

“본 강의 동영상 및 자료는 대한민국 저작권법을 준수합니다. 본 강의 동영상 및 자료는 상명대학교 재학생들의 수업목적으로 제작·배포되는 것이므로, 수업목적으로 내려받은 강의 동영상 및 자료는 수업목적 이외에 다른 용도로 사용할 수 없으며, 다른 장소 및 타인에게 복제, 전송하여 공유할 수 없습니다. 이를 위반해서 발생하는 모든 법적 책임은 행위 주체인 본인에게 있습니다.”



피지컬 컴퓨팅

Lec. 3. 컴퓨터 구조 (IO, GPIO, LED 실습)

Heenam Yoon

Department of
Human-Centered Artificial Intelligence

E-mail) h-yoon@smu.ac.kr
Room) 0112



| 프로그램 설치

- 설치 파일
- ST-Link만 남겨 놓고 체크박스 모두 해제 & 설치

STM32의 구조

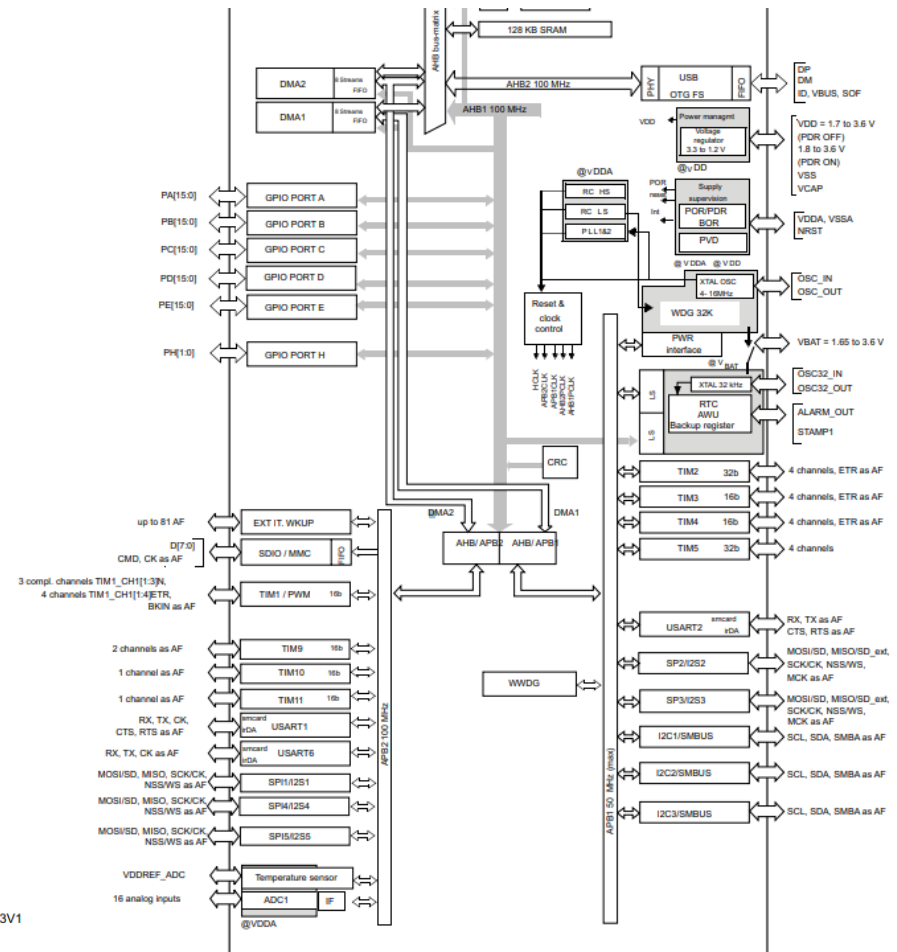
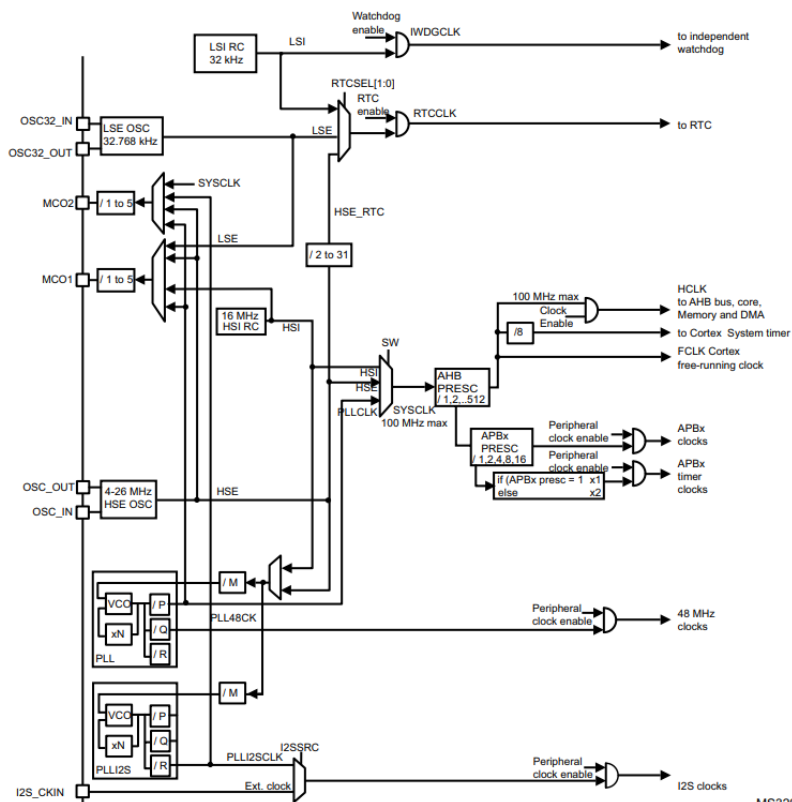
STM32의 클럭

- 메인 시스템 클럭인 SYSCLK는 선택하여 사용할 수 있다.
 - HSE(High speed external clock signal) : 외부로부터 입력되는 클럭으로, 크리스탈이나 오실레이터를 통해 입력된다.
 - HSI(High speed internal clock signal) : 내부의 16MHz RC 오실레이터에서 제공하는 클럭으로 이를 직접 시스템 클럭으로 사용하거나 PLL 블록을 통해 원하는 클럭으로 증감시켜 사용할 수 있다.
- 메인 시스템 클럭과 별도로 저속의 클럭 소스를 제공한다.
 - LSE(Low speed external clock signal) : 32.768kHz의 Low speed의 클럭을 외부 크리스탈이나 오실레이터를 통해 입력 받으며 이는 저전력 또는 높은 정밀도를 요하는 RTC 또는 타이머의 클럭으로 사용된다.
 - LSI(Low speed internal clock signal) : 저전력 설계를 위해 사용할 수 있도록 대략 32kHz의 내부 클럭을 제공한다.

RCC (Reset Clock Controller)

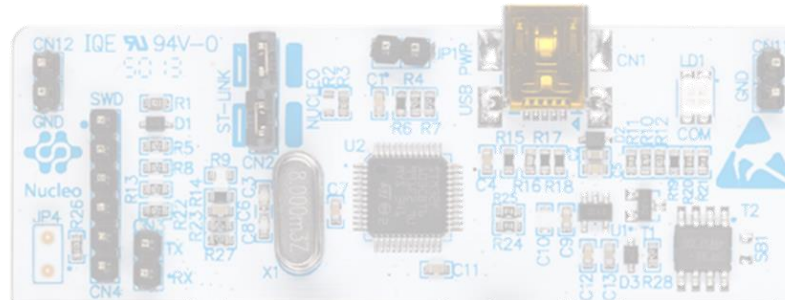
- Reset과 내부 Clock을 관리하는 Controller
- MCU 내의 주변장치의 사용을 위해 Clock 공급
- Clock소스
 - HIS (High Speed Internal): MPU 내부의 발진기에 의해서 생성된 16MHz클럭(8MHz)
 - HSE (High Speed External): 외부 크리스탈 혹은 크리스탈 발진기에서 생성되는 클럭
 - PLL (Phase Locked Loop): HSI 혹은 HSE 클럭이 PLL로 인가되어 생성되는 클럭
- Clock이 공급되는 버스
 - AHB (Advanced High-performance Bus)
 - GPIO로 클럭 공급
 - APB (Advanced Peripheral Bus): AHB로 전달된 클럭을 주변장치로 공급
 - APB1: High Speed
 - APB2: Low Speed

RCC(Reset Clock Controller)

