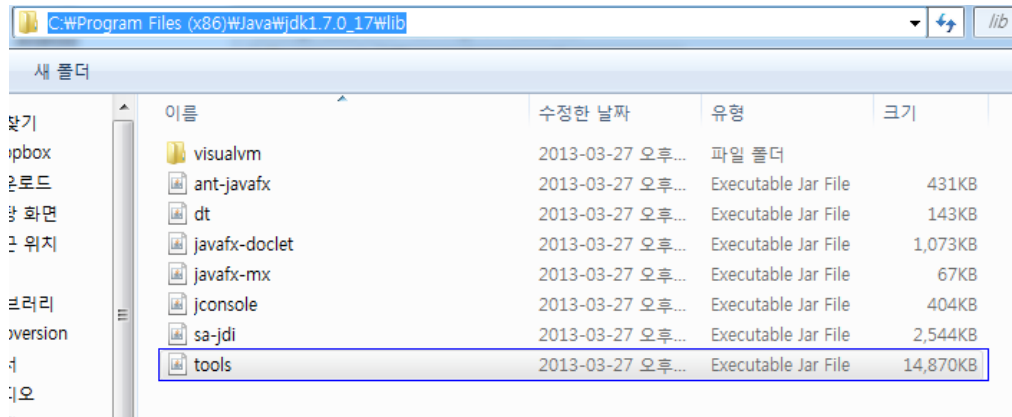
- ③ Ant 툴의 Runtime 옵션에서 tools.jar 를 등록합니다.



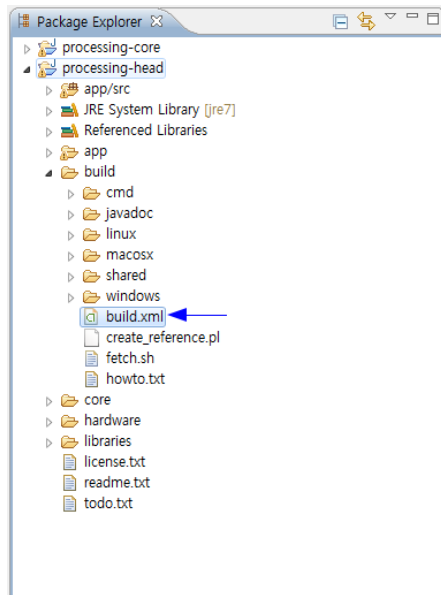
Preferences -> Ant -> Runtime 항목의 Classpath 탭을 클릭합니다.



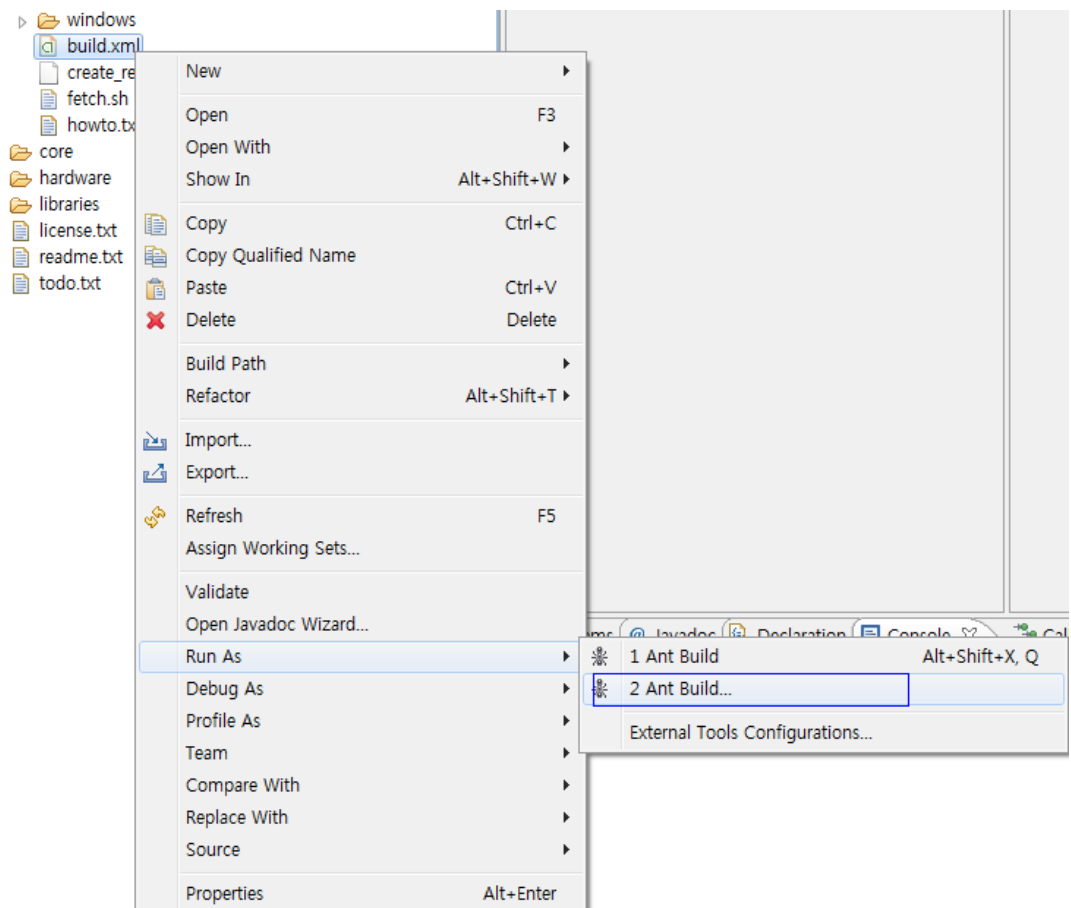
Ant Home Entries(Default)에서 아래 경로의 tools.jar 파일을 Add합니다.



