

# DK-AVR 개발 환경 구축

---

GRIMNES

2017-06-01 version

# INDEX

---

## 1. DK-AVR Kit 소개

- ◆ DK-AVR Kit 사양
- ◆ DK-AVR Kit Block Diagram
- ◆ DK-AVR Kit 주요 부품 소개

## 2. 컴파일 & 퓨징 환경 구축

- ◆ Toolchain install
- ◆ Fuse bits Setting [터미널환경]
- ◆ ISP 사용방법
- ◆ 아두이노 부트로더 다운로드
- ◆ 이클립스 및 AVR plugin 설치

## 3. RGB LED 점등 프로젝트 실습

- ◆ 터미널 환경
- ◆ 이클립스 환경



GRIMMES

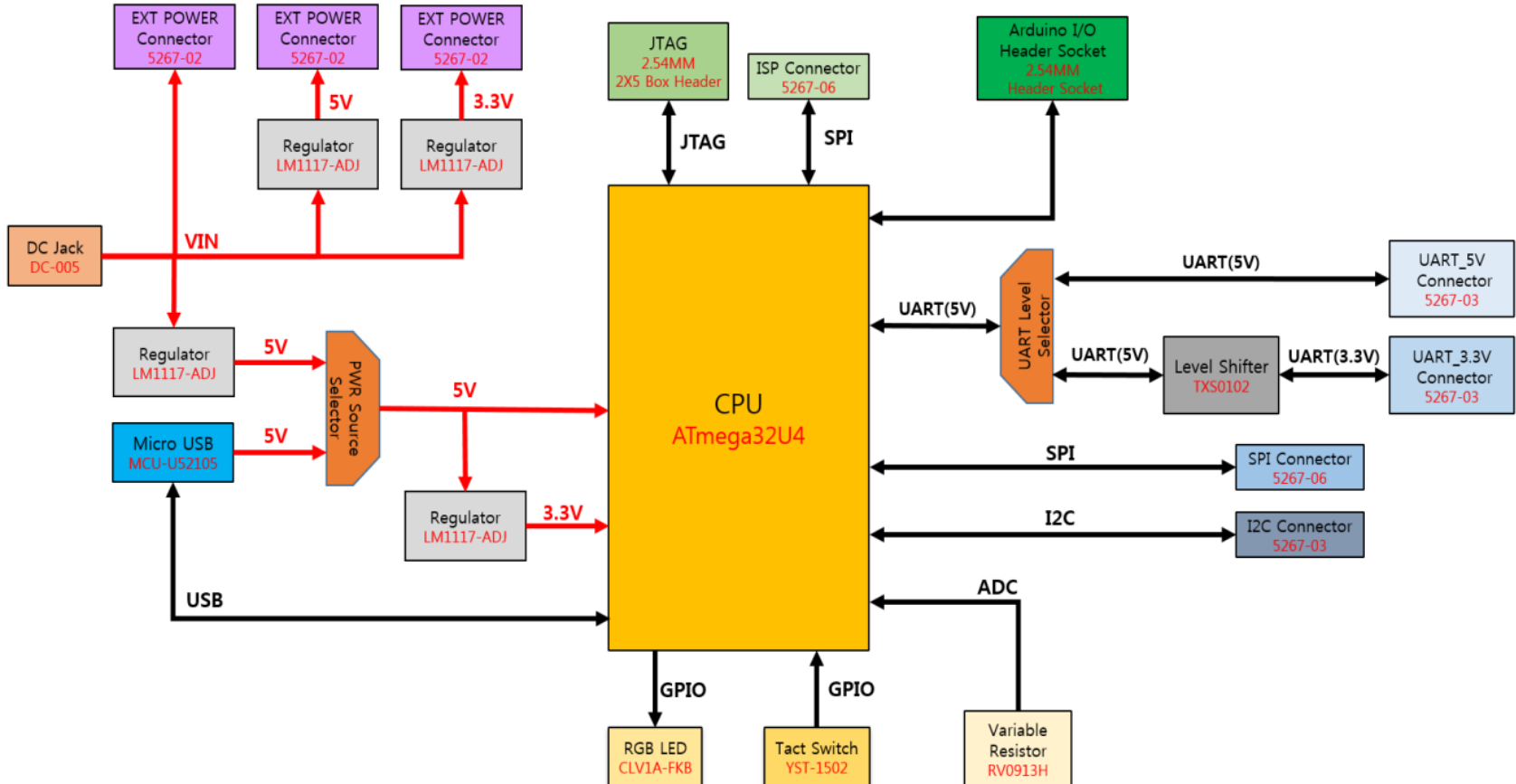
# DK-AVR Kit 소개

- Atmel의 ATmega32U4 사용.
  - <http://www.atmel.com/devices/atmega32u4.aspx>
- 외부전원 : 9V ~ 12V.
- JTAG 연결 커넥터 내장.
- Arduino 커넥터와 호환.
- 사이즈 110mm \* 110mm
- 주요 부품
  - CPU : ATmega32U4-AU

ATmega32U4 사양			
Flash	32KB	Frequency	16MHz
I/O 사양			
UART	2	I2C	1
SPI	1	ISP	1

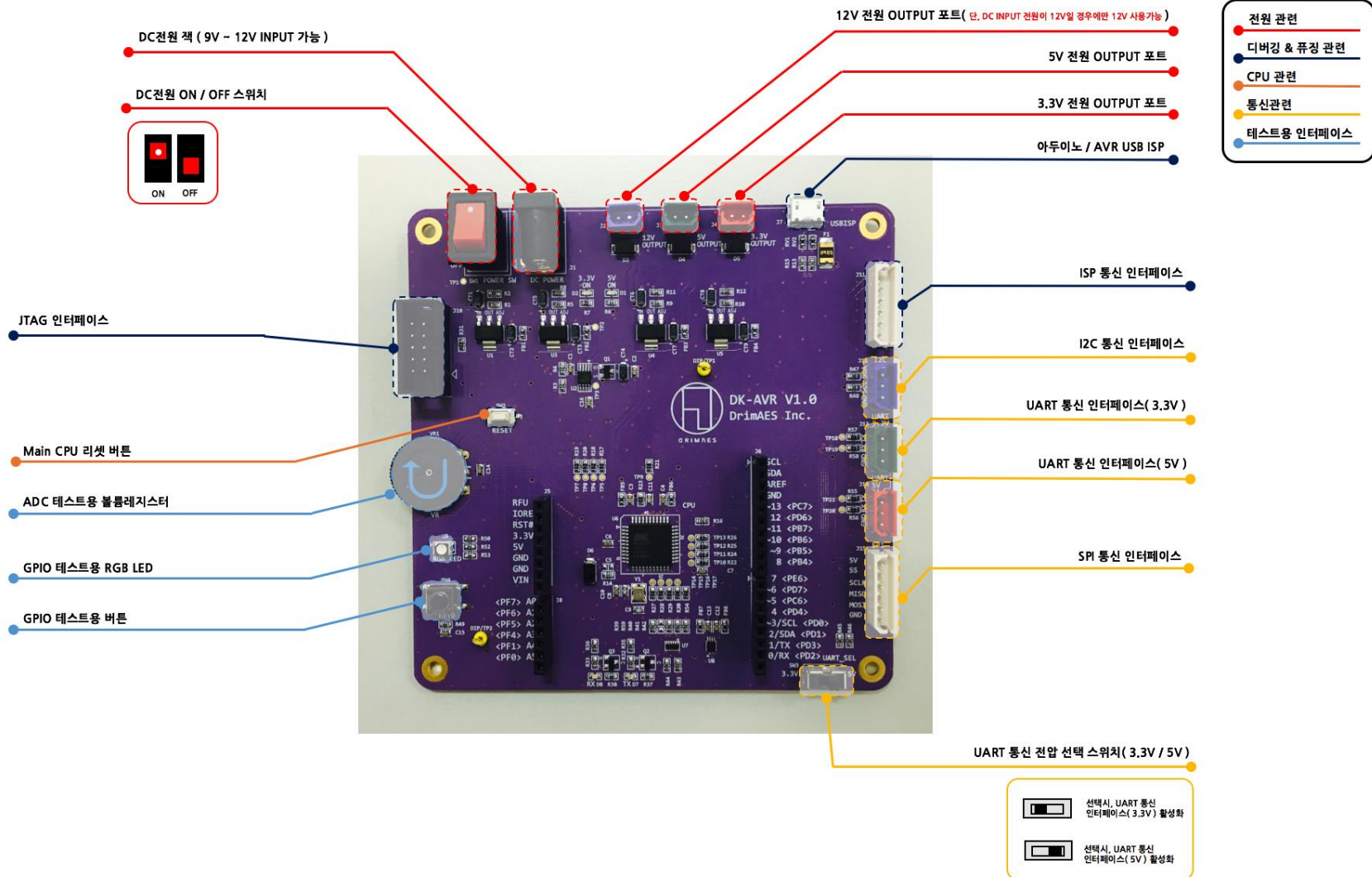
# DK-AVR Kit Block Diagram

DK-AVR Kit 소개



# DK-AVR Kit 주요 부품 소개

DK-AVR Kit 소개



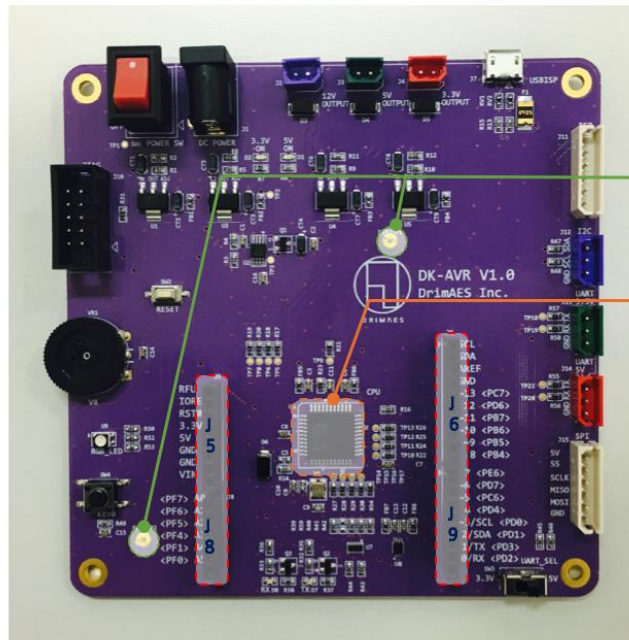
# DK-AVR Kit 주요 부품 소개

DK-AUTOSAR Kit 소개

PIN	PORT	FUNCTION	J5
J5-1	PORTD	N / C	
J5-2	IOREF	IOREF	
J5-3	RESET	RESET	
J5-4	3.3V	3.3V	
J5-5	5V	5V	
J5-6	GND	GND	
J5-7	GND	GND	
J5-8	VIN	VIN	

PIN	PORT	FUNCTION	J8
J8-1	PORTF	ADC7 / TDI / PF7	
J8-2	PORTF	ADC6 / TDO / PF6	
J8-3	PORTF	ADC5 / TDS / PF5	
J8-4	PORTF	ADC4 / TCK / PF4	
J8-5	5V	ADC1 / PF1	



DIP 타입의 GND( H/W 디버깅시 편리하게 사용 가능 )

Main CPU( Atmega32u4-AU )

J6	PIN	PORT	FUNCTION
	J6-1	PORTD	PD0 / INT0 / OCOB / SCL
	J6-2	PORTD	PD1 / INT1 / SDA
	J6-3	AREF	AREF
	J6-4	GND	GND
	J6-5	PORTC	PC7 / ICP3 / CLK0 / OC4A
	J6-6	PORTD	PD6 / T1 / OC4D / ADC9
	J6-7	PORTB	PB7 / PCINT7 / OC1C / OC0A / RTS
	J6-8	PORTB	PB6 / PCINT6 / OC1B / OC4B / ADC13
	J6-9	PORTB	PB5 / PCINT5 / OC1A / OC4B / ADC12
	J6-10	PORTB	PB4 / PCINT4 / ADC11

J9	PIN	PORT	FUNCTION
	J9-1	PORTE	PE6 / AINT0 / INT6
	J9-2	PORTD	PD7 / T0 / OC4D / ADC10
	J9-3	PORTC	PC6 / OC3A / OC4A
	J9-4	PORTD	PD4 / ICP1 / ADC8
	J9-5	PORTD	PD0 / INT0 / OCOB / SCL
	J9-6	PORTD	PD1 / INT1 / SDA
	J9-7	PORTD	PD3 / INT3 / TXD1
	J9-8	PORTD	PD2 / INT2 / RXD1



GRIMES

# 컴파일 & 푸징 환경 구축

> Toolchain  
Install



Step1. AVR 개발환경을 위해서 아래와 같은 toolchain을 설치한다.이때 사용하는 명령어는 아래와 같다.

```
sykang@sykang-DrimAES:~$ sudo apt-get install binutils-avr gcc-avr avr-libc gdb-avr avrdude
```

- binutils-avr : avr용 오브젝트 파일 형식들을 조작하기 위한 프로그램 도구  
모음.(어셈블러, 링커 등)
- gcc-avr : AVR 컴파일러.
- avr-libc : AVR 시스템 라이브러리.
- gdb-avr : AVR 디버거.
- avrdude : 퓨징 툴.



## 컴파일 & 퓨징 환경구축

> Fuse bits  
Setting[ 터미널환경 ]

## Fuse bits란 무엇인가?

- MCU의 환경을 설정하는 부분이다.
- 특정 기능을 하는 회로에 전원을 공급하는 라인을 끊거나, 연결(퓨징)해서 클럭을 조정하거나, 클럭분주, 부가기능 일정 전압이하로 떨어지면 리셋되도록하는 것, ISP 다운로드의 SPI통신 기능등을 설정할 수 있다.
- Extended Fuse, High Fuse, Low Fuse 바이트 모두 3개의 바이트로 구성되어 있다.
- **디폴트로 비트값이 1로 되어 있고 프로그램하면(enabled) 0으로 설정된다.**
- chip erase명령에 의해 영향을 받지 않는다.
- 메모리 Lock비트의 LB1을 0으로 하여 퓨즈 비트를 변경할 수 없도록 할 수 있다.
- 퓨즈 비트를 먼저 설정하고 메모리 Lock비트를 설정해야 한다.
- DK-AVR의 Fuse bits 설정값

Extenede Fuse	0xFB
High Fuse	0x98
Low Fuse	0xFF

# Fuse bits Setting[ 터미널환경 ]

Extended Fuse = 0xFB = 0b1111\_1011

- 아래 표는 Extended Fuse의 각 비트 별 기본 값을 설명한 표이다.
- 각 비트 별 설명은 다음 슬라이드를 참고.