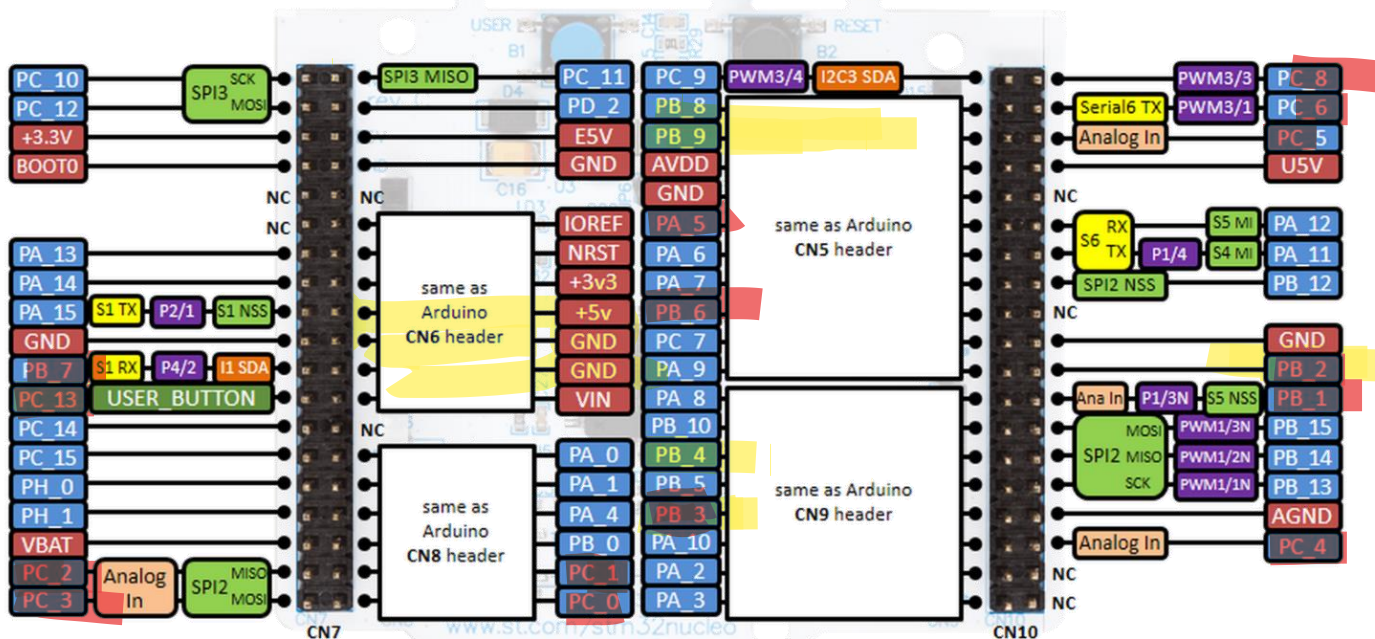


NUCLEO-F411RE board


life.augmented
Nucleo F411RE
Morpho Headers



mbed
Enabled



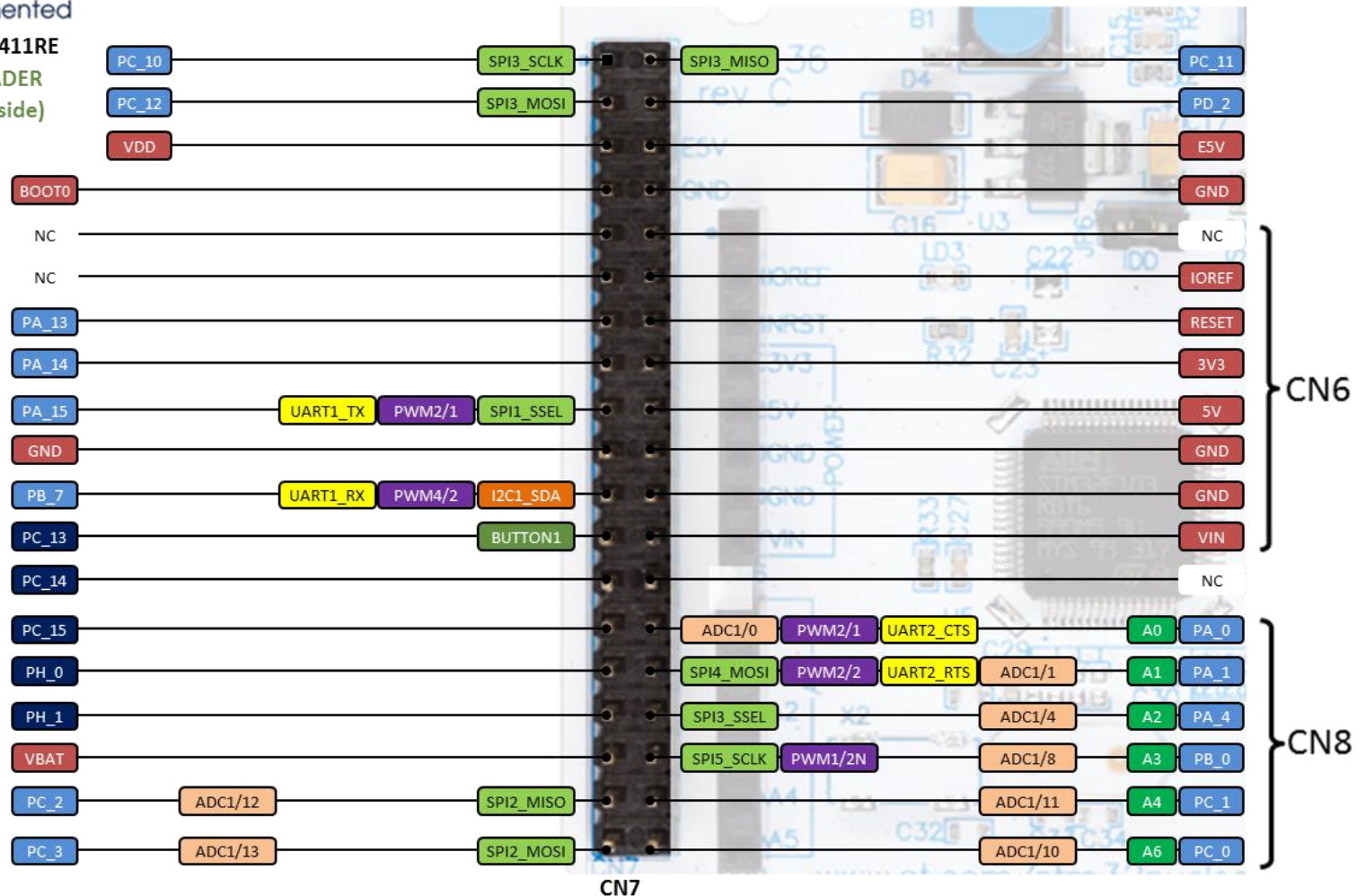
NUCLEO-F411RE board

NUCLEO-F411RE board



NUCLEO-F411RE

CN7 HEADER
(top left side)

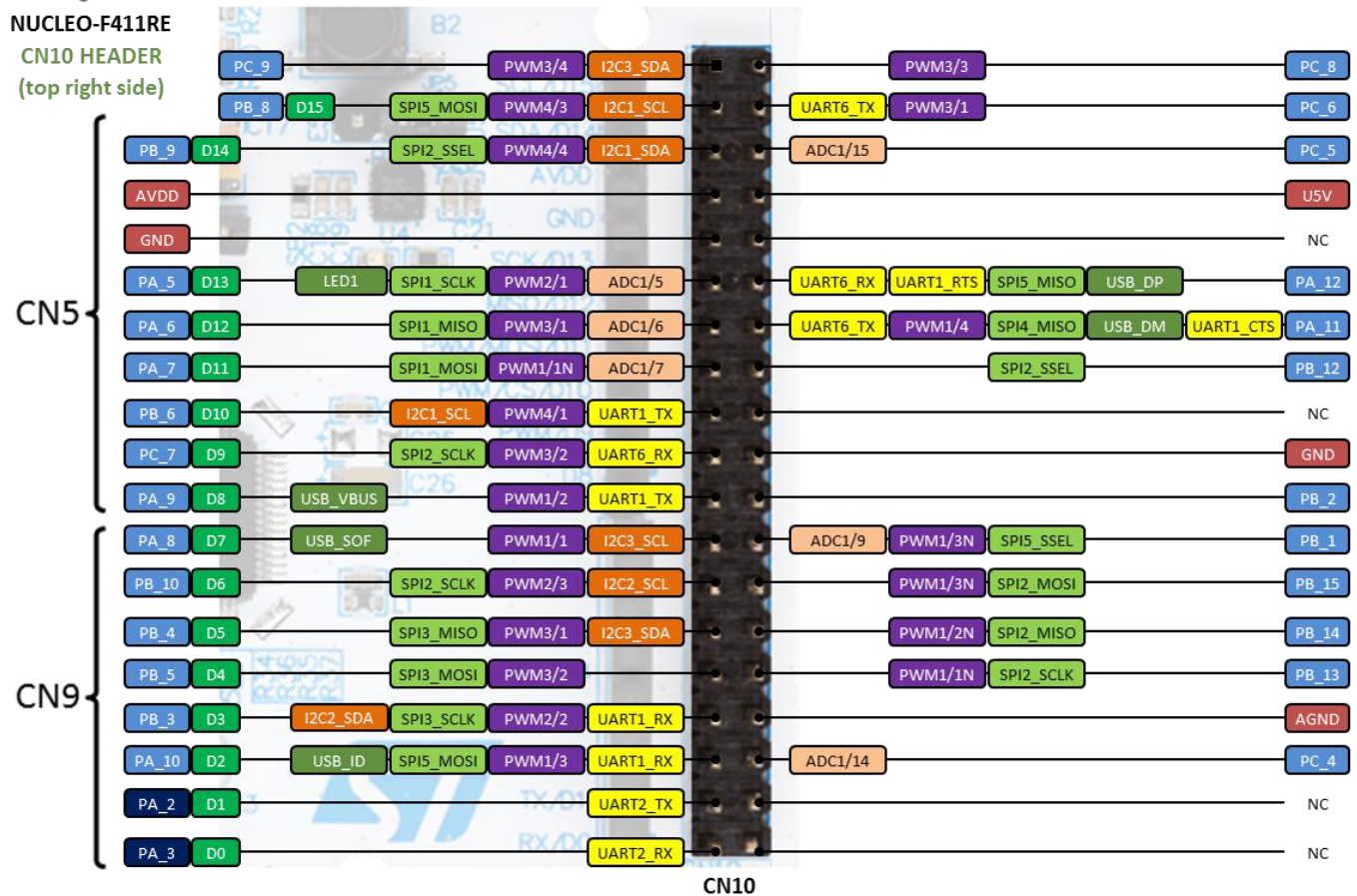


NUCLEO-F411RE board



NUCLEO-F411RE

CN10 HEADER
(top right side)



GPIO

GPIO(General Purpose Input/Output)

- MCU의 주변기기 중에서 가장 기본적인 I/O 장치
- 디지털 신호를 읽거나 출력시킬 수 있는 주변기기
 - 입력 기능: GPIO 핀을 통해 High/Low (3.3V/0V) 신호 입력
 - 출력 기능: GPIO 핀을 통해 High/Low (3.3V/0V) 신호 출력
- AFIO (Alternate Function Input/Output)
 - 범용 입출력이 아닌 특수한 목적으로 사용
 - 하드웨어적으로 설계된 기능으로 GPIO핀이 다른 기능으로 동작하도록 사용자가 프로그램으로 설정 가능

Table 9. Alternate function mapping

Port	AF00	AF01	AF02	AF03	AF04	AF05	AF06	AF07	AF08	AF09	AF10	AF11	AF12	AF13	AF14	AF15
	SYS_AF	TIM1/TIM2	TIM3/ TIM4/ TIM5	TIM9/ TIM10/ TIM11	I2C1/I2C2/ I2C3	SPI1/I2S1S PI2/ I2S2/SPI3/ I2S3	SPI2/I2S2/ SPI3/ I2S3/SPI4/ I2S4/SPI5/ I2S5	SPI3/I2S3/ USART1/ USART2	USART6	I2C2/ I2C3	OTG1_FS		SDIO			
PA0	-	TIM2_CH1/ TIM2_ETR	TIM5_CH1	-	-	-	-	USART2_ CTS	-	-	-	-	-	-	-	EVENT OUT
PA1	-	TIM2_CH2	TIM5_CH2	-	-	SPI4_MOSI /I2S4_SD	-	USART2_ RTS	-	-	-	-	-	-	-	EVENT OUT
PA2	-	TIM2_CH3	TIM5_CH3	TIM9_CH1	-	I2S2_CKIN	-	USART2_ TX	-	-	-	-	-	-	-	EVENT OUT
PA3	-	TIM2_CH4	TIM5_CH4	TIM9_CH2	-	I2S2_MCK	-	USART2_ RX	-	-	-	-	-	-	-	EVENT OUT