④ Project Explorer에서 아래의 경로에 있는 build.xml 파일을 선택합니다.



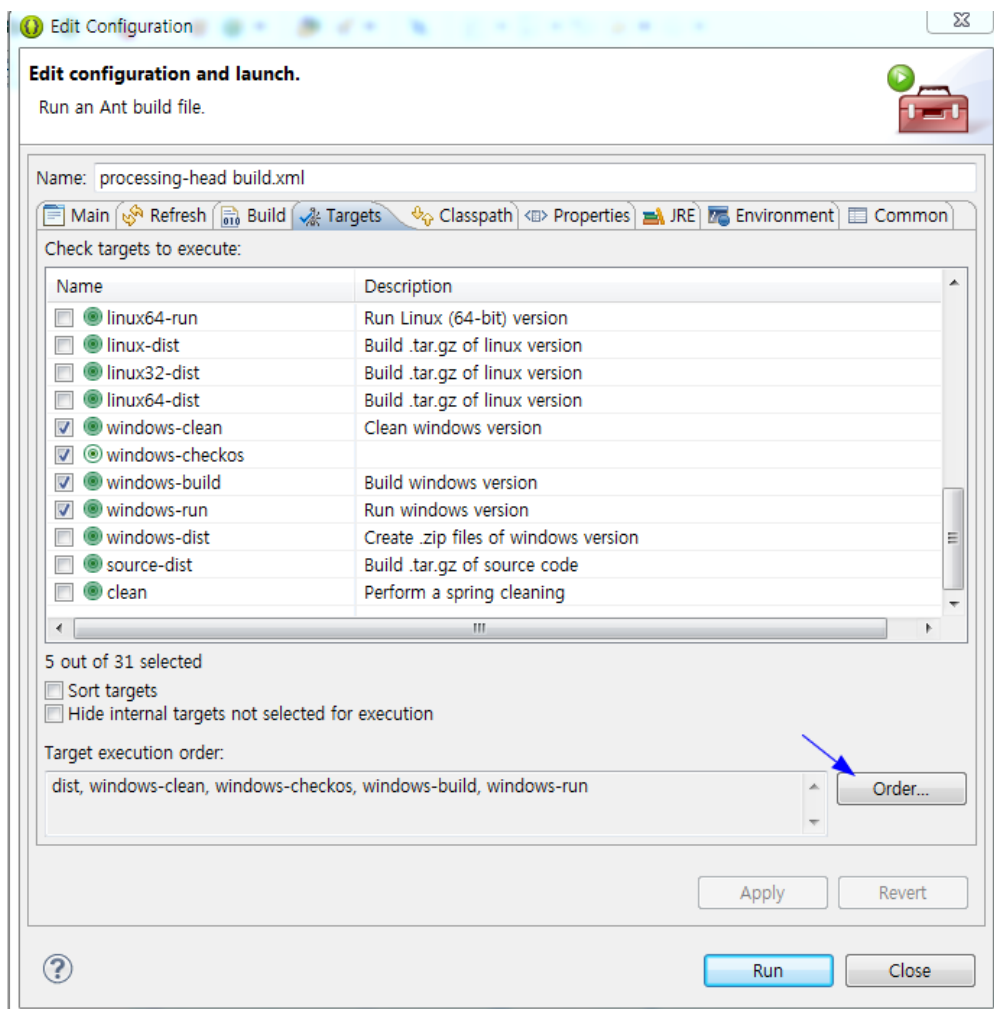
- ⑤ 마우스 오른쪽 버튼을 눌러 팝업 메뉴에서 Run As -> 2 Ant Build...를 선택합니다.