Fuse Low Byte	Bit No	Description	Default Value	
			ATmega16/32U4	ATmega16/32U4RC
—	7	—	1	
—	6	—	1	
—	5	—	1	
—	4	—	1	
HWBE	3	Hardware Boot Enable	0 (programmed)	1 (unprogrammed)
BODLEVEL2	2	Brown-out Detector trigger level	0 (programmed)	
BODLEVEL1	1	Brown-out Detector trigger level	1 (unprogrammed)	
BODLEVEL0	0	Brown-out Detector trigger level	1 (unprogrammed)	

# Fuse bits Setting[ 터미널환경 ]

컴파일 & 퓨징 환경 구축

Extended Fuse = 0xFB = 0b1111\_1011

- HWBE = b1 : Disabled : HWB핀을 사용하여 부트로더영역 으로 들어갈지 어플리케이션(펌웨어) 영역으로 들어갈지 결정한다. 현재는 사용하지 않으므로 Disabled(0b1)로 셋팅 한다.
- BODLEVEL[2...0] = 0b011 – BOD 기능의 사용여부 결정 및 사용할 때에 전압 범위를 설정한다.

※ BOD = Brown-Out Detection / MCU의 정상 동작 전압 범위 밖에서는 MCU의 리셋핀을

확실히

BODLEVEL 2..0 Fuses	Min. V <sub>BOT</sub>	Typ. V <sub>BOT</sub>	Max. V <sub>BOT</sub>	Units
111	BOD Disabled			V
110	1.8	2.0	2.2	
101	2.0	2.2	2.4	
100	2.2	2.4	2.6	
011	2.4	2.6	2.8	
010	3.2	3.4	3.6	
001	3.3	3.5	3.7	
000	4.0	4.3	4.5	

# Fuse bits Setting[ 터미널환경 ]

High Fuse = 0x98 = 0b1001\_1000

- 아래 표는 High Fuse의 각 비트 별 기본 값을 설명한 표이다.
- 각 비트 별 설명은 다음 슬라이드를 참고.

Fuse High Byte	Bit No	Description	Default Value
OCDEN	7	Enable OCD	1 (unprogrammed, OCD disabled)
JTAGEN	6	Enable JTAG	0 (programmed, JTAG enabled)
SPIEN	5	Enable Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
WDTON	4	Watchdog Timer always on	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM preserved)
BOOTSZ1	2	Select Boot Size	0 (programmed)
BOOTSZ0	1	Select Boot Size	0 (programmed)
BOOTRST	0	Select Bootloader Address as Reset Vector	1 (unprogrammed, Reset vector @0x0000)

High Fuse = 0x98 = 0b1001\_1000

- OCDEN = 1(Disabled) : OCD 사용 여부를 설정한다. 사용하지 않는다.
- JTAGEN = 0(Enabled) : JTAG 사용 여부를 설정한다. JTAG를 사용하도록 설정한다.
- SPIEN = 0(Enabled) : SPI 사용 여부를 설정한다.
- WDTON = 1(Disabled) : Watchdog timer를 항상 사용할지에 대한 설정이다.
- EESAVE = 1(Disabled) : 칩 erase시에 EEPROM 영역을 보호할지를 설정한다.
- BOOTSZ[1...0] = 0b00 : 부트로더영역과 어플리케이션 영역의 사이즈를 설정한다.  
설정된 사이즈는 다음 슬라이드를 참고.
- BOOTRST = 0(Enabled) : 전원 인가시에 Reset Vector를 선택하는 비트이다.

# Fuse bits Setting[ 터미널환경 ]

컴파일 & 퓨징 환경 구축

High Fuse = **0x98** = 0b1001\_1000

Device	BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Section	Boot Loader Flash Section	End Application Section	Boot Reset Address (Start Boot Loader Section)
ATmega32U4	1	1	256 words	4	0x0000 - 0x3EFF	0x3F00 - 0x3FFF	0x3EFF	0x3F00
	1	0	512 words	8	0x0000 - 0x3DFF	0x3E00 - 0x3FFF	0x3DFF	0x3E00
	0	1	1024 words	16	0x0000 - 0x3BFF	0x3C00 - 0x3FFF	0x3BFF	0x3C00
	0	0	2048 words	32	0x0000 - 0x37FF	0x3800 - 0x3FFF	0x37FF	0x3800
ATmega16U4	1	1	256 words	4	0x0000 - 0x1EFF	0x1F00 - 0x1FFF	0x1EFF	0x1F00
	1	0	512 words	8	0x0000 - 0x1DFF	0x1E00 - 0x1FFF	0x1DFF	0x1E00
	0	1	1024 words	16	0x0000 - 0x1BFF	0x1C00 - 0x1FFF	0x1BFF	0x1C00
	0	0	2048 words	32	0x0000 - 0x17FF	0x1800 - 0x1FFF	0x17FF	0x1800



# Fuse bits Setting[ 터미널환경 ]

Low Fuse = 0xFF = 0b1111\_1111

- 아래 표는 Low Fuse의 각 비트 별 기본 값을 설명한 표이다.
- 각 비트 별 설명은 다음 슬라이드를 참고.

Fuse Low Byte	Bit Nr	Description	Default Value	
			ATmega16U4/32U4	ATmega16U4RC/32U4RC
CKDIV8	7	Divide clock by 8	0 (programmed)	
CKOUT	6	Clock output	1 (unprogrammed)	
SUT1	5	Select start-up time	0 (programmed)	
SUT0	4	Select start-up time	1 (unprogrammed)	
CKSEL3	3	Select Clock source	1 (unprogrammed)	0 (programmed)
CKSEL2	2	Select Clock source	1 (unprogrammed)	0 (programmed)
CKSEL1	1	Select Clock source	1 (unprogrammed)	1 (unprogrammed)
CKSEL0	0	Select Clock source	0 (programmed)	0 (programmed)

Low Fuse = 0xFF = 0b1111\_1111

- CKDIV8 : b1 : 클럭을 1/8로 분주시켜준다.
- CKOUT : b1 : 클럭을 AVR핀 중 CKOUT기능이 있는 핀으로 출력.
- SUT[1:0] : b11 : 리셋 후의 지연시간 결정한다. 설정한 지연시간값은 다음 슬라이드를 참고.
- CKSEL[3:0] : b1111 : 클럭소스를 설정한다.

설정값에 대한 자세한 설명은 다음 슬라이드를 참고.

# Fuse bits Setting[ 터미널환경 ]

컴파일 & 퓨징 환경 구축

Low Fuse = 0xFF = 0b1111\_1111

CKSEL[0] = 0b1 / SUT[1:0] = 0b11

Oscillator Source / Power Conditions	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V <sub>CC</sub> = 5.0V)	CKSEL0	SUT1..0
Ceramic resonator, fast rising power	258CK	14CK + 4.1ms	0	00
Ceramic resonator, slowly rising power	258CK	14CK + 65ms	0	01
Ceramic resonator, BOD enabled	1K CK	14CK	0	10
Ceramic resonator, fast rising power	1K CK	14CK + 4.1ms	0	11
Ceramic resonator, slowly rising power	1K CK	14CK + 65ms	1	00
Crystal Oscillator, BOD enabled	16K CK	14CK	1	01
Crystal Oscillator, fast rising power	16K CK	14CK + 4.1ms	1	10
Crystal Oscillator, slowly rising power	16K CK	14CK + 65ms	1	11

# Fuse bits Setting[ 터미널환경 ]

컴파일 & 퓨징 환경 구축

Low Fuse = 0xFF = 0b1111\_1111

CKSEL[3:0] = 0b1111

Device Clocking Option	CKSEL[3:0] (or EXCKSEL[3:0])
Low Power Crystal Oscillator	1111 - 1000
Reserved	0111 - 0110
Low Frequency Crystal Oscillator	0101 - 0100
Reserved	0011
Calibrated Internal RC Oscillator	0010
External Clock	0000
Reserved	0001

# Fuse bits Setting[ 터미널환경 ]

컴파일 & 퓨징 환경 구축

\* Fuse bits Setting을 위해서는 **AVRDUDUE**를 사용해야한다.

## AVRDUDE 란?

ISP 기술을 이용해서 AVR 플래쉬메모리, EEPROM등에 데이터를 write / read / verify 가능하도록 하는 유틸리티이다.

## AVRDUDE 옵션

- p 옵션** partnumber로 칩명을 입력( ex. -p m32u4 )
- c 옵션** programmer로 programmer의 타입을 입력( ex. -c ispavr2 )
- P 옵션** Port로 사용하는 programmer의 포트를 입력( ex. -P /dev/ttyUSB0 )
- b 옵션** baudrate로 사용할 통신 속도를 입력( ex. -b 115200 )
- t 옵션** 터미널 모드로 진입,( ex. -t )
- U 옵션** 메모리 관련 옵션 :: -U<memtype>:**r|w|v**<filename>[:format]

**memtype**

- flash이면 플래쉬메모리 관련된 처리
- lfuse,hfuse,efuse는 퓨즈비트 관련된 처리
- EEPROM은 EEPROM관련된 처리

**r|w|v** r : read, w : write, v : verify

**filename** hex파일 이름 또는 바이트값

**format**

- i 이면 hex파일
- m이면 바이트값

### 예시 퓨즈비트 관련된 레지스터 값 다운로드

```
avrdude -p m32u4 -c avrispv2 -P /dev/ttyUSB0  
-b 115200 -u  
-U lfuse:w:0xff:m
```

Atmega32u4에 ttyUSB0로 잡힌 ISP를 이용해서 115200의 속도로 Low fuse 레지스터를 0xff로 write 하라는 의미.

### 예시 플래쉬 메모리에 hex파일을 다운로드

```
avrdude -p atmega328p -P com3 -c avrispv2  
-U flash:w:drimaes.hex:i
```

Atmega328p에 com3으로 잡힌 ISP를 이용해서 플래쉬메모리에 'drimaes.hex' 파일을 write 하라는 의미.

## Fuse bits Setting 방법

Step1. fuse bits 세팅 모드로 들어가기위해서 아래와 같은 명령어를 터미널에 입력한다.

```
josh$ avrdude -c avrisp2 -p m32u4 -P /dev/tty.SLAB_USBtoUART -t
```

- AVR 퓨징 시 사용되는 ISP 종류를 입력한다.
- 사용하는 AVR 칩 이름을 입력한다.( 만약, Atmega328p를 사용하면, m328p를 입력 )
- Ubuntu에 인식된 사용할 ISP의 경로를 입력한다. ( 만약, ttyUSB0으로 잡혔다면, /dev/ttyUSB0으로 입력 )

# Fuse bits Setting[ 터미널환경 ]

컴파일 & 퓨징 환경 구축

Step2. Step1이 정상동작하면 아래와 같은 화면이 뜨는것을 볼 수 있다.

```
avrdude: AVR device initialized and ready to accept instructions

Reading | ##### | 100% 0.01s

avrdude: Device signature = 0x1e9587
avrdude> █
```

Step3. 명령어 입력창에 **help** 를 입력하면 아래의 이미지처럼 사용 가능한 명령어가 출력된다.( \* 여기서 우리는 read, write, erase를 주로 사용한다. )

```
avrdude> help
>>> help
Valid commands:

dump      : dump memory : dump <memtype> <addr> <N-Bytes>
read      : alias for dump
write     : write memory : write <memtype> <addr> <b1> <b2> ... <bN>
erase     : perform a chip erase
sig       : display device signature bytes
part      : display the current part information
send      : send a raw command : send <b1> <b2> <b3> <b4>
parms     : display adjustable parameters (STK500 only)
vtarg     : set <V[target]> (STK500 only)
varef     : set <V[aref]> (STK500 only)
fosc      : set <oscillator frequency> (STK500 only)
sck       : set <SCK period> (STK500 only)
spi       : enter direct SPI mode
pgm       : return to programming mode
verbose   : change verbosity
help      : help
?         : help
quit      : quit

Use the 'part' command to display valid memory types for use with the
'dump' and 'write' commands.

avrdude>
```

# Fuse bits Setting[ 터미널환경 ]

컴파일 & 퓨징 환경 구축

Step4. 현재 fuse bit를 읽고 싶으면 아래와 같이 입력하면 fuse bit값을 읽을 수 있다.

( \* Ex, read efuse : extend fuse bit를 읽어라 )

```
avrdude> read efuse  efuse -> Extend fuse bit
>>> read efuse
0000 cb |. |

avrdude> read hfuse  hfuse -> High fuse bit
>>> read hfuse
0000 98 |. |

avrdude> read lfuse  lfuse -> Low fuse bit
>>> read lfuse
0000 ff |. |
```

Step5. 초기의 AVR 칩은 Lock bit 설정으로 fuse bit 설정이 안된다, 그래서 먼저 erase 명령으로 Lock bit 설정을 해제해준다.

```
>>> erase
avrdude: erasing chip
```



Step6. fuse bits 를 세팅하기위해서는 아래와 같이 입력하면 된다.