GPIO

- 패키지에 따라 GPIOA, GPIOB, GPIOC, GPIOD, GPIOE, GPIOH의 최대 6개의 포트를 가질 수 있음
- 각 포트는 최대 16개의 핀을 가짐
- 5V-tolerant I/Os 기능이 있어 3.3V외에 5V 인터페이스도 가능

Table 8. STM32F411xC/xE pin definitions (continued)

Pin number					Pin name (function after reset) ⁽¹⁾	Pin type	I/O structure	Notes	Alternate functions	Additional functions
UFQFPN48	LQFP64	WLSP49	LQFP100	UFBGA100						
-	-	-	5	D2	PE6	I/O	FT	-	TRACED3, TIM9_CH2, SPI4_MOSI/I2S4_SD, SPI5_MOSI/I2S5_SD, EVENTOUT	-
-	-	-	-	D3	VSS	S	-	-	-	-
-	-	-	-	C4	VDD	S	-	-	-	-
1	1	B7	6	E2	VBAT	S	-	-	-	-
2	2	D5	7	C1	PC13- ANTI_TAMP	I/O	FT	(2)(3)	-	RTC_AMP1, RTC_OUT, RTC_TS
3	3	C7	8	D1	PC14- OSC32_IN	I/O	FT	(2)(3) (4)	-	OSC32_IN
4	4	C6	9	E1	PC15- OSC32_OUT	I/O	FT	-	-	OSC32_OUT
-	-	-	10	F2	VSS	S	-	-	-	-

Table 7. Legend/abbreviations used in the pinout table

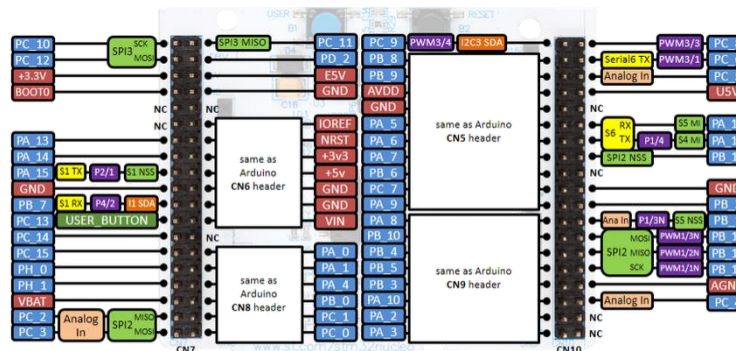
Name	Abbreviation	Definition
Pin name	Unless otherwise specified in brackets below the pin name, the pin function during and after reset is the same as the actual pin name	
Pin type	S	Supply pin
	I	Input only pin
	I/O	Input/ output pin
I/O structure	FT	5 V tolerant I/O
	TC	Standard 3.3 V I/O
	B	Dedicated BOOT0 pin
	NRST	Bidirectional reset pin with embedded weak pull-up resistor
Notes	Unless otherwise specified by a note, all I/Os are set as floating inputs during and after reset	
Alternate functions	Functions selected through GPIOx_AFR registers	
Additional functions	Functions directly selected/enabled through peripheral registers	

GPIO

GPIO main features

- Up to 16 I/Os under control
- Output states: push-pull or open drain + pull-up/down
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Analog function
- Alternate function input/output selection registers (at most 16 AFs per I/O)

PB15	PB14	PB13	PB12	PB11	PB10	PB09	PB08	PB07	PB06	PB05	PB04	PB03	PB02	PB01	PB00
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------



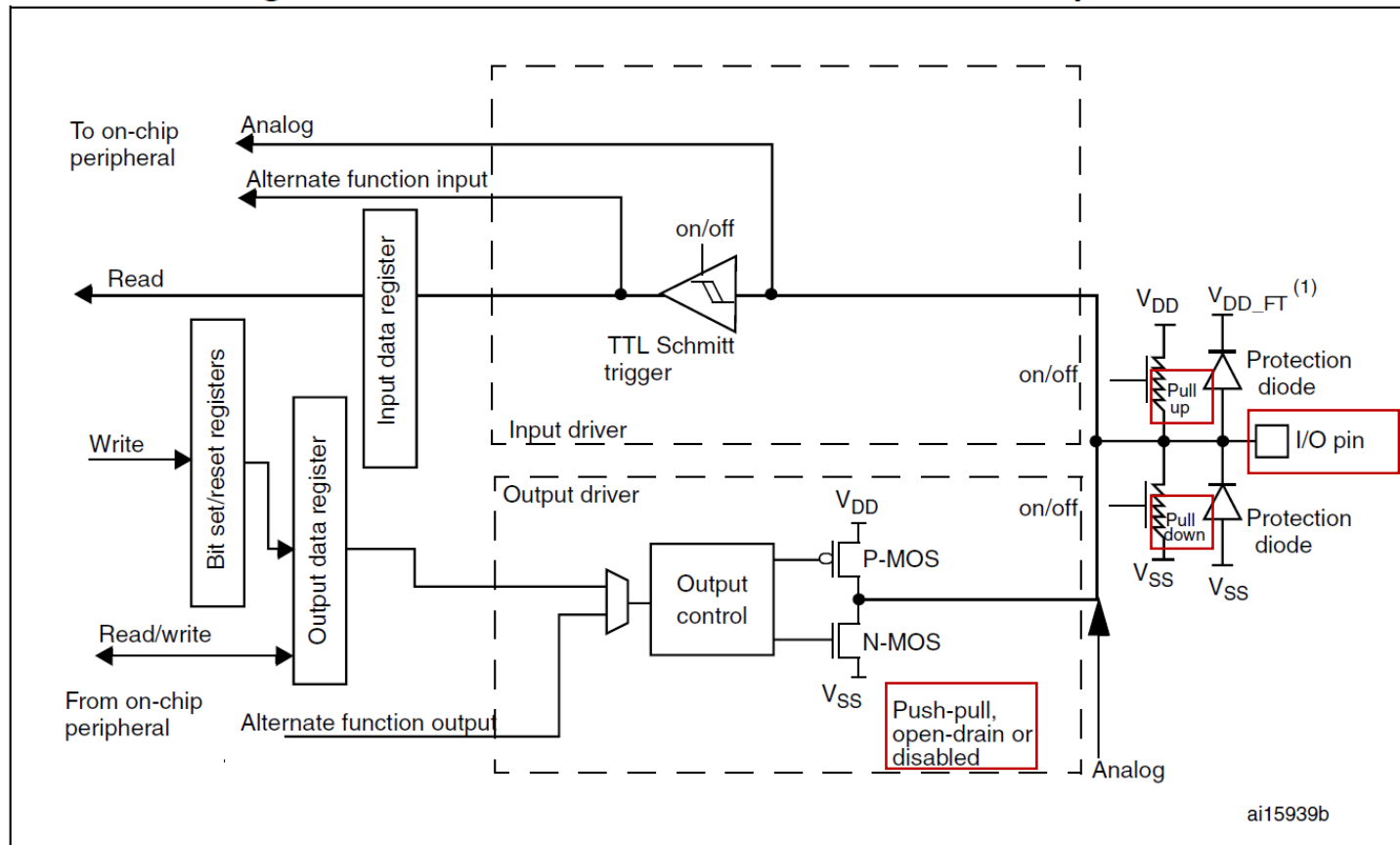
I GPIO

GPIO functional description

- Input floating/pull-up/pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

- General-Purpose I/O Port

Figure 25. Basic structure of a five-volt tolerant I/O port bit

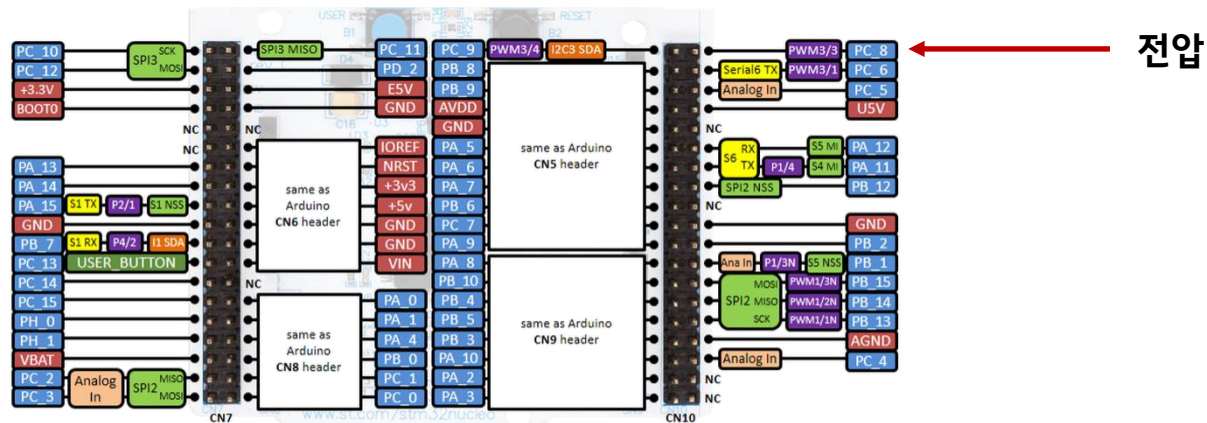


1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD} .