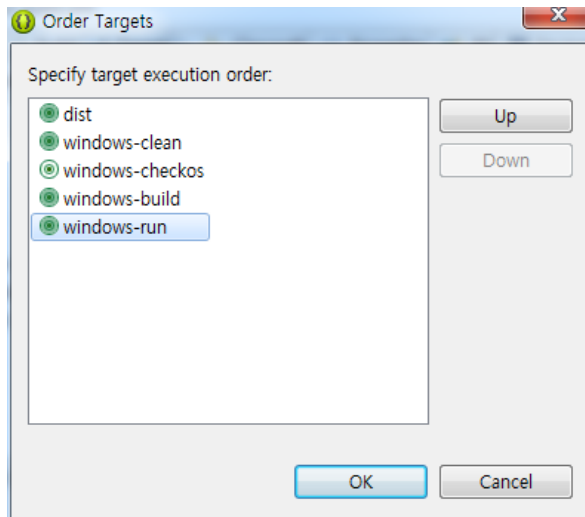
- ⑥ 아래의 빌드 구성요소를 모두 선택합니다.

<input type="checkbox"/> build [default]	Build Arduino.
<input type="checkbox"/> run	Run Arduino.
<input checked="" type="checkbox"/> dist	Build Arduino for distribution.
<input type="checkbox"/> help	Show project help
<input checked="" type="checkbox"/> windows-clean	Clean windows version
<input checked="" type="checkbox"/> windows-checkos	
<input checked="" type="checkbox"/> windows-build	Build windows version
<input checked="" type="checkbox"/> windows-run	Run windows version

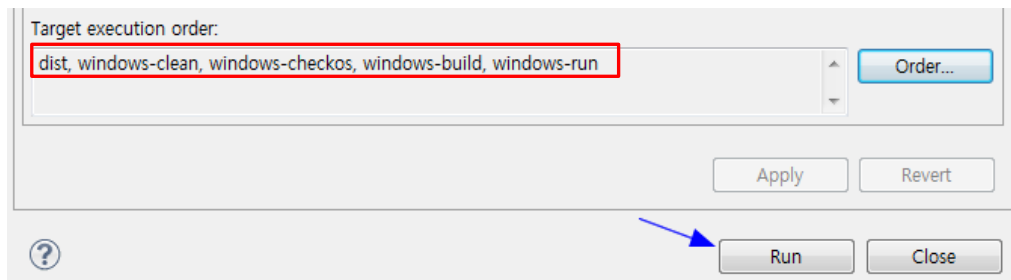
⑦ 빌드 순서를 아래와 같이 조정합니다.



아래의 창이 뜨면 Up/Down 버튼을 통해 순위를 조정합니다.

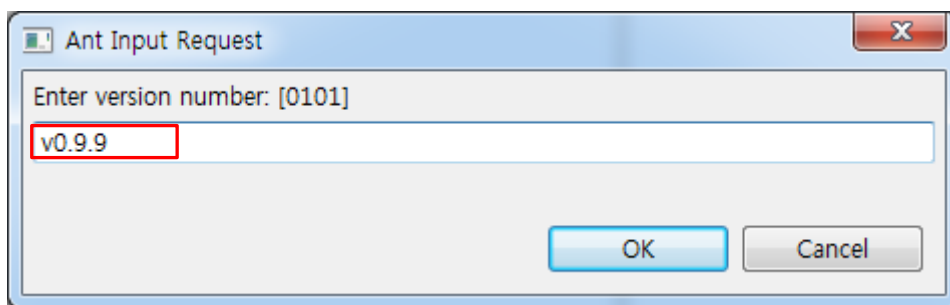


반드시 위의 순서로 합니다. Ok를 누릅니다.