( \* Ex, write efuse 0 0xcb : extend fuse bit에 0xcb를 써라)

```
avrdude> write efuse 0 0xcb
>>> write efuse 0 0xcb

avrdude> write hfuse 0 0xd8
>>> write hfuse 0 0xd8

avrdude> write lfuse 0 0xff
>>> write lfuse 0 0xff
```

※ 한줄의 명령어로도 퓨즈비트 셋팅이 가능하다.

```
Avrdude -p m32u4 -c avrispv2 -P /dev/ttyUSB0 -b 115200 -u
-U lfuse:w:0xff:m
-U hfuse:w:0x98:m
-U efuse:w:0xfb:m
```

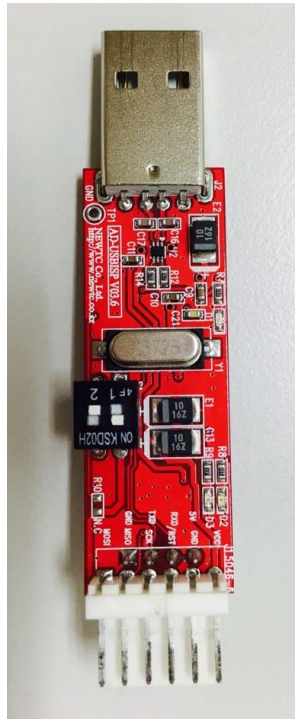


# 컴파일 & 푸징 환경구축

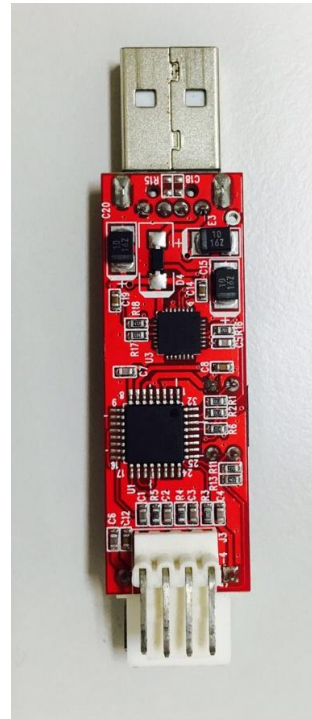
## > ISP 사용방법

## ISP 란?

In System Programming 의 약자로 Atmel AVR의 내부 플래쉬, EEPROM에 데이터를 다운로드 할 수 있게 해주는 툴을 의미한다. 쉽게 말해서 작성한 펌웨어 결과파일(.HEX)를 AVR 내부 플래쉬 메모리에 다운로드 해주는 역할을 한다.



ISP 앞면 (ISP)



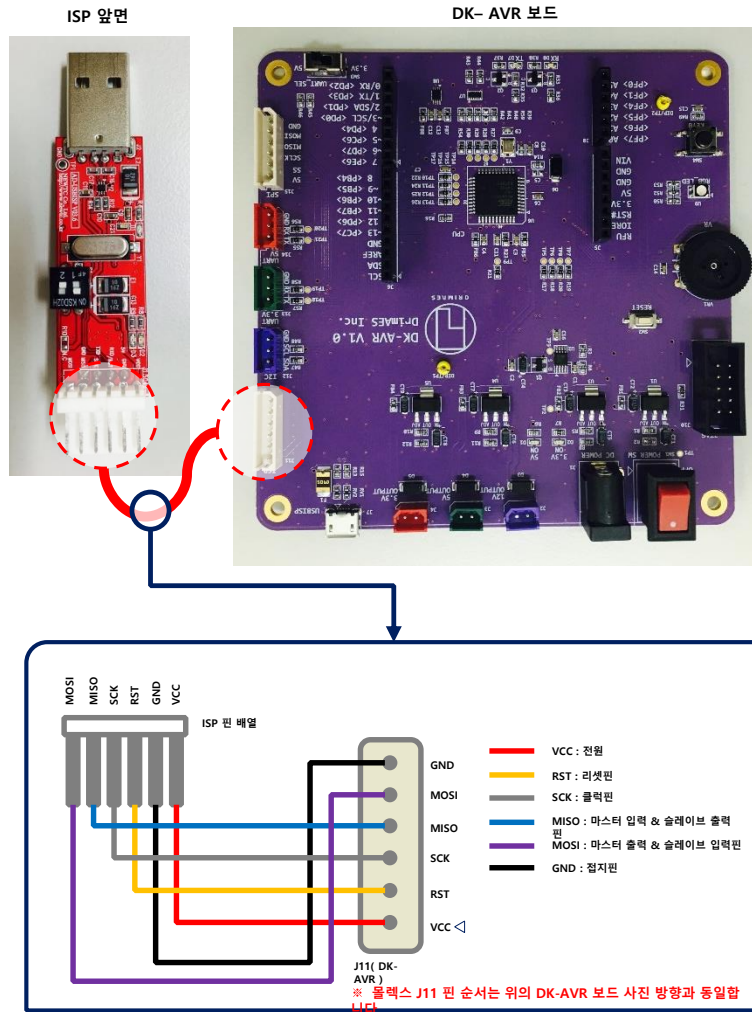
ISP 뒷면 (UART)

본 매뉴얼에서 사용하는 ISP는 **ISP 기능** 뿐만 아니라 **UART 통신 기능**으로도 사용 가능하다. 2가지의 기능을 구분하기 위해 **왼쪽 사진을 ISP 앞면, 오른쪽 사진을 ISP 뒷면**으로 언급 하겠습니다.

※ 본 매뉴얼에서 사용한 ISP는 뉴티씨라는 회사에서 만든 ISP를 사용한다.  
( [http://www.newtc.co.kr/dpsshop/shop/item.php?it\\_id=1314600027](http://www.newtc.co.kr/dpsshop/shop/item.php?it_id=1314600027) )

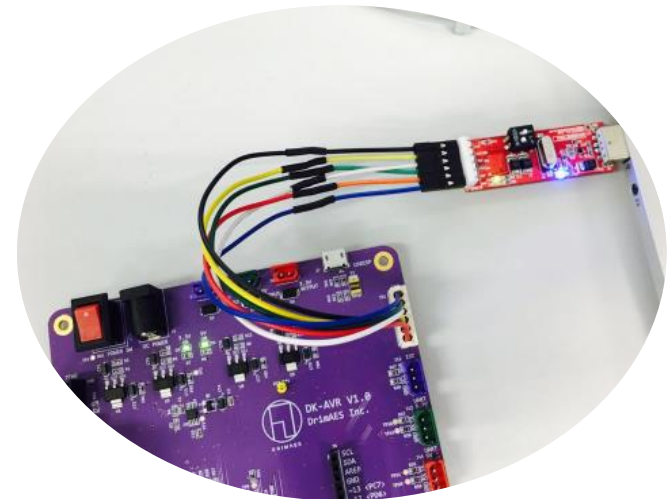
# ISP 사용방법

컴파일 & 퓨징 환경 구축



ISP 장비 <-> DK-AVR ISP 포트 핀 연결도

## ISP기능 사용방법

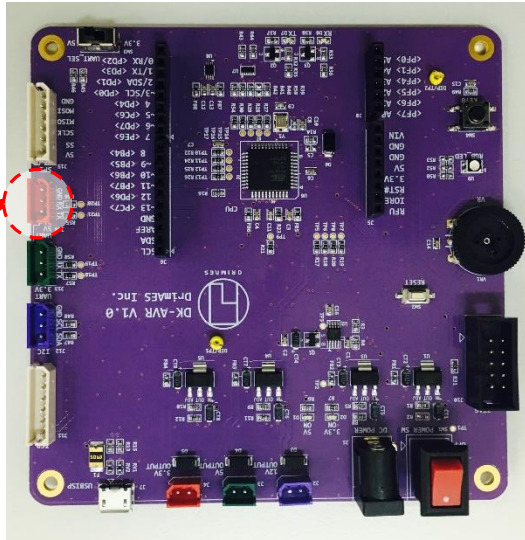


실제 사용 예시

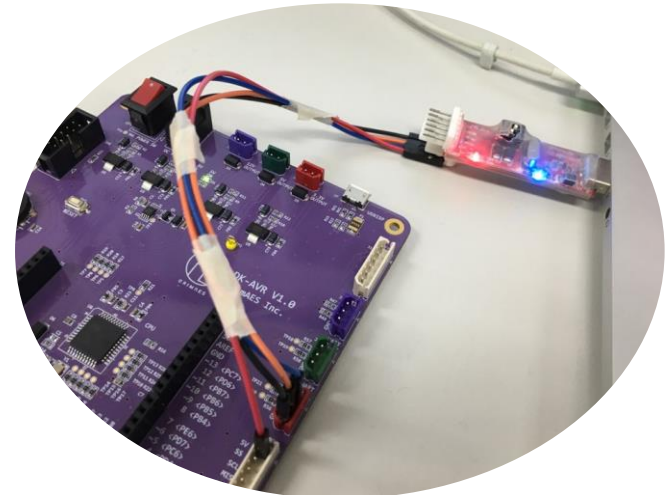
ISP 뒷면



DK- AVR 보드

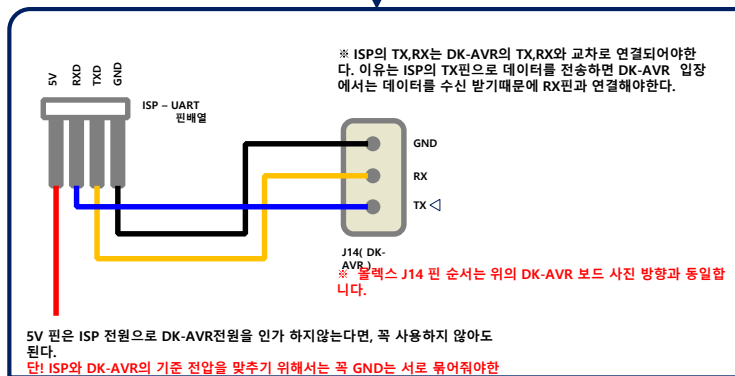


ISP에 있는 UART 사용방



실제 사용 예시

\* ISP 전원으로 DK-AVR보드에 전원을 인가하여 ISP의 5V 핀을 사용



ISP(UART) 장비 <-> DK-AVR UART 포트 핀 연결도

## ISP에 있는 UART 사용방

Step1. 아래의 명령어로 minicom을 설치한다.(minicom은 우분투에서 사용되는 시리얼 통신 프로그램이다)

```
$sudo apt-get install minicom
```

```
josh@josh-System-Product-Name:~$ sudo apt-get install minicom
```

Step2. 아래의 명령어로 minicom을 실행해서 설정모드로 들어간다.  
\$sudo minicom -s

```
josh@josh-System-Product-Name:~$ sudo minicom -s
```

Step3. Serial port setup을 선택한다.

```
josh@josh-System-Product-Name:~$ sudo minicom -s

+-----[configuration]-----+
| Filenames and paths          |
| File transfer protocols      |
| Serial port setup           |
| Modem and dialing            |
| Screen and keyboard          |
| Save setup as dfl             |
| Save setup as..              |
| Exit                          |
| Exit from Minicom            |
+-----+


```

Step4. 아래의 2가지를 세팅한다.

```
josh@josh-System-Product-Name:~$ sudo minicom -s

A - Serial Device      : /dev/ttyUSB0
B - Lockfile Location  : /var/lock
C - Callin Program     :
D - Callout Program    :
E - Bps/Par/Bits       : 9600 8N1
F - Hardware Flow Control : No
G - Software Flow Control : No

Change which setting?

Screen and keyboard
Save setup as dfl
Save setup as..
Exit
Exit from Minicom


```

인식된 ISP의 포트를 입력한다  
만약, ttyUSB1으로 잡혔다면,  
/dev/ttyUSB1으로 입력하면 된다.

UART 통신속도를 선택한다.  
본 매뉴얼에 사용된 펌웨어의 UART  
통신 속도는 9600이다.

※ 본 매뉴얼에서 사용된 UART 펌웨어 예제는 통신속도는 9600이며,  
수신받은 데이터를 그대로 다시 전송 ( 에코 ) 하는 기능을 가진 예제이다.

## 44

```
josh@josh-System-Product-Name: ~  
  
+-----[configuration]-----+  
| Filenames and paths          |  
| File transfer protocols      |  
| Serial port setup           |  
| Modem and dialing           |  
| Screen and keyboard         |  
| Save setup as dfl            |  
| Save setup as..             |  
| Exit                         |  
| Exit from Minicom           |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
|
```

```
x - □ josh@josh-System-Product-Name: ~  
Welcome to minicom 2.7  
  
OPTIONS: I18n  
Compiled on Jan 1 2014, 17:13:19.  
Port /dev/ttyUSB0, 15:19:52  
  
Press CTRL-A Z for help on special keys  
  
DRIMAES AVR UART TEST
```



GRIMNES

## 컴파일 & 푸징 환경구축


> 아두이노 부드로더  
다운로드



Step1. 아두이노 홈페이지에서 Linux용 IDE를 다운받는다.