GPIO: 입력

- 포트를 "입력"으로 쓴다면
- 외부의 어떤 값 (전압)이 해당 포트에 입력으로 들어오고, MCU는 이에 반응하여 무언가를 수행하게 됨

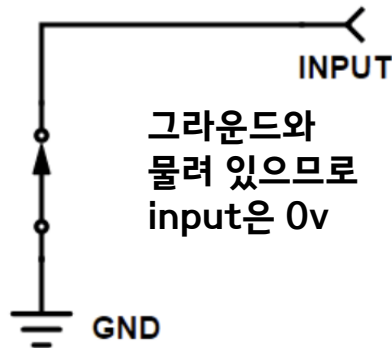
예. PC_8에 3.3v가 걸리면 (입력으로 들어오면), user LED를 켜라



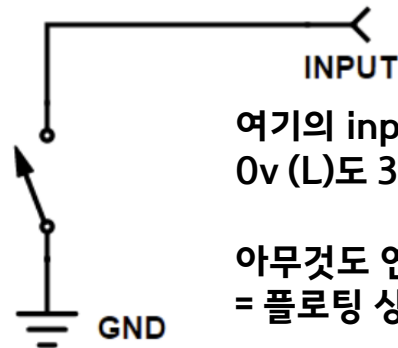
I GPIO: 입력

플로팅 (floating)

- 보통 입력으로 0v (L)또는 3.3 v (H) 를 걸어줌



그라운드와
물려 있으므로
input은 0v



여기의 input은 몇 v인가?
0v (L)도 3.3v (H)도 아닌 상태

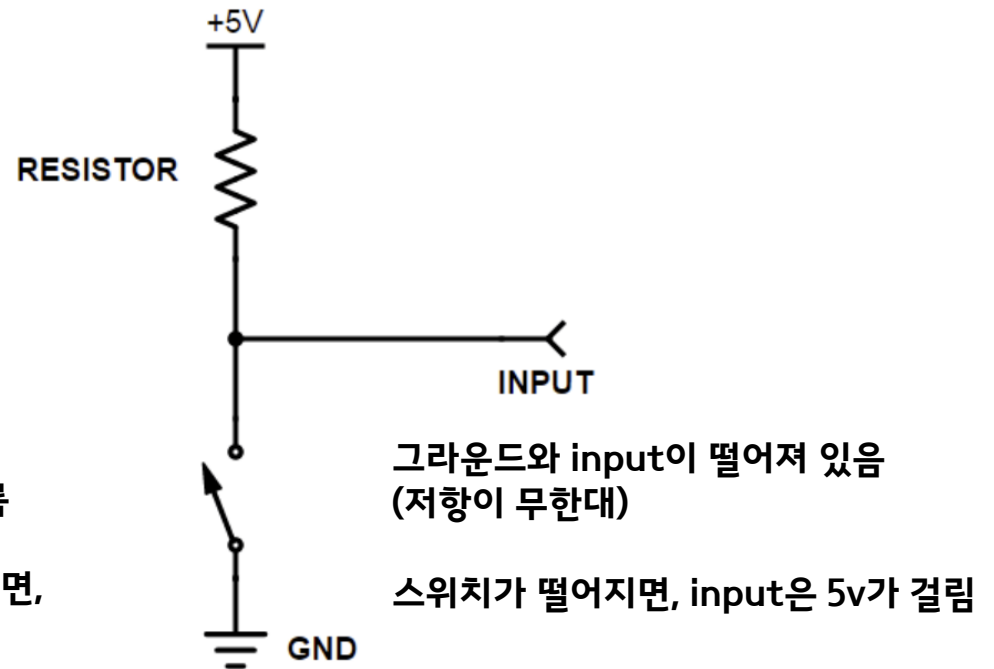
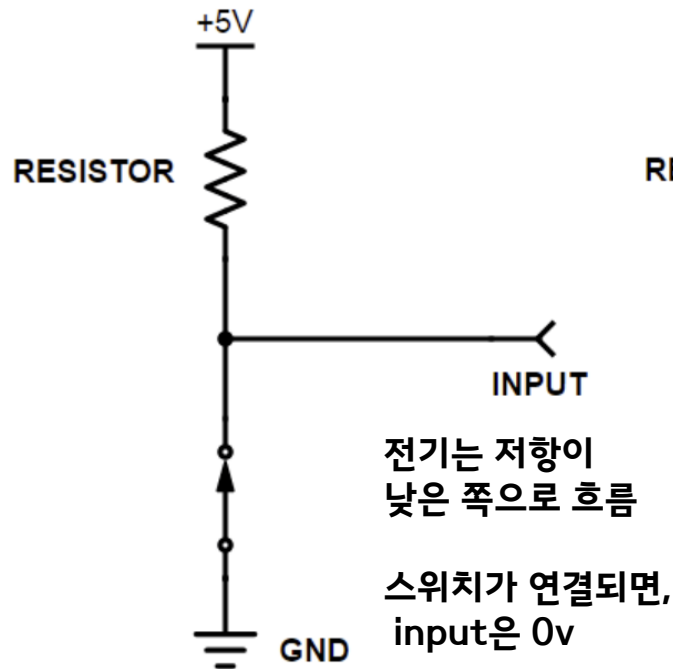
아무것도 연결되지 않을 채 떠있다
= 플로팅 상태

플로팅 상태에서는 아주 작은 노이즈에도
High, Low 사이를 변하기 때문에 오동작을 유발

플로팅은 일반적으로 방지되어야 함!

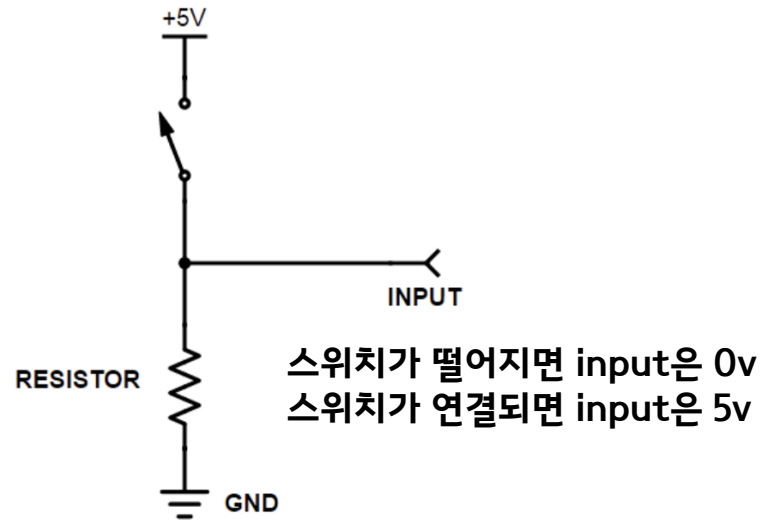
GPIO: 입력

풀업 (Pull-Up)



I GPIO: 입력

풀다운 (Pull-down)