Run을 눌러서 빌드를 시작합니다.

Ant Input Request창이 뜨면 버전을 아래와 같이 입력합니다.



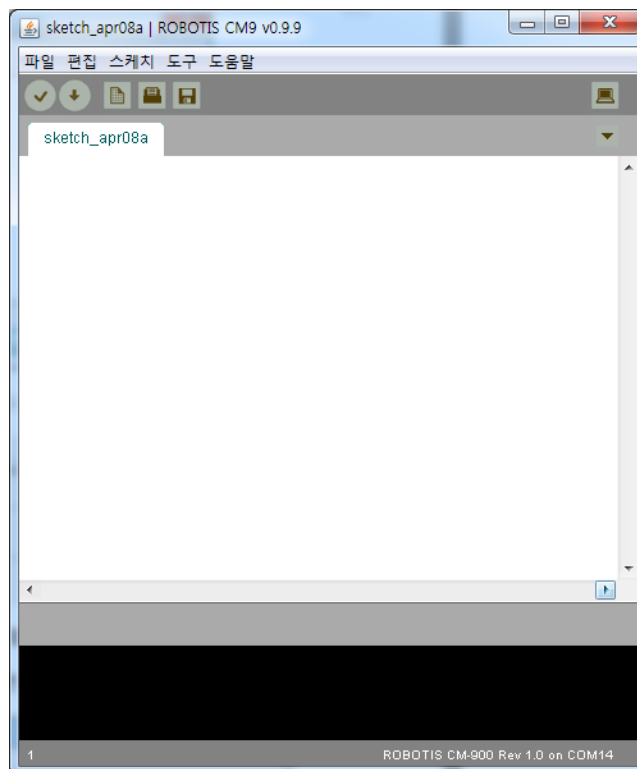
Ok를 누르면 해당 버전으로 빌드가 계속 됩니다. 빌드는 사용하고 있는 PC의 성능에 따라 결정됩니다. 현재 PC에서는 1분 35초가 걸렸습니다.

```

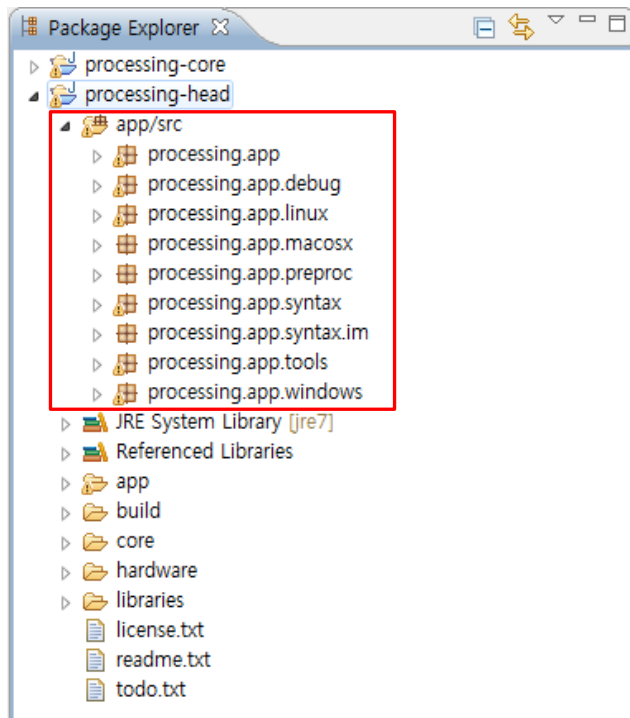
build:
  [jar] Building jar: D:\Java_Workspace\cm-9_ide\processing-head\app\pde.jar
windows-build:
  [copy] Copying 1 file to D:\Java_Workspace\cm-9_ide\processing-head\build\windows\work\lib
  [unzip] Expanding: D:\Java_Workspace\cm-9_ide\processing-head\build\windows\arm_tools.zip into D:\Java_Worksp
assemble:
  [unzip] Expanding: D:\Java_Workspace\cm-9_ide\processing-head\build\shared\reference.zip into D:\Java_Workspa
  [copy] Copying 3 files to D:\Java_Workspace\cm-9_ide\processing-head\build\windows\work
  [launch4j] Compiling resources
  [launch4j] Linking
  [launch4j] Successfully created D:\Java_Workspace\cm-9_ide\processing-head\build\windows\work\ROBOTIS_CM9.exe
windows-run:
BUILD SUCCESSFUL
Total time: 1 minute 35 seconds