( 다운로드 경로 : <https://www.arduino.cc/en/Main/Software> )

## Download the Arduino Software



### ARDUINO 1.6.11

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

**Windows** Installer  
Windows ZIP file for non admin install

Mac OS X 10.7 Lion or newer

**Linux** 32 bits  
**Linux** 64 bits  
**Linux** ARM (experimental)

Release Notes  
Source Code  
Checksums (sha512)

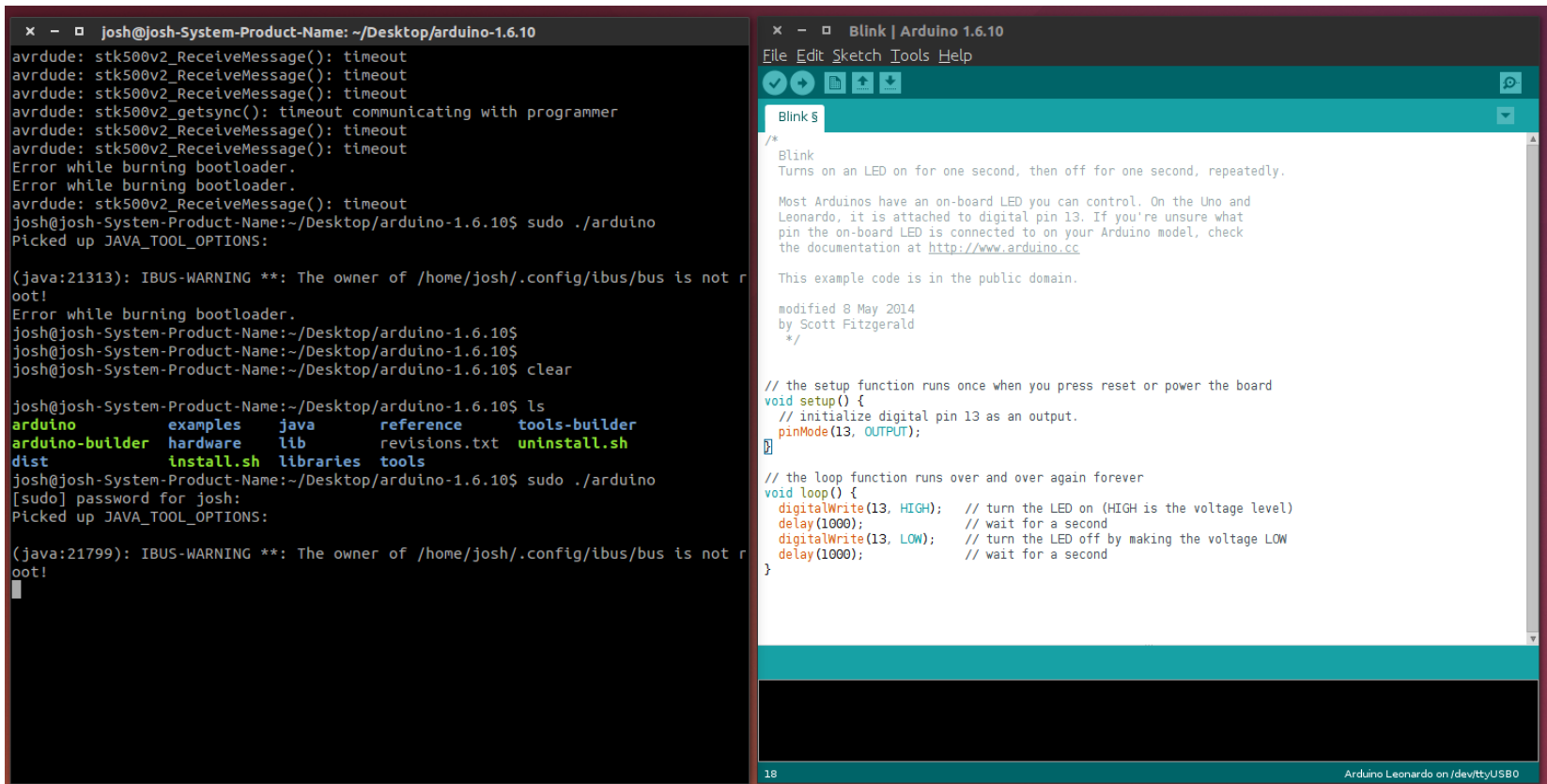
CONNECT, COLLABORATE, CREATE. [Learn more about the Create platform.](#)

**Try out the new  
Arduino Web Editor**

Step2. 아두이노 IDE를 터미널에서 명령어로 실행한다.

터미널에서 압축을 푼 경로에 들어가면 **arduino**라는 실행파일이 있는데 이를 아래와 같은 명령어로 실행한다.

**\$ sudo ./arduino**



```
josh@josh-System-Product-Name: ~/Desktop/arduino-1.6.10
avrdude: stk500v2_ReceiveMessage(): timeout
avrdude: stk500v2_ReceiveMessage(): timeout
avrdude: stk500v2_ReceiveMessage(): timeout
avrdude: stk500v2_getsync(): timeout communicating with programmer
avrdude: stk500v2_ReceiveMessage(): timeout
avrdude: stk500v2_ReceiveMessage(): timeout
Error while burning bootloader.
Error while burning bootloader.
avrdude: stk500v2_ReceiveMessage(): timeout
josh@josh-System-Product-Name:~/Desktop/arduino-1.6.10$ sudo ./arduino
Picked up JAVA_TOOL_OPTIONS:

(java:21313): IBUS-WARNING **: The owner of /home/josh/.config/ibus/bus is not root!
Error while burning bootloader.
josh@josh-System-Product-Name:~/Desktop/arduino-1.6.10$ ls
arduino  examples  java  reference  tools-builder
arduino-builder  hardware  lib  revisions.txt  uninstall.sh
dist      install.sh  libraries  tools
josh@josh-System-Product-Name:~/Desktop/arduino-1.6.10$ sudo ./arduino
[sudo] password for josh:
Picked up JAVA_TOOL_OPTIONS:

(java:21799): IBUS-WARNING **: The owner of /home/josh/.config/ibus/bus is not root!
```

```
Blink | Arduino 1.6.10
File Edit Sketch Tools Help

Blink $
/*
 * Blink
 * Turns on an LED on for one second, then off for one second, repeatedly.
 *
 * Most Arduinos have an on-board LED you can control. On the Uno and Leonardo, it is attached to digital pin 13. If you're unsure what pin the on-board LED is connected to on your Arduino model, check the documentation at http://www.arduino.cc
 *
 * This example code is in the public domain.
 *
 * modified 8 May 2014
 * by Scott Fitzgerald
 */

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

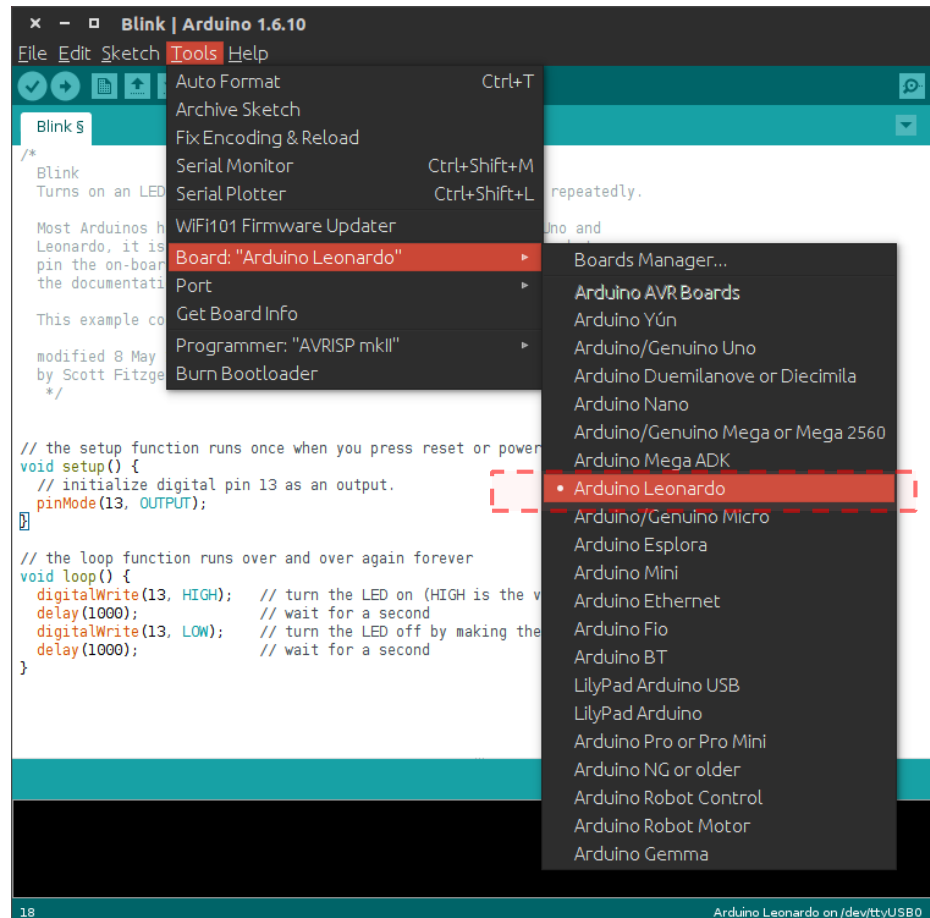
// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);            // wait for a second
  digitalWrite(13, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);            // wait for a second
}
```

( 좌, 터미널에서 아두이노 실행 명령어 입력

우, 터미널에서 명령어 입력 후 실행된 아두이노 IDE 모습

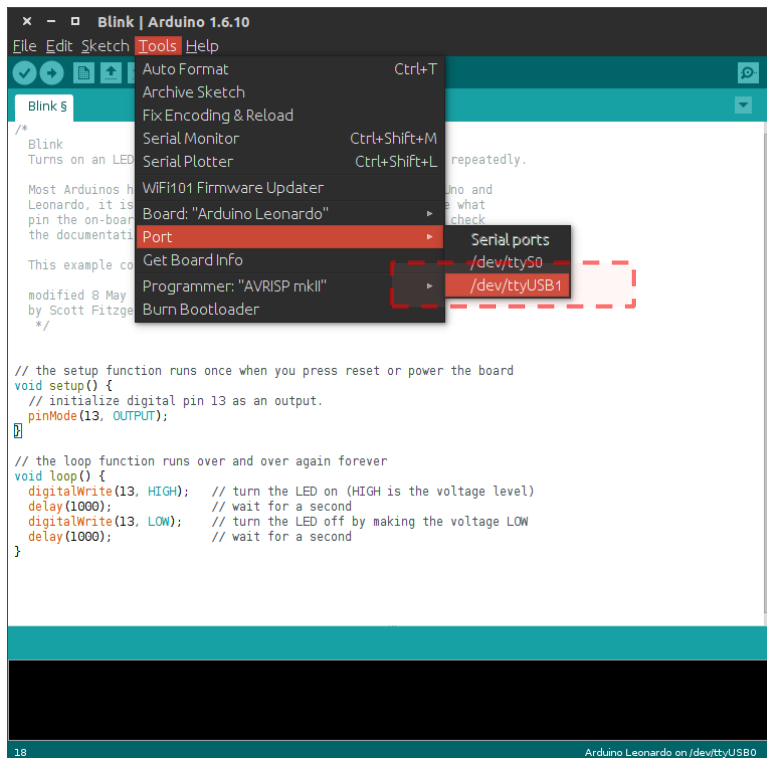
Step3. 다운로드하려는 아두이노 부트로더(Leonardo or Micro)를 선택한다.

( **Tools -> Board -> Arduino Leonardo** )



## Step4. 사용하는 ISP Port를 선택해준다.

( Tools -> Port-> /dev/ttyUSBn \*n은 0,1,2... 로 변경가능 )

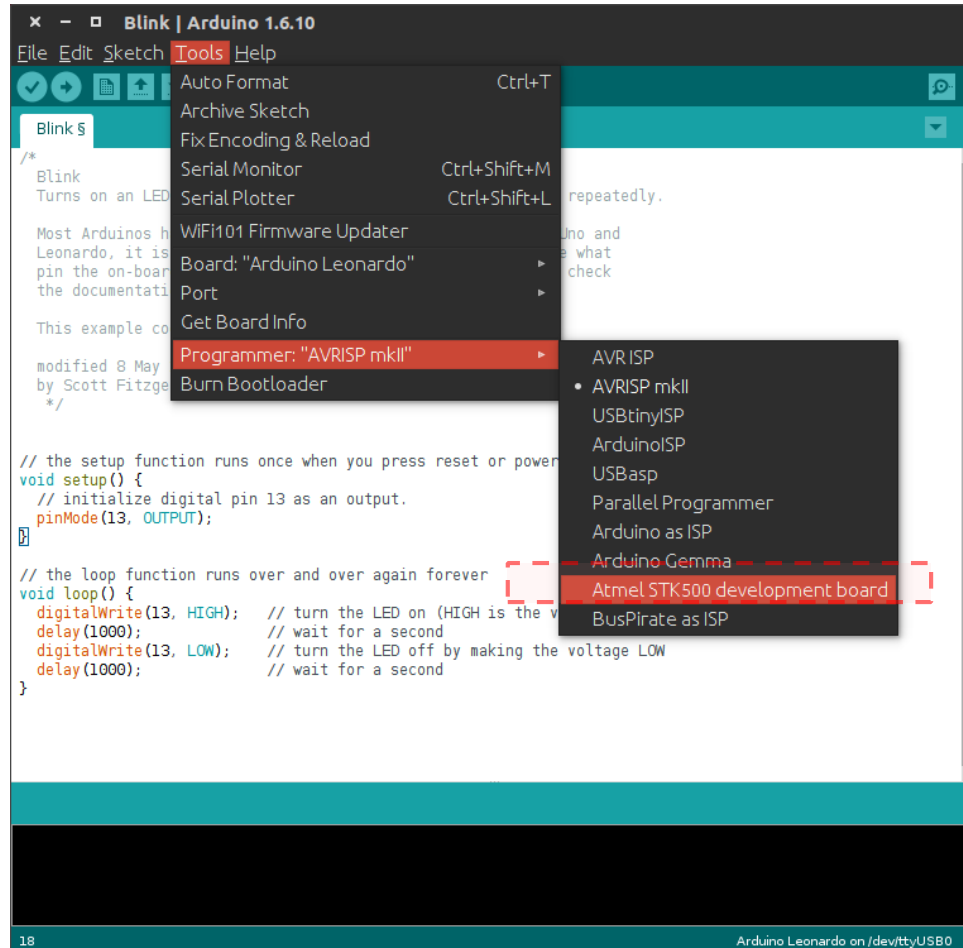


※ 연결된 ISP 포트값은 [dmesg]명령어를 사용하여 확인 가능하다.  
아래 캡처화면은 [dmesg] 명령어를 실행시킨 결과 화면이다.  
로그메시지들 중 [..... attached to **ttyUSB0**]를 통해 현재  
/dev/ttyUSB0로 디바이스가 연결되어 있다는 것을 확인할 수 있다.

```
9200.763600] usb 1-3.1: new full-speed USB device number 18 using xhci_hcd
9200.853512] usb 1-3.1: New USB device found, idVendor=10c4, idProduct=ea60
9200.853520] usb 1-3.1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
9200.853525] usb 1-3.1: Product: CP2103 USB to UART Bridge Controller
9200.853529] usb 1-3.1: Manufacturer: Silicon Labs
9200.853532] usb 1-3.1: SerialNumber: 0001
9200.854976] cp210x 1-3.1:1.0: cp210x converter detected
9200.855215] usb 1-3.1: cp210x converter now attached to ttyUSB0
sykang@sykang-DrinAES:~/project/dk-avr/upload_tests$
```