GPIO

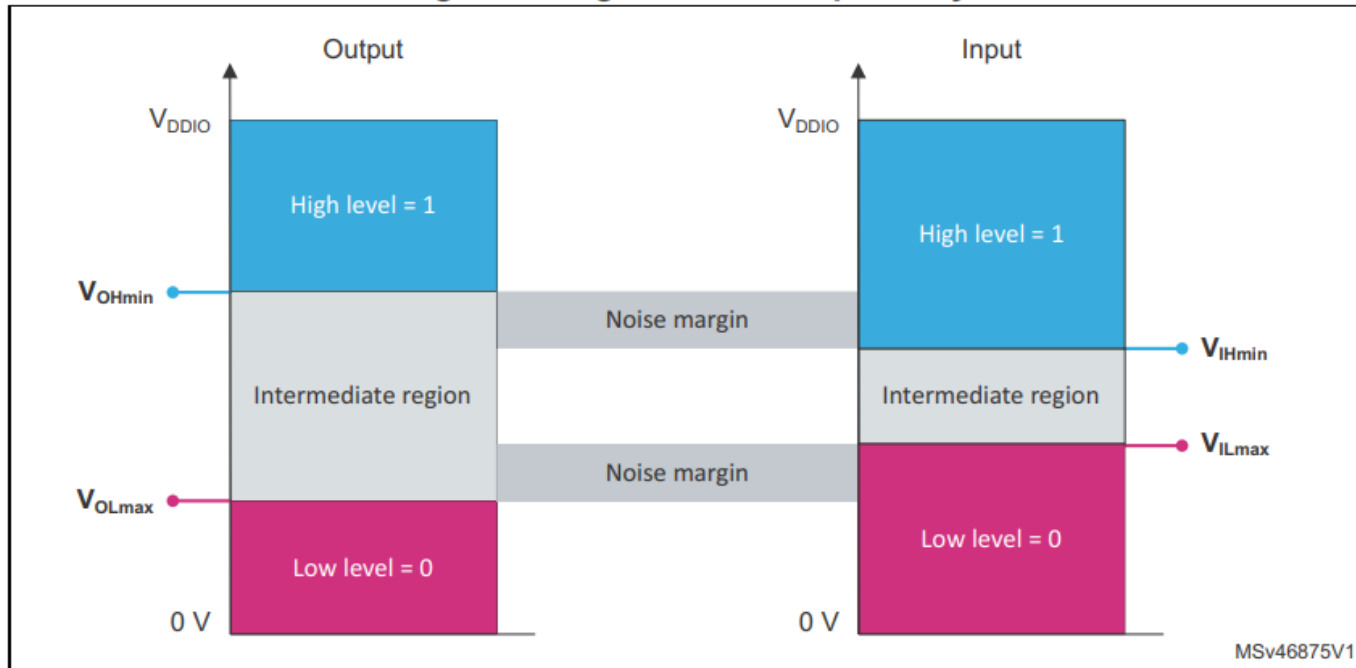
GPIO 입출력 포트의 전기적 특성

- 입력

- $V_{IHmin} = 0.7 * V_{DD}$

- $V_{ILmax} = 0.3 * V_{DD}$

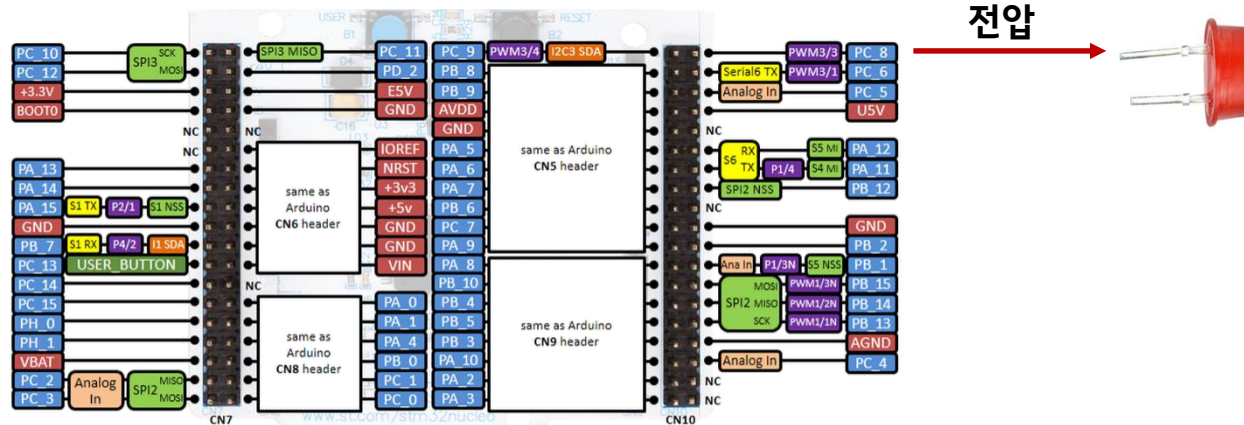
Figure 4. Logical level compatibility



GPIO: 출력

- 포트를 "출력"으로 쓴다면
- MCU의 전압을 해당 포트로 걸어주고, 포트에 연결된 외부 장비에 전압을 인가

예. Count 변수를 증가시키다가, 10보다 커지면, PC_8에 3.3v가 출력되게 하고, PC_8에 연결된 LED가 켜진다

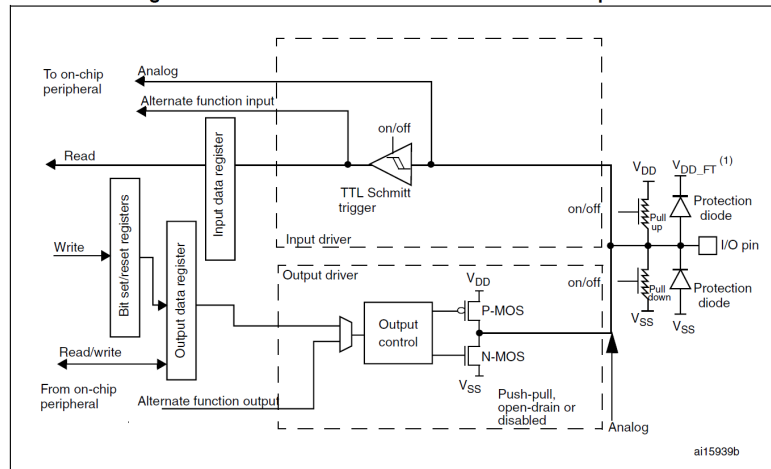


GPIO: 출력

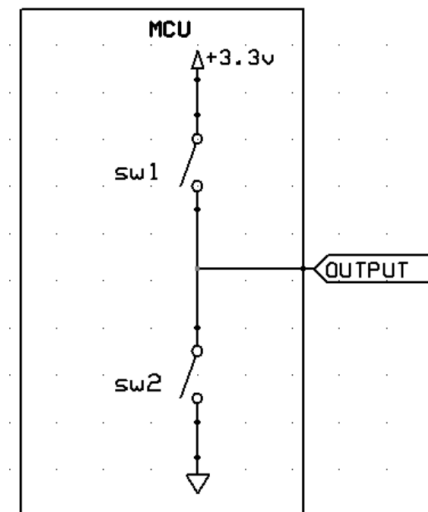
Push-Pull

- 스위치 1이 연결되면, output에 3.3v가 출력
- 스위치 2가 연결되면, output에 0v가 출력
- Push-Pull은 MCU내부에 스위치 2개가 logic high, logic low를 만들어 줌

Figure 25. Basic structure of a five-volt tolerant I/O port bit



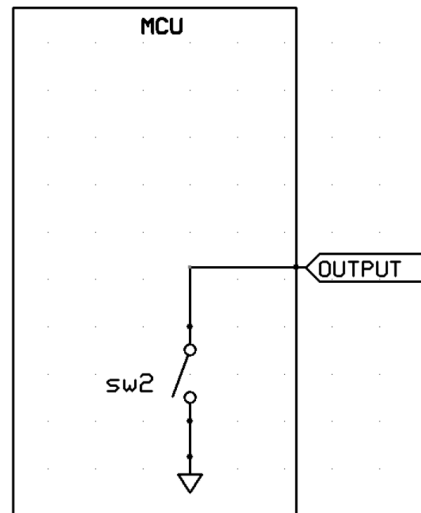
1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD} .



I GPIO: 출력

Open-drain

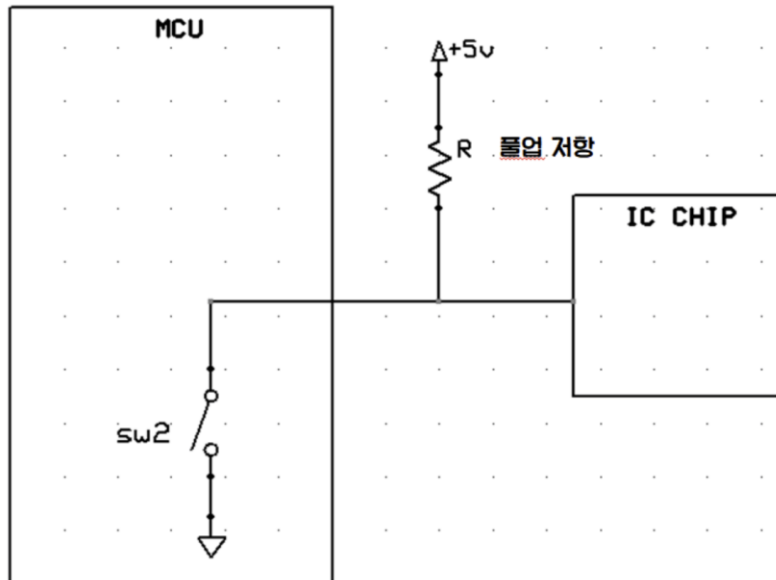
- 스위치 2가 연결되면 output에 0v가 출력되고, 스위치 2가 떨어지면 output이 플로팅 (floating) 상태가 됨



I GPIO: 출력

Open-drain을 쓰는 이유

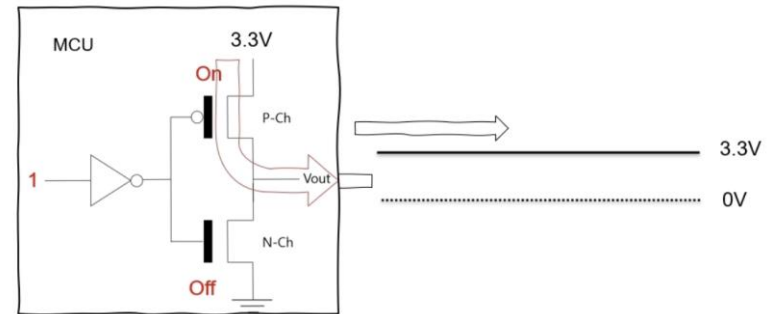
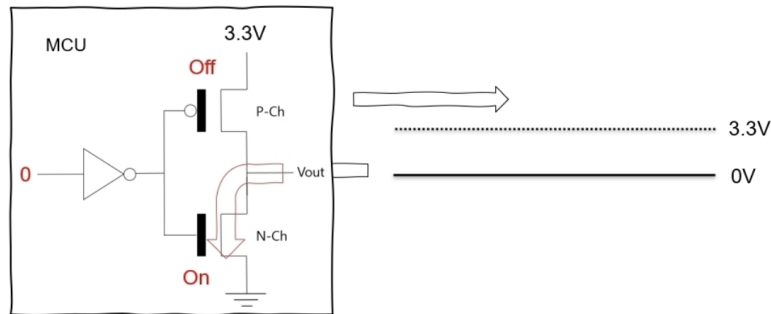
- Level converter로 사용하기 위해
- 일반적으로 MCU는 3.3v를 출력하는데, 외부에 연결될 칩의 구동 전압이 5v 일 경우 외부에 pull-up저항을 달면 logic high가 3.3v가 아닌 5v가 되어 칩이 동작할 수 있게 됨



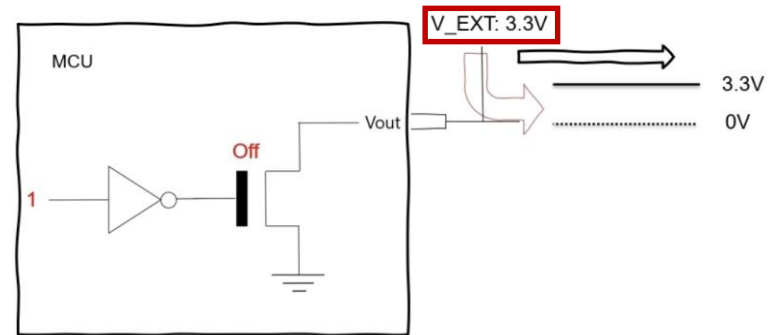
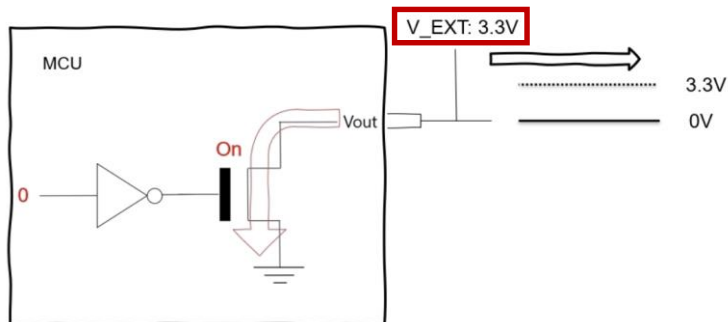
스위치 2가 떨어지면 칩에 5v가 공급되고,
스위치 2가 연결되면 칩에 0v가 공급됨

GPIO: 출력

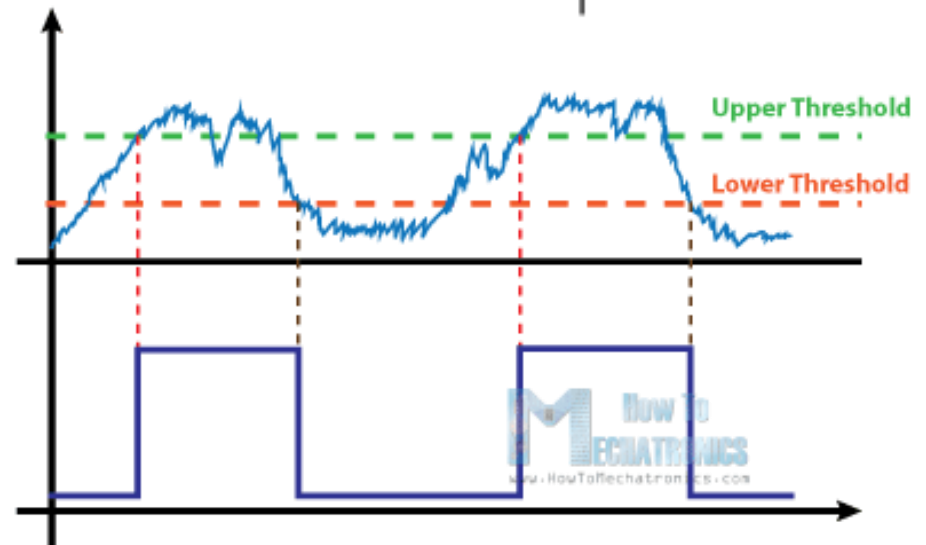
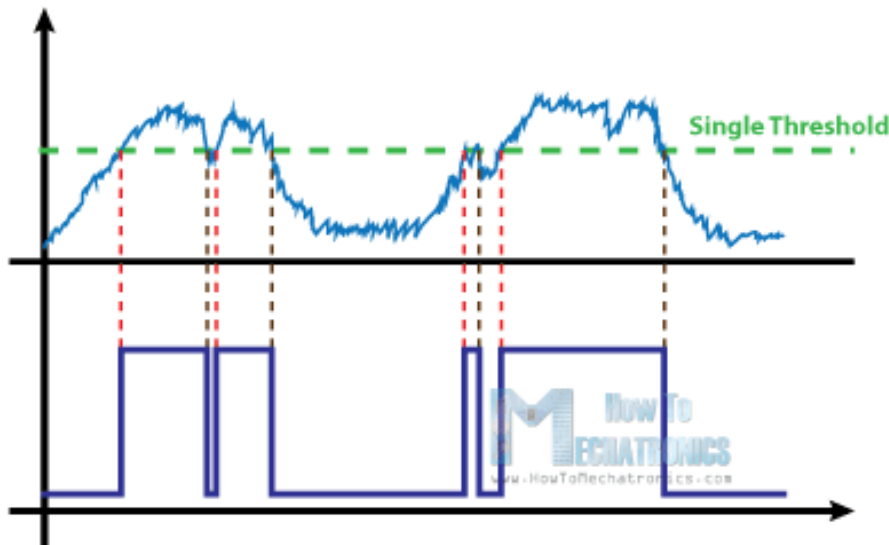
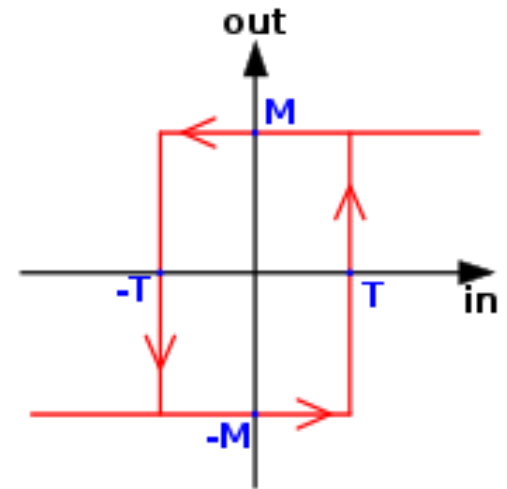
Push-pull 모드