```

- ⑧ ROBOTIS CM9이 실행되는 것을 확인합니다.

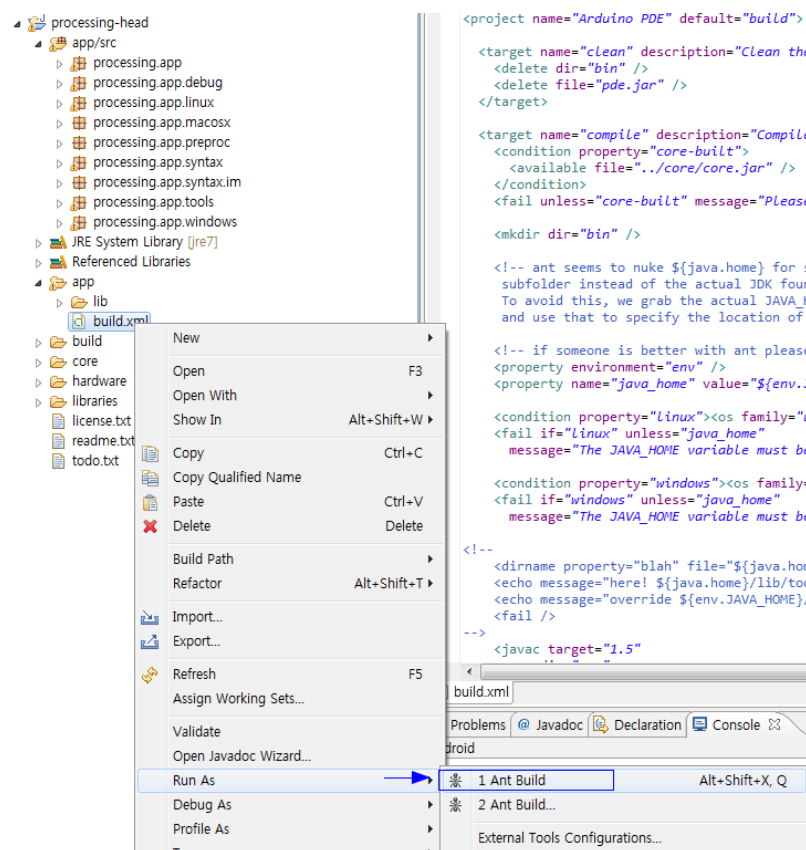


- ⑨ ROBOTIS CM9 IDE의 구현은 모두 java 언어로 되어 있으며 Package Explorer의 아래의 processing-head/app/src 폴더에 모두 구현되어 있습니다.