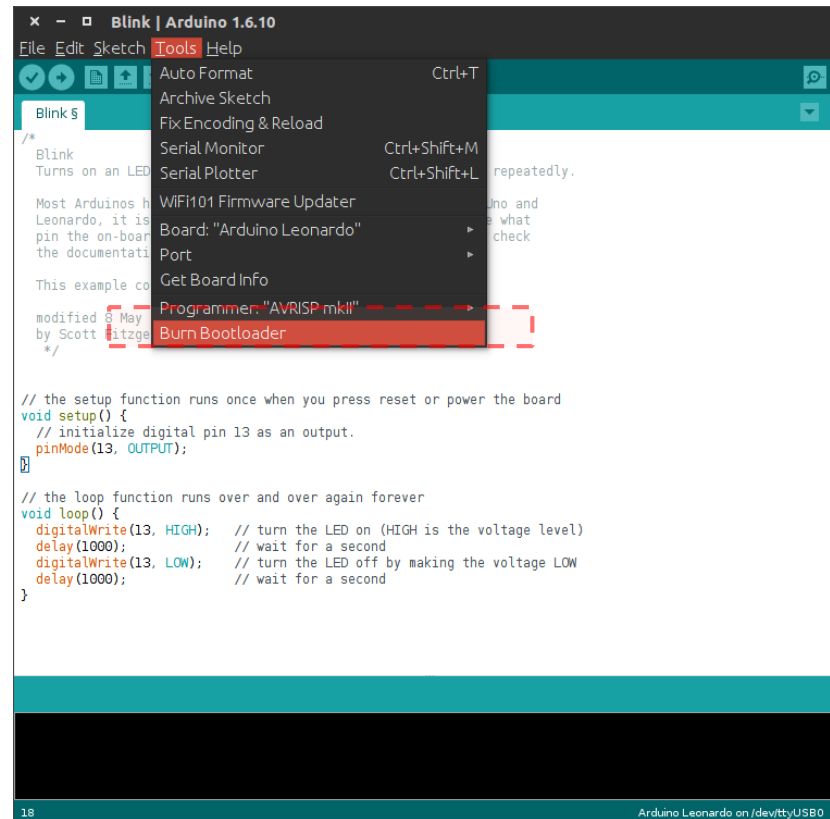
Step5. 사용하는 ISP 종류를 선택한다,

( **Tools -> Programmer -> Atmel STK500 development board** )



Step6. 모든 설정이 끝나고 Burn Bootloader를 클릭해서 부트로더를 올린다.

( **Tools -> Burn Bootloader** )



부트로더를 올리는데 꽤 많은 시간이 소요된다.



# 컴파일 & 퓨징 환경 구축

> 이클립스 및 AVR plugin  
설치

Step1. 이클립스 설치 및 실행에 필요한 Java JDK를 설치한다. 설치하는 아래 명령어를 순서대로 입력한다.

1

```
sykang@sykang-DrimAES:~$ sudo apt-add-repository ppa:webupd8team/java
```

2

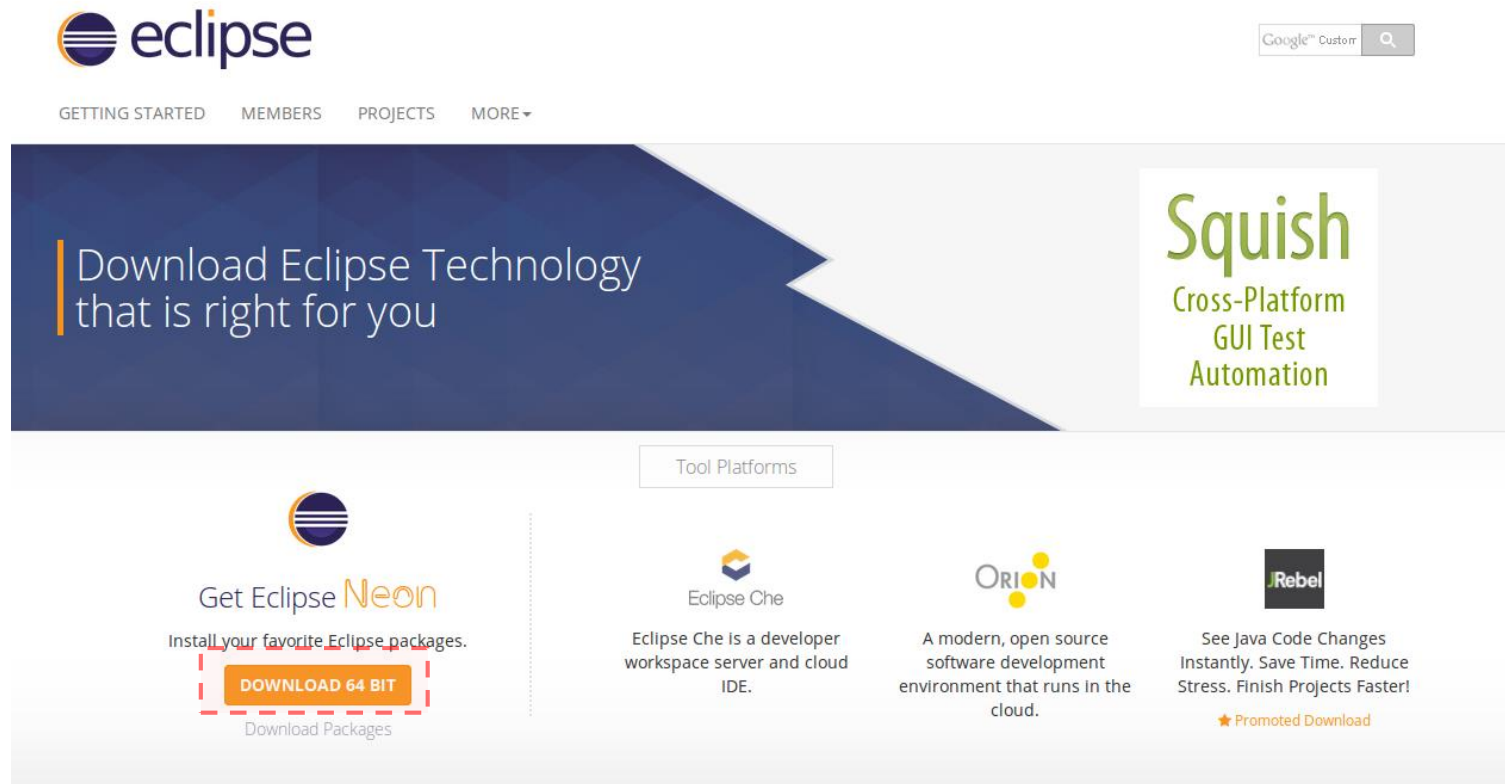
```
sykang@sykang-DrimAES:~$ sudo apt-get update
```

3

```
sykang@sykang-DrimAES:~$ sudo apt-get install oracle-java8-installer
```



Step2. Eclipse 다운로드 사이트(<https://eclipse.org/downloads/>)에서 인스톨 파일을 다운 받은 후 압축을 해제한다.



# 이클립스 및 AVR plugin 설치

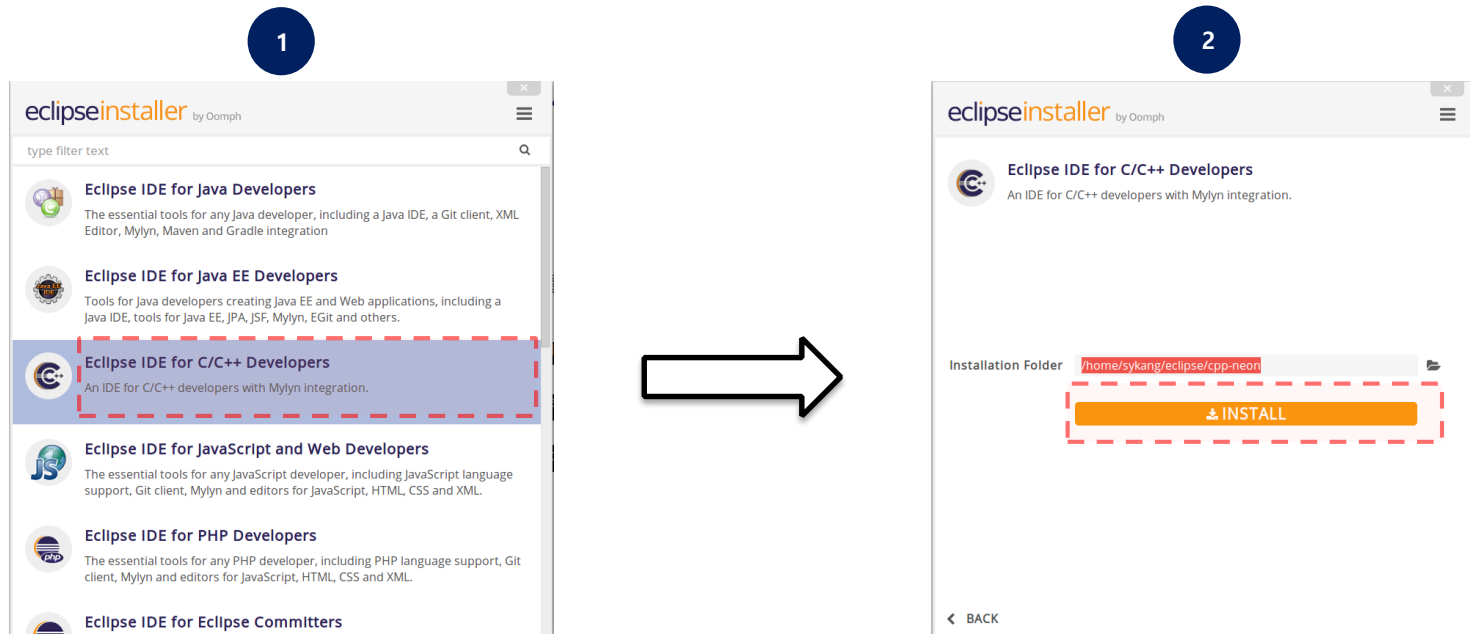
컴파일 & 튜닝 환경 구축

Step3-1. 터미널을 실행하여 위에서 압축을 해지한 폴더로 이동한다.

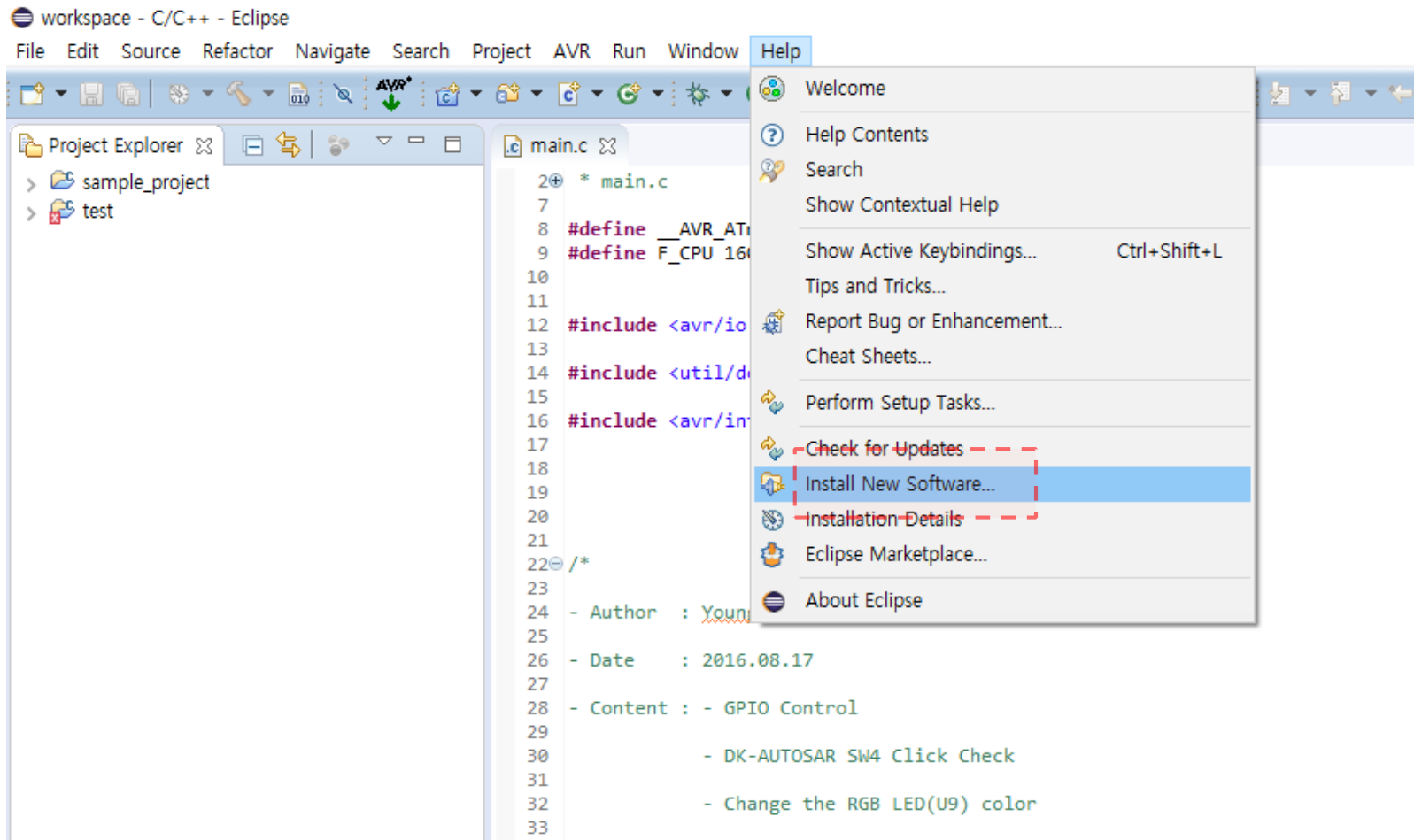
Step3-2. 터미널 창에 **`$.eclipse-inst`**를 입력 후 실행한다.