Open-drain 모드



- Schmitt trigger
 - 잡음에 강인한 디지털 값을 출력



I GPIO

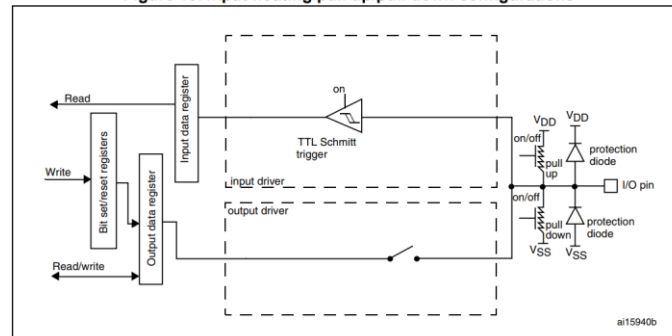
- 우리는 어떤 GPIO 포트를 어떻게 사용할 것인지를 MCU에게 알려주어야 함
(= 프로그래밍 + HW 구성)
- 어떤 포트 (예. Port A: GPIOA, Port B: GPIOB, ...)
- 몇 번 핀 (예. Port A의 1번 핀 GPIOA_Pin_1)
- In / Out / AF/ Analog
- Pull-up / pull-down / floating
- Push-pull / Open-drain
- Clock 공급
- ...

GPIO

Input configuration

- When the I/O port is programmed as Input:
 - the output buffer is disabled
 - the Schmitt trigger input is activated
 - the pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register
 - The data present on the I/O pin are sampled into the input data register every AHB clock cycle
 - A read access to the input data register provides the I/O State

Figure 18. Input floating/pull up/pull down configurations



Output configuration

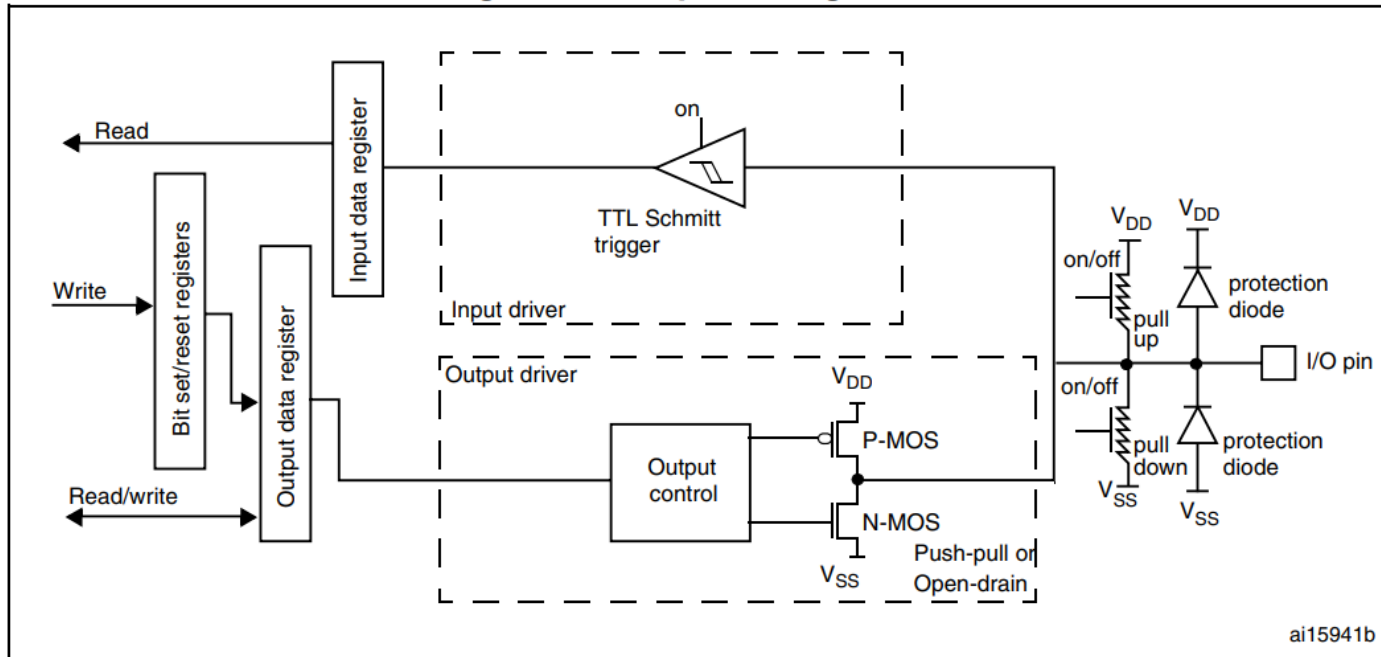
- When the I/O port is programmed as output:
 - The output buffer is enabled:
 - Open drain mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register leaves the port in Hi-Z (the P-MOS is never activated)
 - Push-pull mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register activates the P-MOS
 - The Schmitt trigger input is activated
 - The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx_PUPDR register
 - The data present on the I/O pin are sampled into the input data register every AHB clock cycle
 - A read access to the input data register gets the I/O state
 - A read access to the output data register gets the last written value

I GPIO

Output configuration

- When the I/O port is programmed as output:

Figure 19. Output configuration



Alternate function configuration

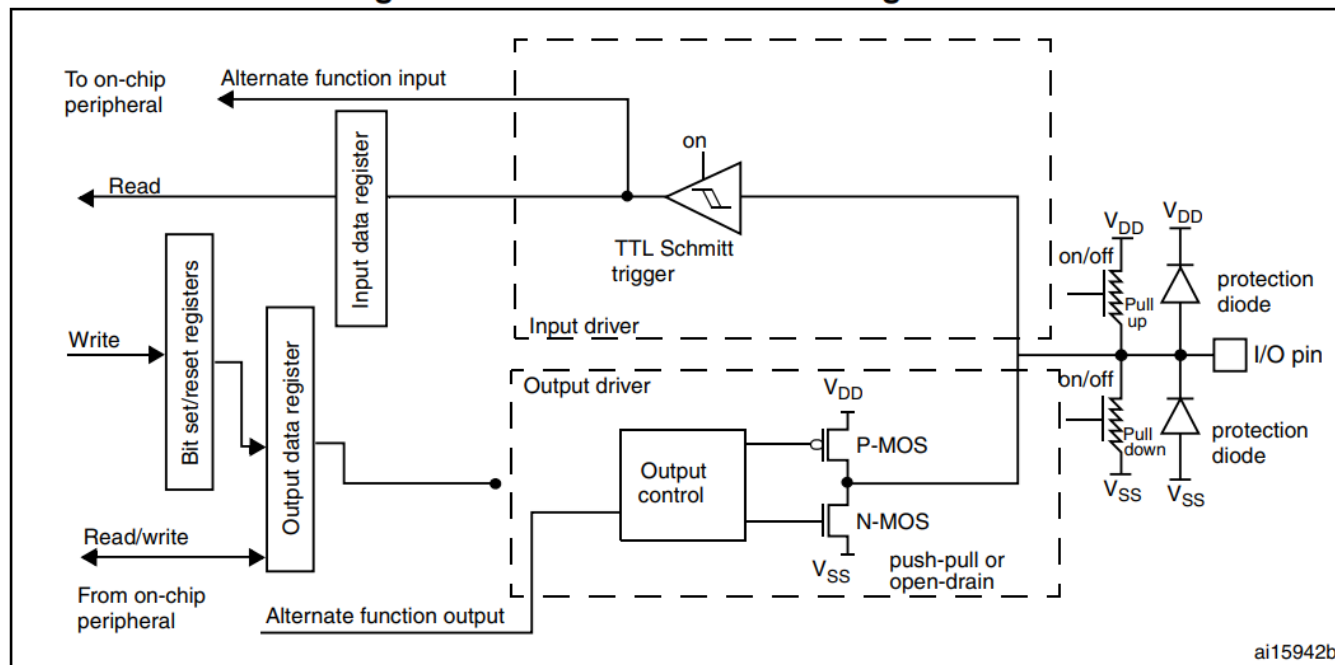
- When the I/O port is programmed as alternate function:
 - The output buffer can be configured as open-drain or push-pull
 - The output buffer is driven by the signal coming from the peripheral (transmitter enable and data)
 - The Schmitt trigger input is activated
 - The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx_PUPDR register
 - The data present on the I/O pin are sampled into the input data register every AHB clock cycle
 - A read access to the input data register gets the I/O state

GPIO

Alternate function configuration

- When the I/O port is programmed as alternate function:

Figure 20. Alternate function configuration



Analog configuration

- When the I/O port is programmed as analog configuration:
 - The output buffer is disabled
 - The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
 - The weak pull-up and pull-down resistors are disabled
 - Read access to the input data register gets the value "0"

GPIO

Port bit configuration table

- GP = general-purpose, PP = push-pull, PU = pull-up
- PD = pull-down, OD = open-drain, AF = alternate function

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [B:A]	PUPDR(i) [1:0]		I/O configuration	
01	0	SPEED [B:A]	0	0	GP output	PP
	0		0	1	GP output	PP + PU
	0		1	0	GP output	PP + PD
	0		1	1	Reserved	
	1		0	0	GP output	OD
	1		0	1	GP output	OD + PU
	1		1	0	GP output	OD + PD
	1		1	1	Reserved (GP output OD)	

Port bit configuration table

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [B:A]		PUPDR(i) [1:0]		I/O configuration	
10	0	SPEED [B:A]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0		
	x	x	x	1	1		

GPIO

GPIO registers

- GPIO port mode register (GPIOx_MODER) (x = A..I/J/K)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **MODERy[1:0]**: Port x configuration bits (y = 0..15)

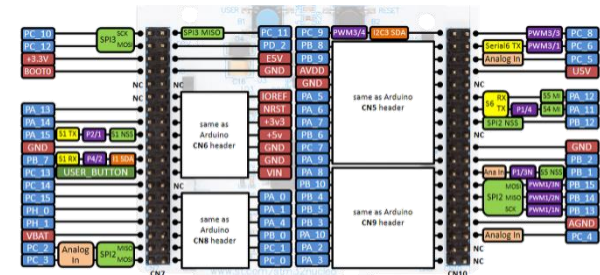
These bits are written by software to configure the I/O direction mode.