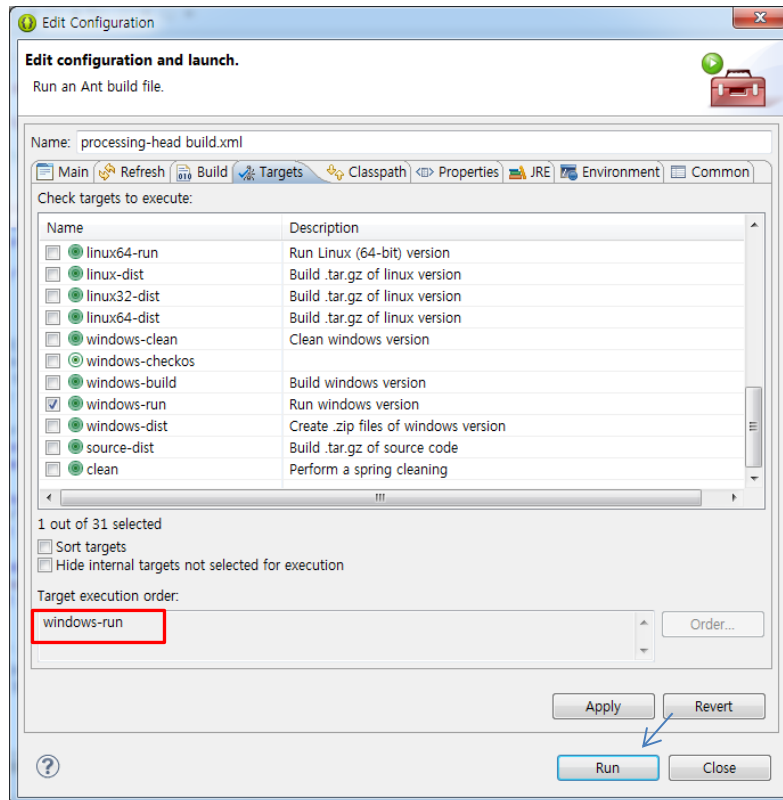


따라서 이 부분을 수정하면 ROBOTIS CM9을 개선 시킬 수 있습니다.

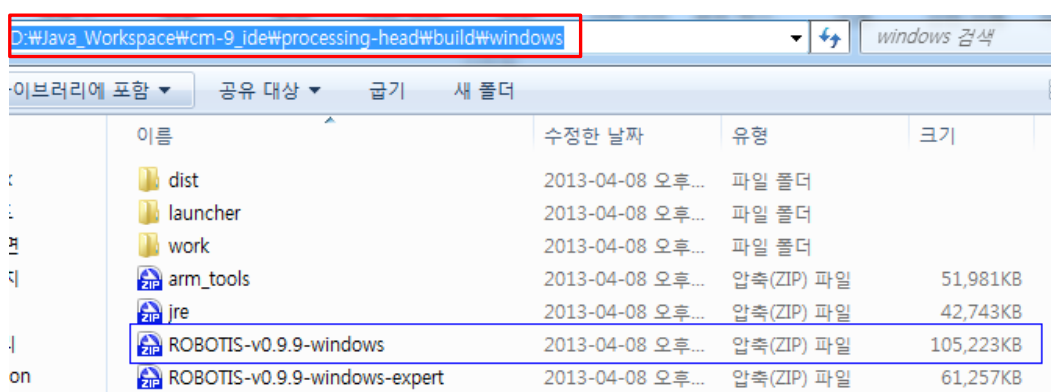
이미 전체 빌드를 수행한 뒤에 processing-head/app/src 부분의 일부를 수정하였다면 반드시 processing-head/app/build.xml 을 통해 core.jar를 새로 빌드해야 합니다.



그리고 processing-head/build/build.xml을 통해 전체 빌드를 수행합니다. 이번에는 Run As...에서 Windows-run 부분만 수행하는 것이 전체 빌드하는 것보다 훨씬 빠르게 빌드 결과를 확인할 수 있습니다.



- ⑩ zip 패키지 파일과 실제 경로는 아래와 같습니다. 아래의 ROBOTIS-v0.9.9-windows.zip파일이 실제 java가 포함된 배포 압축 파일입니다.



실행 파일은 work폴더에 있습니다.

D:\Java_Workspace\cm-9_idet\processing-head\build\windows\work					work 검색
이브러리에 포함 공유 대상 급기 새 폴더					
이름	수정한 날짜	유형	크기		
drivers	2013-04-08 오후...	파일 폴더			
examples	2013-04-08 오후...	파일 폴더			
hardware	2013-04-08 오후...	파일 폴더			
lib	2013-04-08 오후...	파일 폴더			
libraries	2013-04-08 오후...	파일 폴더			
reference	2013-04-08 오후...	파일 폴더			
tools	2013-04-08 오후...	파일 폴더			
cygiconv-2.dll	2013-04-08 오후...	응용 프로그램 확장	947KB		
cygwin1.dll	2013-04-08 오후...	응용 프로그램 확장	1,829KB		
libusb0.dll	2013-04-08 오후...	응용 프로그램 확장	43KB		
revisions	2013-04-08 오후...	텍스트 문서	33KB		
ROBOTIS_CM9	2013-04-08 오후...	응용 프로그램	840KB		
rxTxSerial.dll	2013-04-08 오후...	응용 프로그램 확장	97KB		