Step3-3. 이클립스 설치프로그램 창<sup>1</sup>에서 **`[Eclipse IDE for C/C++ Developers]`**를 선택한다.

Step3-4. 설치할 경로 선택 후 **`[Install]`**을 클릭한다<sup>2</sup> [ ]



## Step1. plugin install을 실행한다.[Help -> Install New Software]

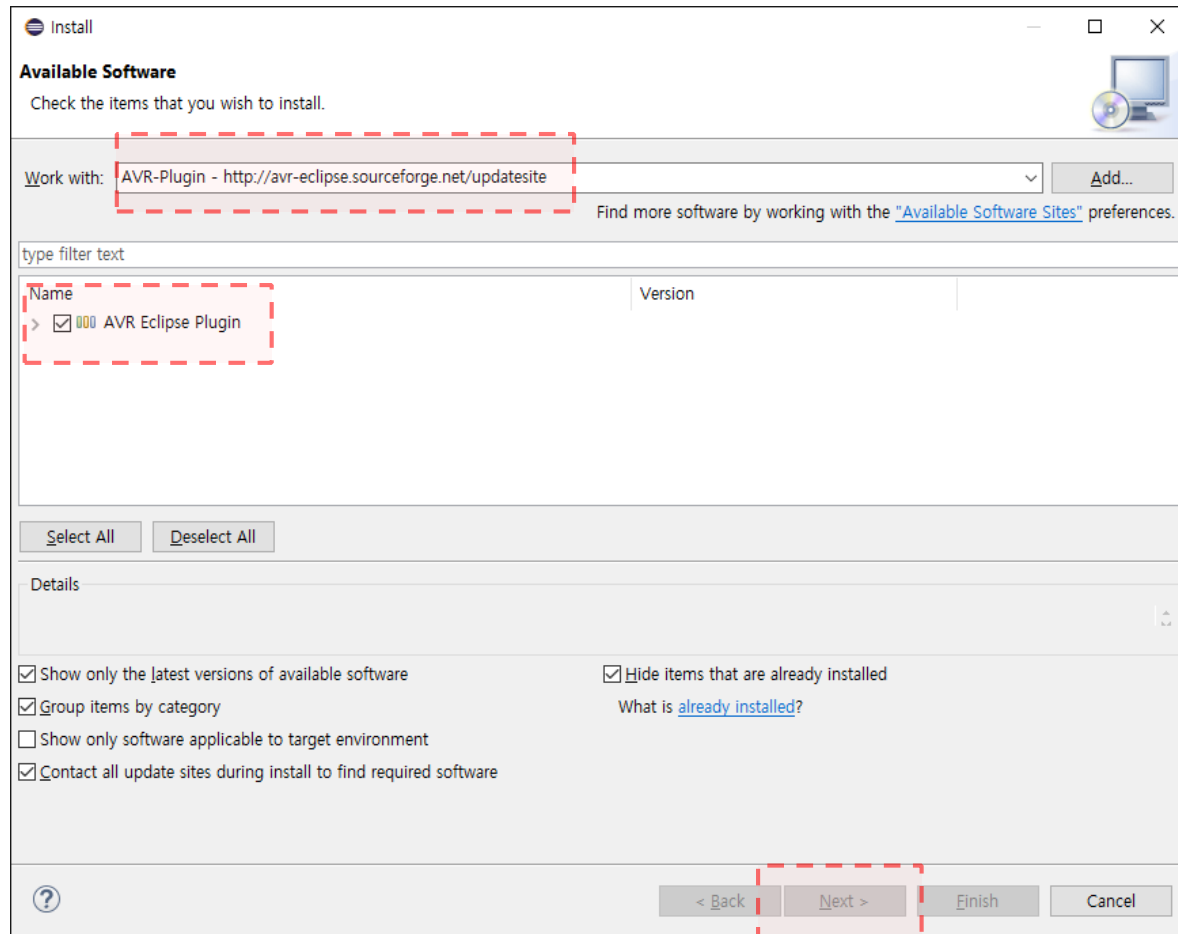


# 이클립스 및 AVR plugin 설치

컴파일 & 퓨징 환경 구축

**Step2. install URL을 입력 후 설치 가능한 plugin 리스트에서 [AVR Eclipse Plugin] 체크한다. 그리고 install 을 진행한다.**

**URL - <http://avr-eclipse.sourceforge.net/updatesite>**





GRIMNES

# RGB LED 점등 프로젝트 실습

> 터미널 환경

Step1. 원하는 경로에 GPIO\_LED라는 디렉토리를 생성한다.

`$ mkdir GPIO_LED`

Step2. 생성한 디렉토리에 gpio.c라는 파일을 생성하고 아래의 그림처럼 소스를 작성한다.

`$ cd GPIO_LED`

`$ vi gpio.c`

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

/*
- Author : Youngmoon Jung in DrimAES
- Date : 2016.08.17
- Content : - GPIO Control
            - DK-AUTOSAR SW4 Click Check
            - Change the RGB LED(U9) color
            * LED is Low Active
*/

/*
- SW4          -> PB4( Set INPUT Mode )
- RGB LED(Red) -> PD4( Set OUTPUT Mode )
- RGB LED(Green) -> PD6( Set OUTPUT Mode )
- RGB LED(Blue) -> PD7( Set OUTPUT Mode )
*/

void init_GPIO()
{
    /* Set PortB4 to Input mode */
    //DDRB &= ~(1<<PB4);
    DDRB = 0x00;
    /* Set PortD4,6,7 to Outputmode */
    DDRD |= (1<<PD4)|(1<<PD6)|(1<<PD7);

    /* Turn off All LED */
    PORTD = 0xff;
}

void main(void)
{
    init_GPIO();

    while(1)
    {
        /* Turn on Red LED */
        PORTD = ~(1<<PD4);
        _delay_ms(1000);

        /* Turn on Green LED */
        PORTD = ~(1<<PD6);
        _delay_ms(1000);

        /* Turn on Blue LED */
        PORTD = ~(1<<PD7);
        _delay_ms(1000);
    }
}
```

※ 소스는 샘플 폴더를 참고.

## Step3. 빌드 & 퓨징 환경을 만들기위해서 첨부파일에있는 Makefile 을 현재 경로에 복사 후 수정한다.

※ 기존 Makefile에서 사용하는 Programmer, 프로젝트명 등에 맞게 수정 작업을 해줘야한다. ( 5개의 항목의 수정이 필요 )

```
# Makefile for DK-AVR @DrimAES Inc.
#
# Usage :
#   $ make
#   $ sudo make upload
#   $ make clean
#
CC = avr-gcc
AR = avr-ar

MCU:=atmega32u4
F_CPU:=16000000

AVRDUDE_PROGRAMMER := avrisp2
AVRDUDE_PORT := /dev/ttyUSB0

TARGETNAME = gpio
SRC = $(TARGETNAME).c

# Flags for the linker and the compiler
COMMON_FLAGS = -DF_CPU=$(F_CPU) -mmcu=$(MCU)
COMMON_FLAGS += -g -Os -Wall -funsigned-char -fpack-struct -fshort-enums
COMMON_FLAGS += -ffunction-sections -fdata-sections -WL,--gc-sections
COMMON_FLAGS += -WL,--relax -mcalls-prologues
CFLAGS = $(COMMON_FLAGS) -std-gnu99 -Wstrict-prototypes

OBJ = $(SRC:.c=.o)

.SUFFIXES: .elf .hex .dis

.PHONY: all
all: $(TARGETNAME).dis $(TARGETNAME).hex
    avr-size $(TARGETNAME).elf

.PHONY: upload
upload: $(TARGETNAME).dis $(TARGETNAME).hex
    avrdude -F -p $(MCU) -P $(AVRDUDE_PORT) -c $(AVRDUDE_PROGRAMMER) -v -v -U flash:w:$(TARGETNAME).hex
    avr-size $(TARGETNAME).elf

.PHONY: clean
clean:
    $(RM) $(TARGETNAME).hex $(TARGETNAME).elf $(TARGETNAME).a $(TARGETNAME).dis $(OBJ)

# implicit rules
.elf.hex:
    avr-objcopy -O ihex -R .eeprom $< $@

# explicit rules
$(TARGETNAME).elf: $(TARGETNAME).a $(OBJ)
    $(LINK.o) $(COMMON_FLAGS) $(TARGETNAME).a $(LOADLIBES) $(LDLIBS) -o $@