00: Input (reset state)

01: General purpose output mode

10: Alternate function mode

11: Analog mode



I GPIO

GPIO registers

- GPIO port output type register (GPIOx_OTYPER) (x = A..I/J/K)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OTy**: Port x configuration bits (y = 0..15)

These bits are written by software to configure the output type of the I/O port.

0: Output push-pull (reset state)

1: Output open-drain

I GPIO

GPIO registers

- GPIO port output speed register (GPIOx_OSPEEDR)(x = A..I/J/K)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15 [1:0]		OSPEEDR14 [1:0]		OSPEEDR13 [1:0]		OSPEEDR12 [1:0]		OSPEEDR11 [1:0]		OSPEEDR10 [1:0]		OSPEEDR9 [1:0]		OSPEEDR8 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1 [1:0]		OSPEEDR0 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits $2y:2y+1$ **OSPEEDR y [1:0]**: Port x configuration bits ($y = 0..15$)

These bits are written by software to configure the I/O output speed.

00: Low speed

01: Medium speed

10: High speed

11: Very high speed

Note: Refer to the product datasheets for the values of OSPEEDR y bits versus VDD range and external load.

GPIO

GPIO registers

- GPIO port pull-up/pull-down register (GPIOx_PUPDR)(x = A..I/J/K)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **PUPDRy[1:0]**: Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O pull-up or pull-down

00: No pull-up, pull-down

01: Pull-up

10: Pull-down

11: Reserved

I GPIO

GPIO registers

- GPIO port input data register (GPIOx_IDR) (x = A..I/J/K)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **IDRy**: Port input data (y = 0..15)

These bits are read-only and can be accessed in word mode only. They contain the input value of the corresponding I/O port.

GPIO

GPIO registers

- GPIO port output data register (GPIOx_ODR) (x = A..I/J/K)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODRy**: Port output data (y = 0..15)

These bits can be read and written by software.

Note: For atomic bit set/reset, the ODR bits can be individually set and reset by writing to the GPIOx_BSRR register (x = A..I/J/K).

I GPIO

GPIO registers

- GPIO port bit set/reset register (GPIOx_BSRR) (x = A..I/J/K)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy**: Port x reset bit y (y = 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Resets the corresponding ODRx bit

Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BSy**: Port x set bit y (y= 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

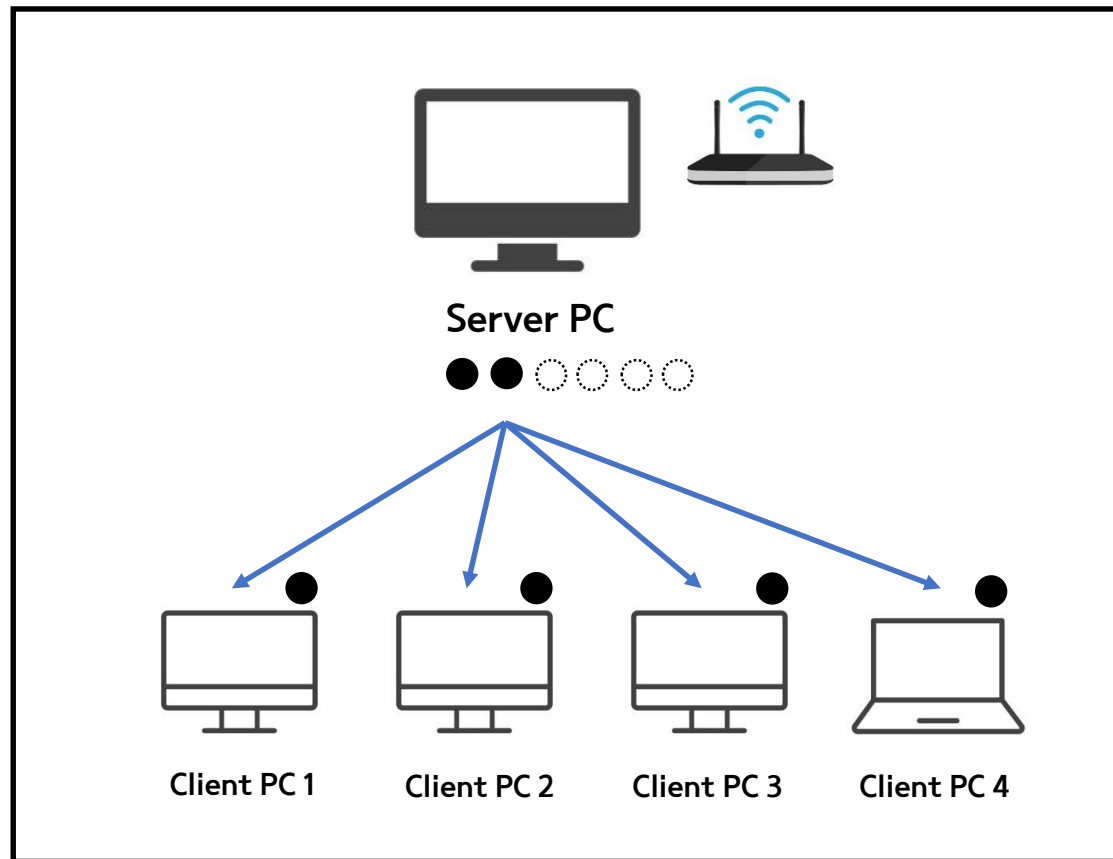
0: No action on the corresponding ODRx bit

1: Sets the corresponding ODRx bit

IAR Embedded Workbench for Arm

교육용 정책

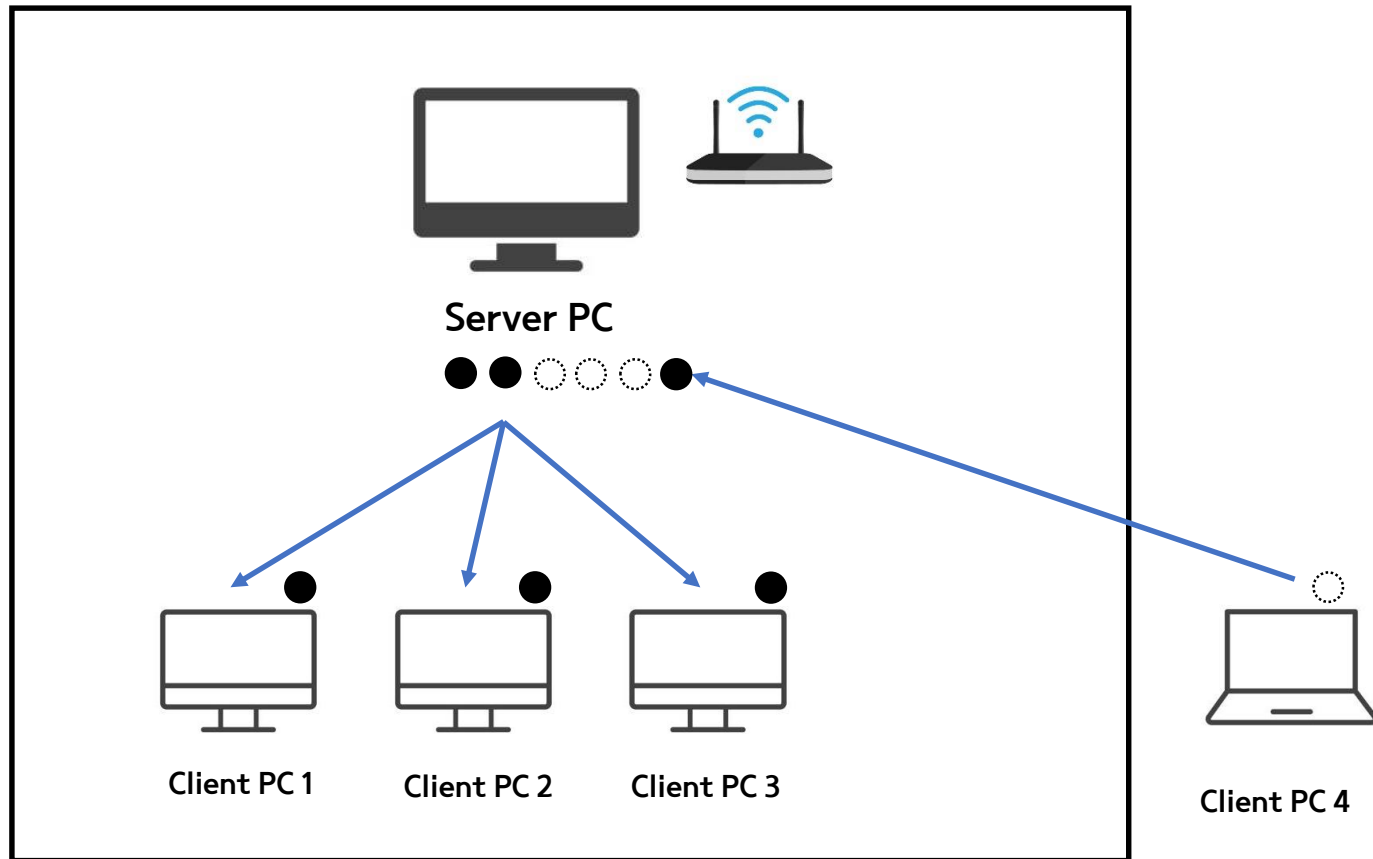
- 망내 client PC가 프로그램을 실행시키면, server PC가 client PC에 라이선스를 할당