$(TARGETNAME).dis: $(TARGETNAME).elf
    avr-objdump -S $< > $@
```

MCU : 사용하는 타겟칩 이름을 입력  
( Ex. atmega32u4 )

F\_CPU : 사용되는 클럭을 입력  
( Ex. 16Mhz 일경우 16000000 )

AVRDUDE\_PROGRAMMER  
: 사용 할 programmer를 입력

AVRDUDE\_PORT  
: 인식 된 ISP의 포트를 입력한다  
( 만약, ttyUSB0으로 잡혔다면, /dev/ttyUSB0으로 입력 )

TARGETNAME : main 소스의 파일명을 입력  
( Ex. main.c 일경우에는 TARGETNAME = main )

**Step4. 아래의 명령어를 입력하면 빌드 부터 푸징까지의 작업이 한번에 이뤄진다.**

( 단, Makefile과 소스파일이 모두있는 경로에서 명령어를 입력해야한다.)

## \$ sudo make upload

[illegible]



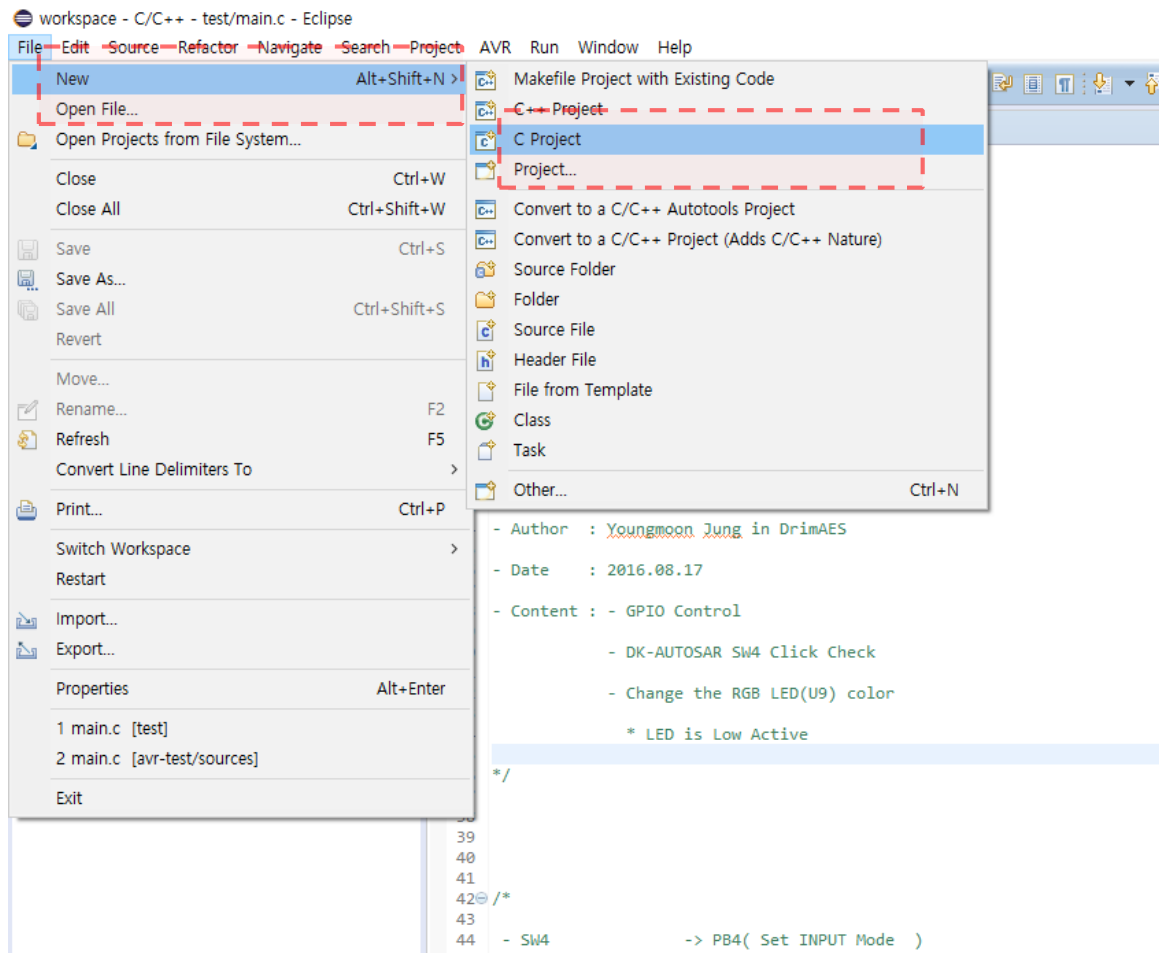


GRIMNES

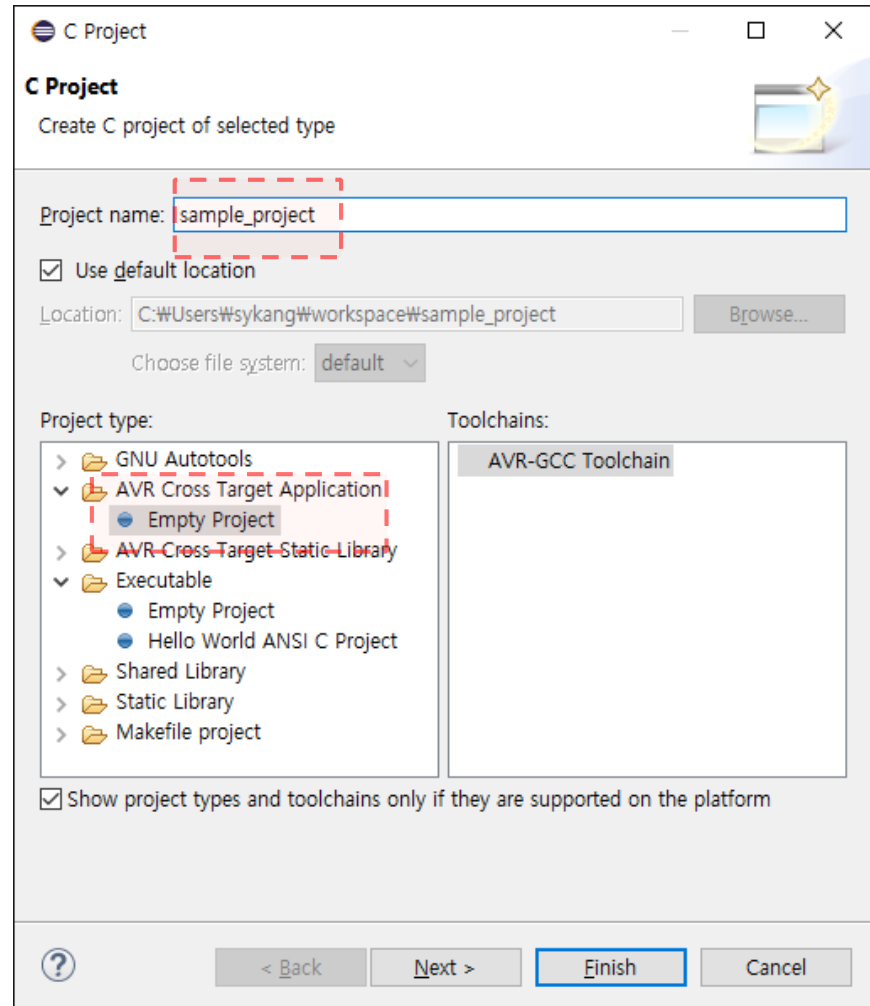
# RGB LED 점등 프로젝트 실습

> 이클립스 환경

### Step1. [File -> New -> C Project]를 실행한다.

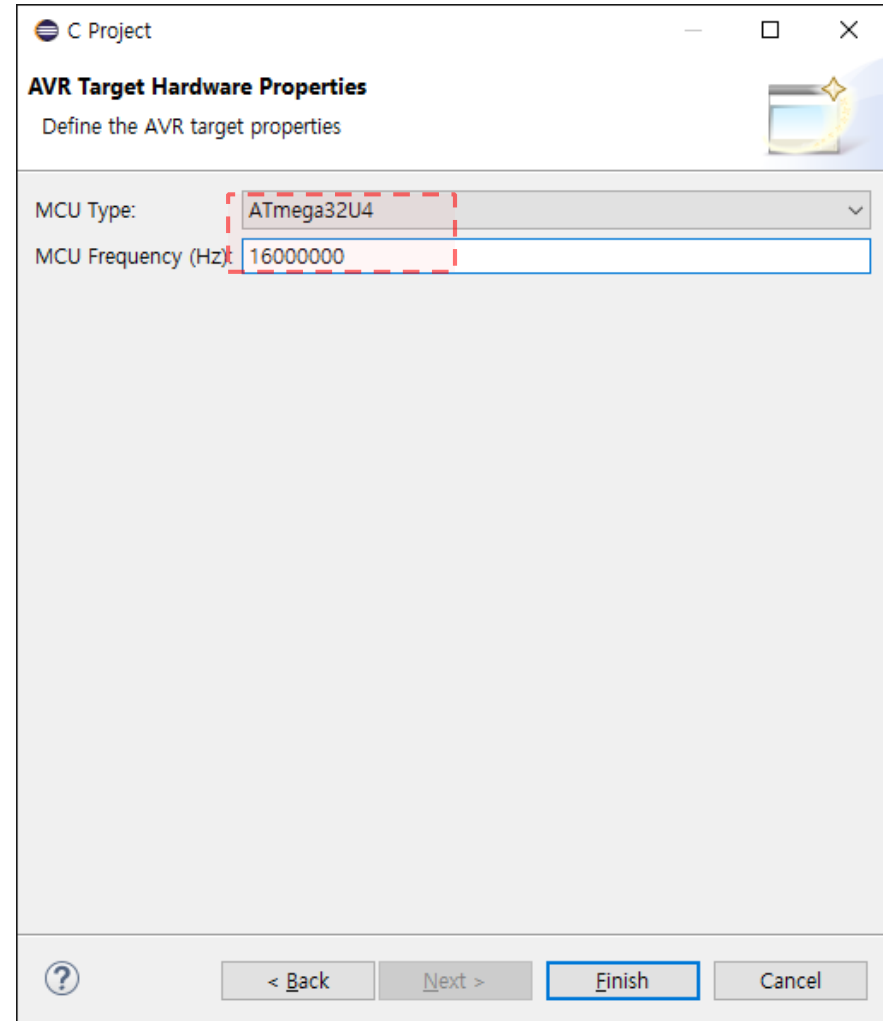


Step2. Project type은 **AVR Cross Target Application**을 선택 후 Project name 작성 한다. 그리고 [Next]를 클릭한다.



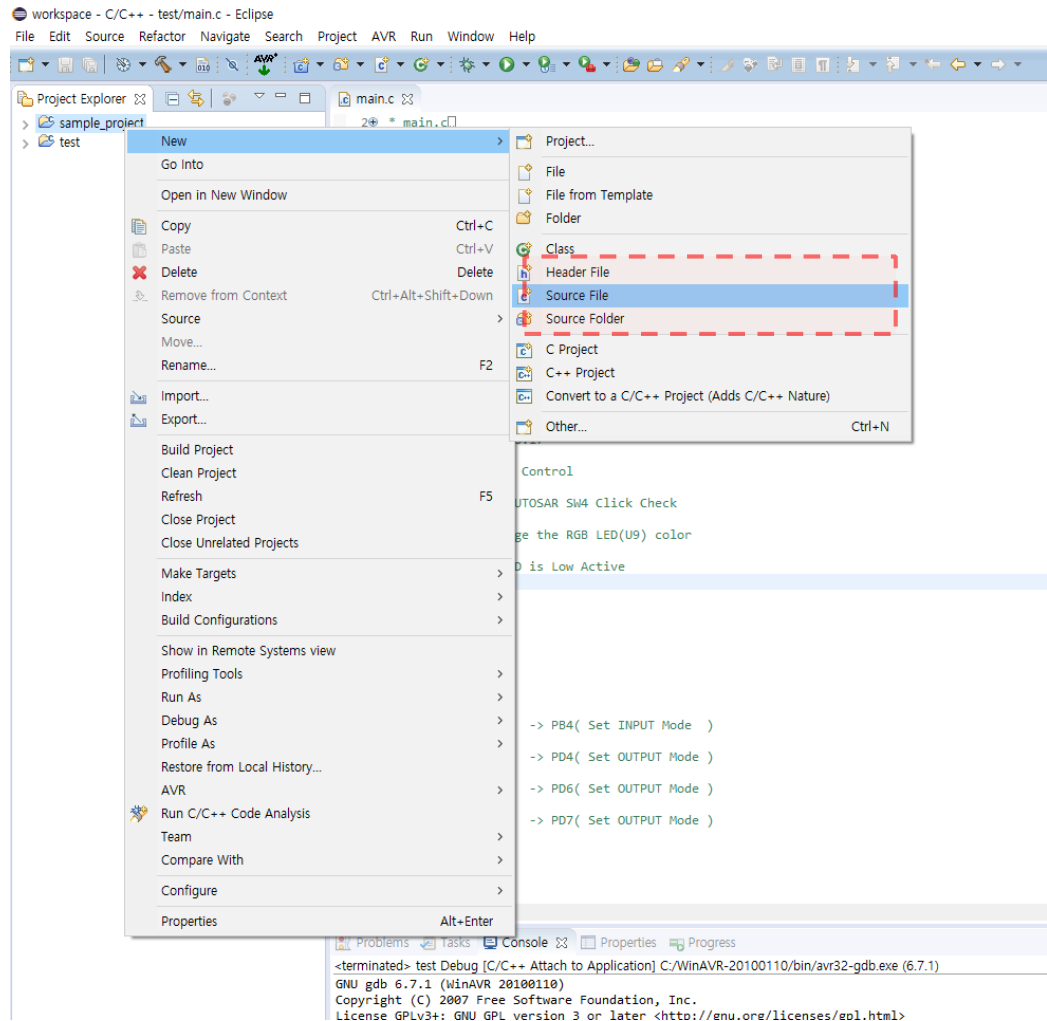
Step3. MCU Type과 Frequency를 설정한다.

- Type : **ATmega32U4**
  - Frequency : **16000000**
- 그리고 [Finish]를 클릭한다.

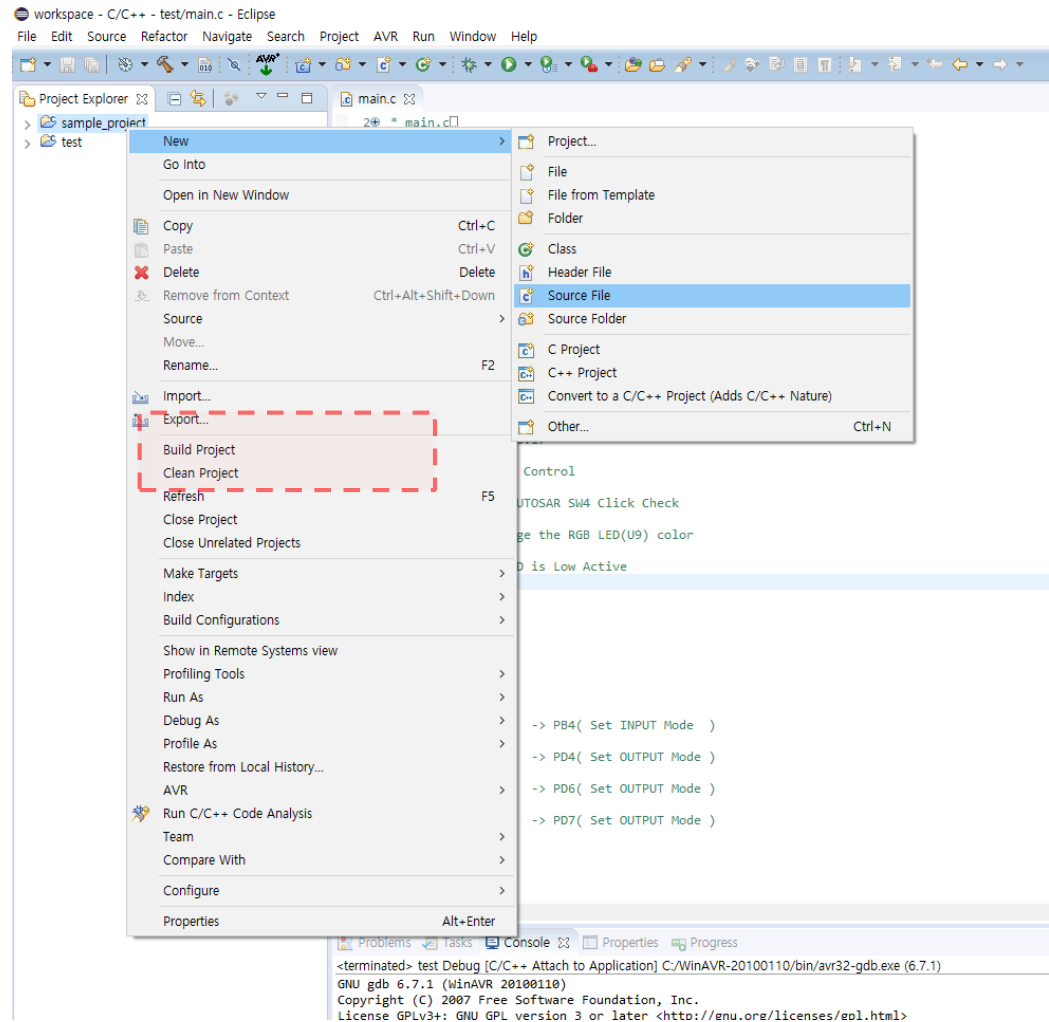


**Step4. 생성된 프로젝트에  
소스파일을 추가하여 코드를  
작성한다.**

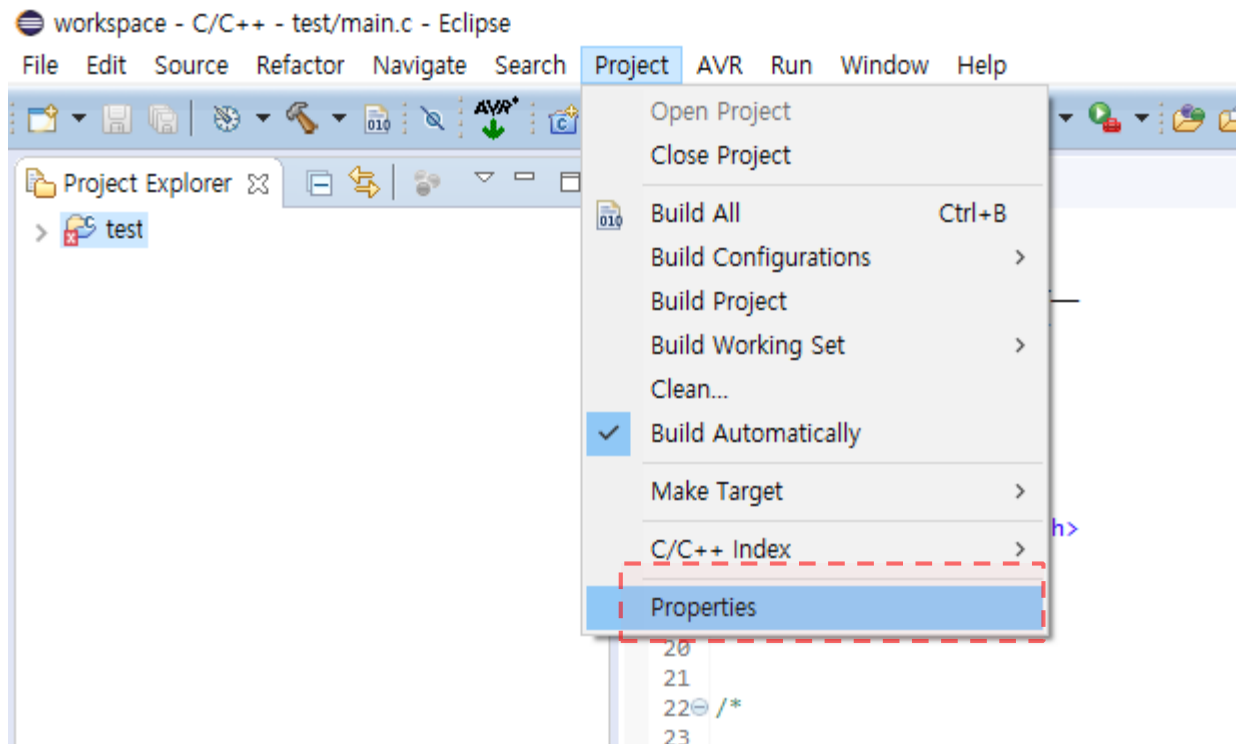
**※ 소스코드는 첨부파일 참조.**



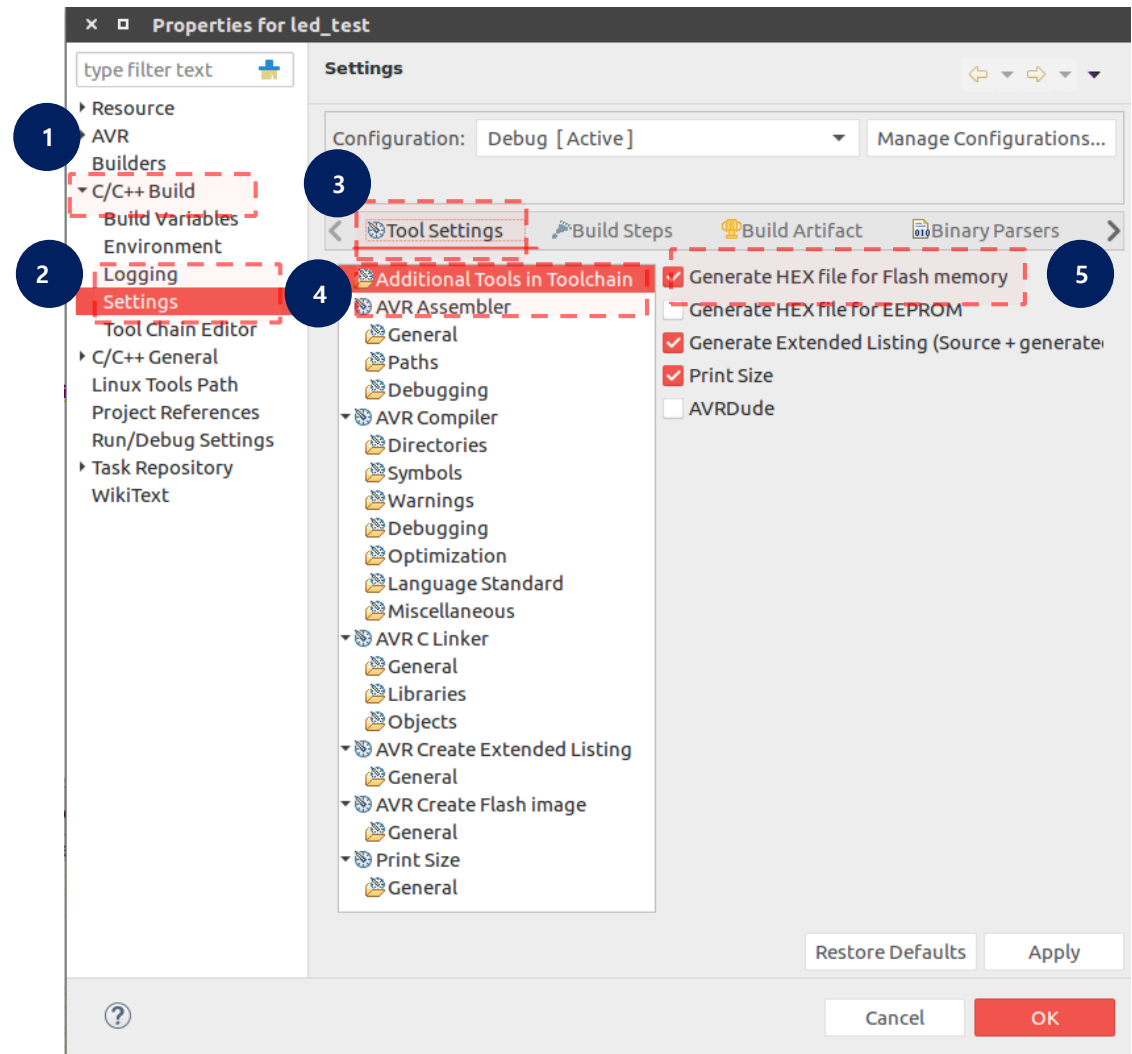
**Step5. 코드작성이 마무리 되면  
[Build Project]를 실행하여  
빌드한다.**



## Step6. [Project -> Properties]를 실행한다.

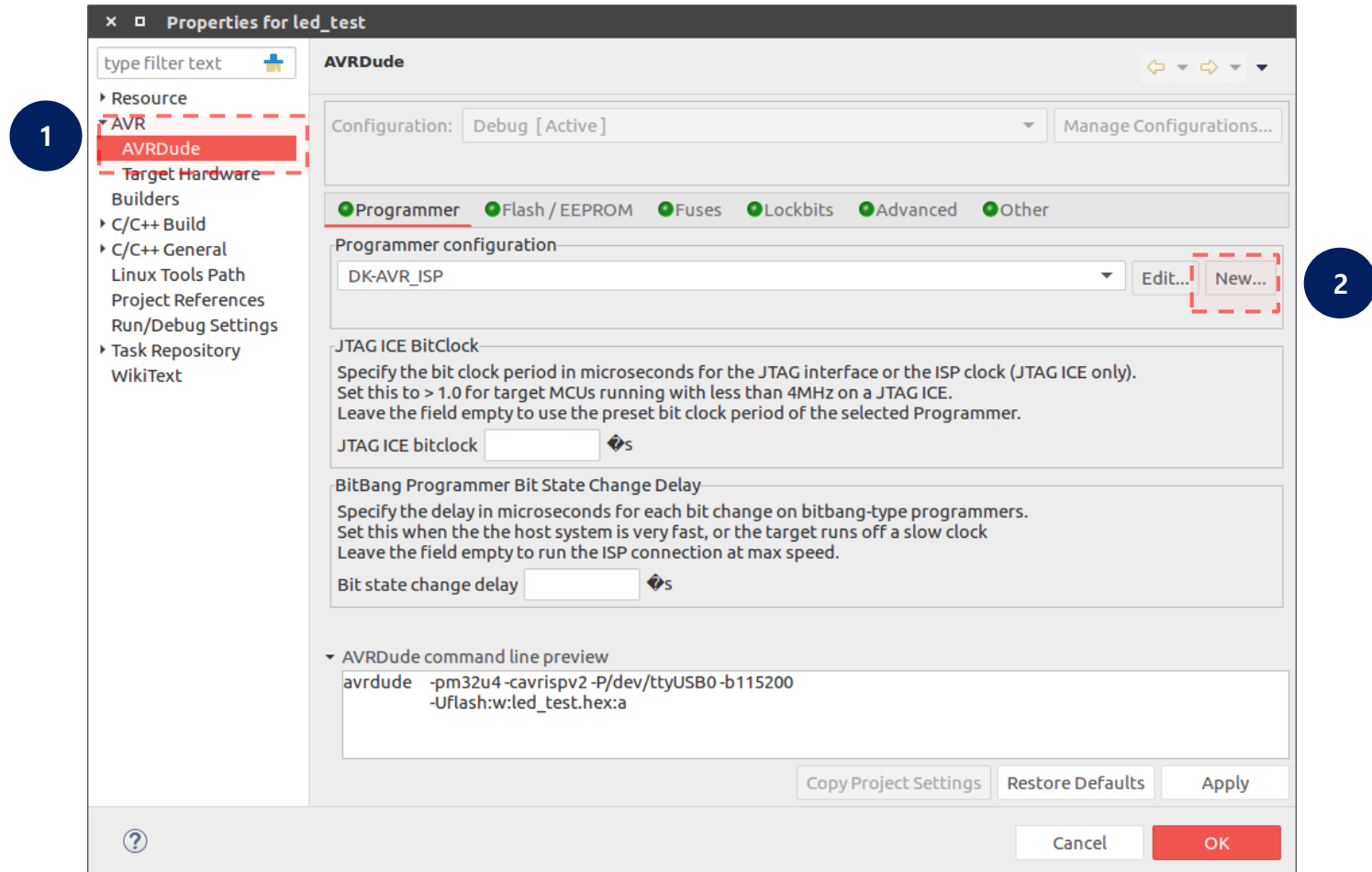


Step7. Hex파일도 생성되는  
옵션 추가한다.  
[C/C++ Build -> Settings ->  
Tool Settings -> Additional  
Tools in Toolchain ->  
Generate HEX file for Flash  
memory]





Step8. [AVRDude]를 선택 후, Programmer configuration 설정을 위해 [New...]를 클릭한다.



## Step9. 설정값을 입력한다.

- Programmer Hardware = **Atmel AVR ISP V2**
- Port : **/dev/ttyUSB0**
- ※ port값 확인 방법은 다음 페이지 참조.
- Baudrate : **115200**

입력이 끝난 후 [OK]를 클릭한다.

Configuration name: DK-AVR\_ISP

Description: Default AVRDUDE Programmer Configuration. Modify as required for your setup.

Programmer Hardware (-c):

- ABCMini Board, aka Dick Smith HOTCHIP
- alias for arduino-ft232r (decimila)
- Altera ByteBlaster
- Amontec JTAGKey, JTAGKey-Tiny and JTAGKey2
- Any usbasp clone with correct VID/PID
- Arduino
- Arduino: FT232R connected to ISP
- AT-ISP V1.1 programming cable for AVR-SDK1 from <http://micro-research.co.th/> micro-research.co.th
- Atmel AppNote AVR109 Boot Loader
- Atmel AppNote AVR911 AVROSP
- Atmel at89isp cable
- Atmel AVR Dragon in debugWire mode
- Atmel AVR Dragon in HVSP mode
- Atmel AVR Dragon in ISP mode
- Atmel AVR Dragon in JTAG mode
- Atmel AVR Dragon in PDI mode
- Atmel AVR Dragon in PP mode
- Atmel AVR ISP
- Atmel AVR ISP mkII
- Atmel AVR ISP mkII (avrisp2)
- Atmel AVR ISP V2**
- Atmel AVR JTAGICE3 in debugWire mode
- Atmel AVR JTAGICE3 in ISP mode
- Atmel AVR JTAGICE3 in JTAG mode
- Atmel AVR JTAGICE3 in PDI mode
- Atmel AVR XplainedPro in JTAG mode
- Atmel Butterfly Development Board
- Atmel JTAG ICE (mkI)
- Atmel JTAG ICE (mkI) (itag1)
- Atmel JTAG ICE (mkI) (itag1slow)
- Atmel JTAG ICE mkII
- Atmel JTAG ICE mkII (itag2)
- Atmel JTAG ICE mkII (itag2fast)
- Atmel JTAG ICE mkII (itag2slow)
- Atmel JTAG ICE mkII in AVR32 mode
- Atmel JTAG ICE mkII in debugWire mode
- Atmel JTAG ICE mkII in ISP mode
- Atmel JTAG ICE mkII PDI mode
- Atmel Low Cost Serial Programmer
- Atmel STK500
- Atmel STK500 V2 in high-voltage serial programming mode

Override default port (-P): /dev/ttyUSB0

Override default baudrate (-b): 115200

Other options: Use this field to add any avrdude option not covered by the plugin.

State of Parallel Port lines after AVRDUDE exit:

/Reset Line

- ☒ restore to previous state
- ☐ activated (-E reset)
- ☐ deactivated (-E noreset)

Vcc Lines

- ☒ restore to previous state
- ☐ activated (-E vcc)
- ☐ deactivated (-E novcc)

Delay between avrdude invocations: milliseconds

Command line preview: avrdude -cavrispv2 -P/dev/ttyUSB0 -b115200 [...part specific options...]

Buttons: Cancel, OK

※ 연결된 ISP 포트값은 [dmesg]명령어를 사용하여 확인 가능하다.

아래 캡처화면은 [dmesg] 명령어를 실행시킨 결과 화면이다.

로그메시지들 중 [..... attached to **ttyUSB0**]를 통해 현재 /dev/ttyUSB0로 디바이스가 연결되어 있다는 것을 확인할 수 있다.

```
[ 9200.763669] usb 1-3.1: new full-speed USB device number 18 using xhci_hcd
[ 9200.853512] usb 1-3.1: New USB device found, idVendor=10c4, idProduct=ea60
[ 9200.853520] usb 1-3.1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 9200.853525] usb 1-3.1: Product: CP2103 USB to UART Bridge Controller
[ 9200.853529] usb 1-3.1: Manufacturer: Silicon Labs
[ 9200.853532] usb 1-3.1: SerialNumber: 0001
[ 9200.854976] cp210x 1-3.1:1.0: cp210x converter detected
[ 9200.855215] usb 1-3.1: cp210x converter now attached to ttyUSB0
sykang@sykang-Dr1mAES:~/project/dk-avr/upload_test$
```

**Step10. [AVR -> Uploader Project to Target Device]를 실행한다.**