IAR Embedded Workbench for Arm

교육용 정책

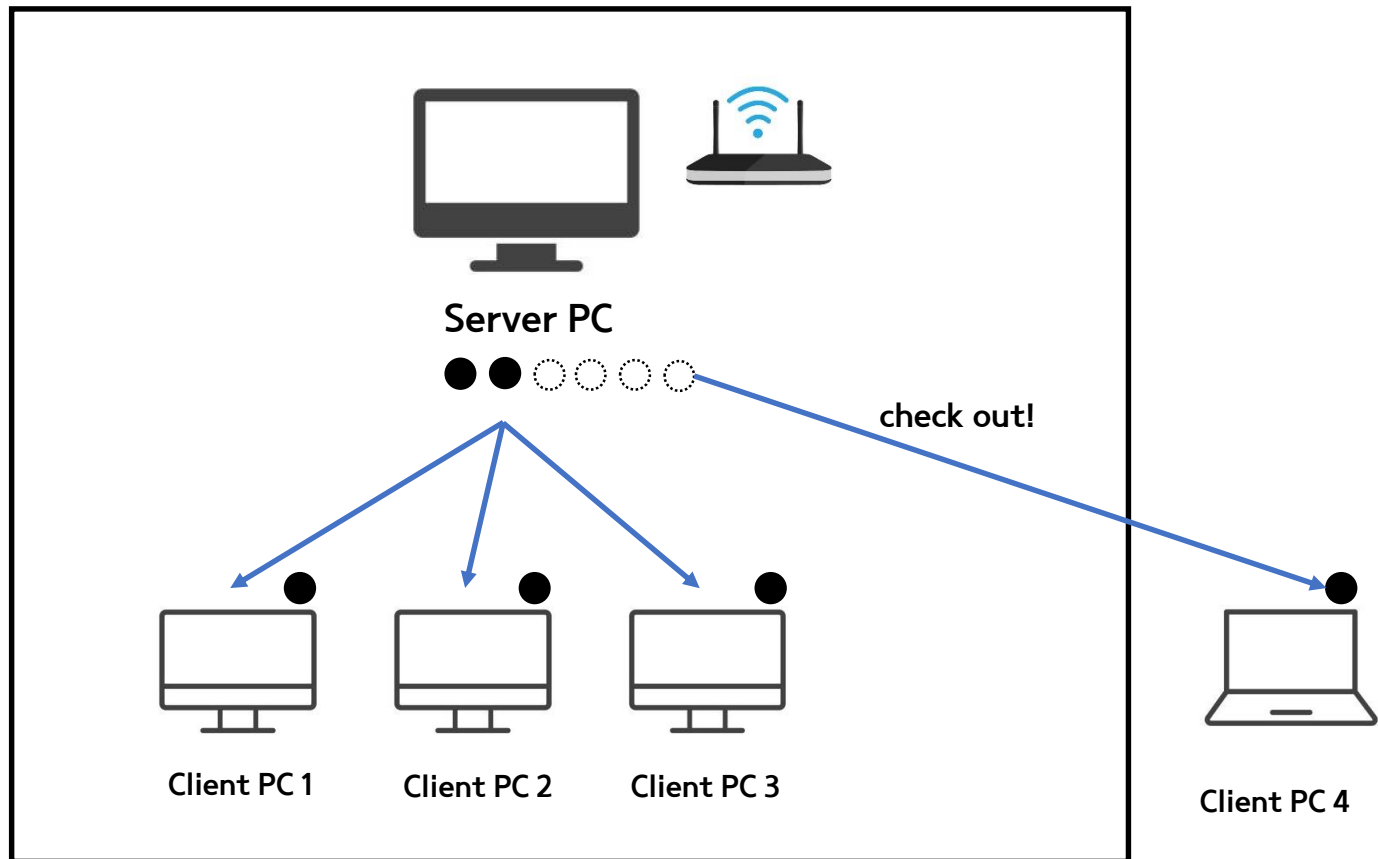
- Client PC가 망을 벗어나면 30분 뒤 라이선스를 반납



IAR Embedded Workbench for Arm

교육용 정책

- “check out”이라는 기능이 있고, 망 외에서 최대 15일까지 프로그램 실행이 가능



IAR Embedded Workbench for Arm

실습 및 과제

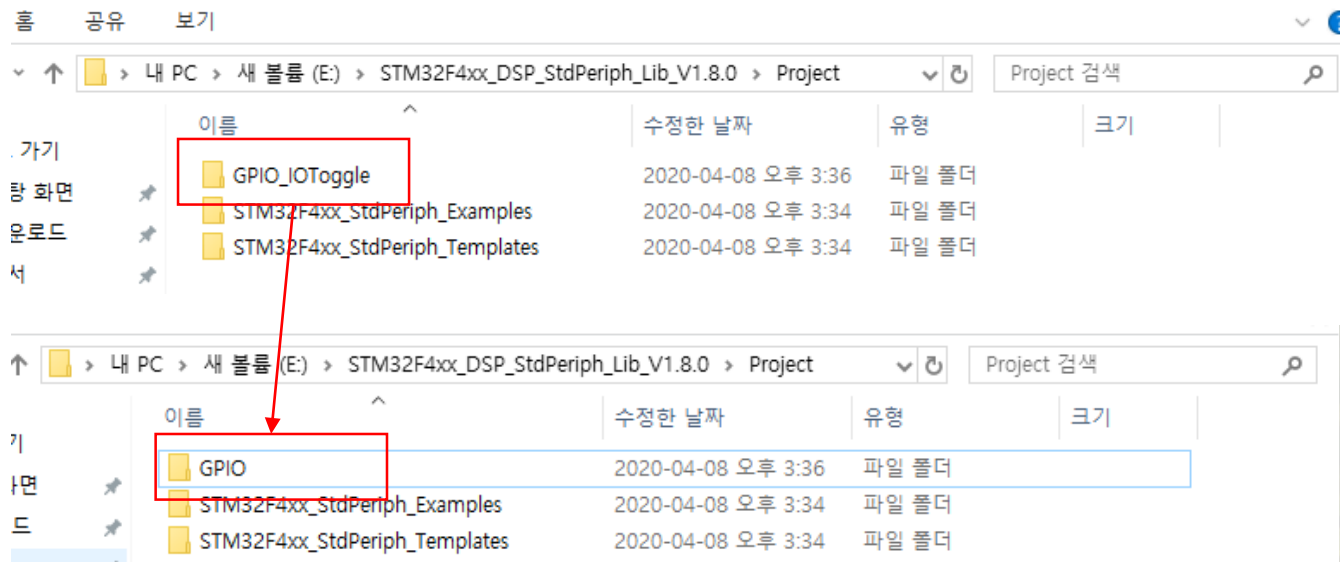
- 노트북으로 진행
- 노트북으로 R312호 wifi 연결
- 프로그램 실행 & 실습
- 수업 종료 후, check out
- 집에서 과제 수행
- 다음 수업에서 노트북으로 R312호 wifi 연결
- 프로그램 실행 & check in & 실습
- ...

I 실습환경 구축

- SPL (Standard Peripheral Libraries) 다운로드
 - <https://www.st.com/en/embedded-software/stsw-stm32065.html>
 - 이메일 인증 필요

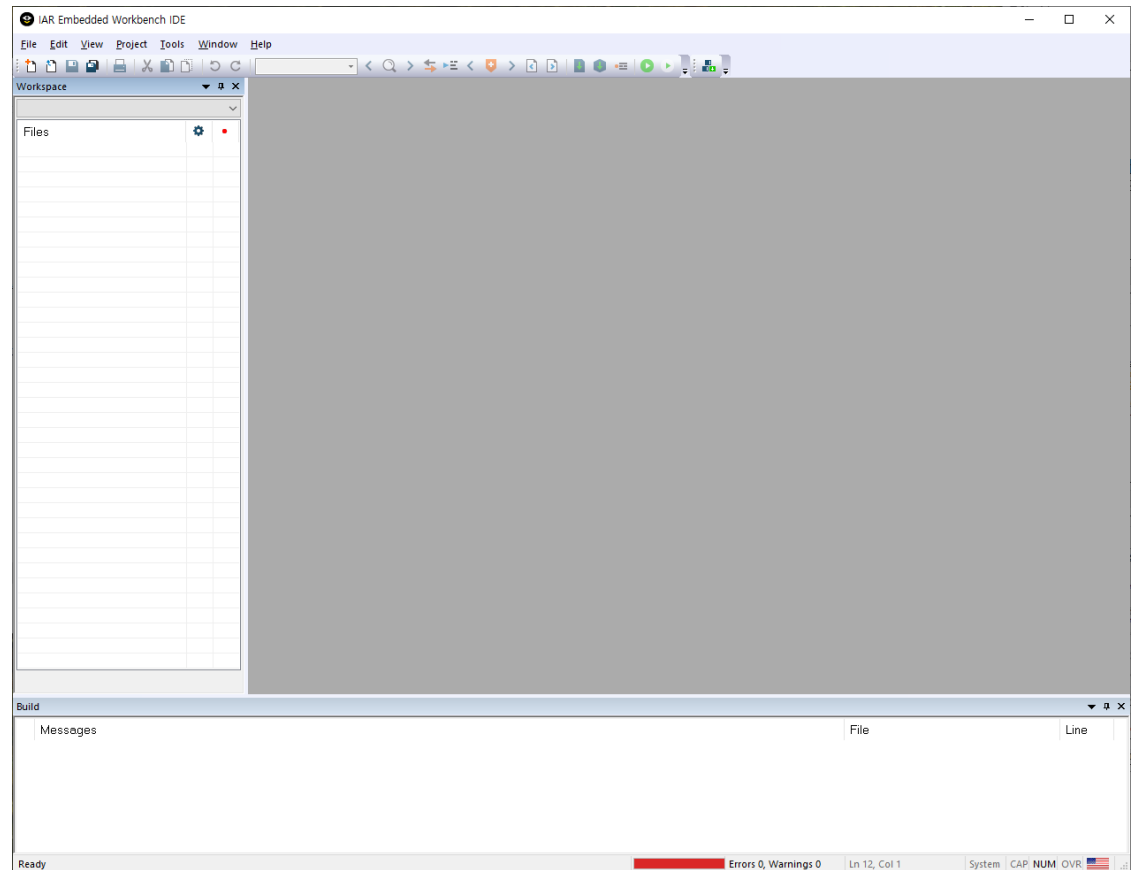
실습환경 구축

- SPL 압축해제 후 다음 폴더 복사
 - STM32F4xx_DSP_StdPeriph_Lib_V1.8.0\Project\STM32F4xx_StdPeriph_Examples\GPIO\GPIO_IOToggle
- 아래 폴더에 붙여넣기 후 폴더명 변경
 - STM32F4xx_DSP_StdPeriph_Lib_V1.8.0\Project



실습환경 구축

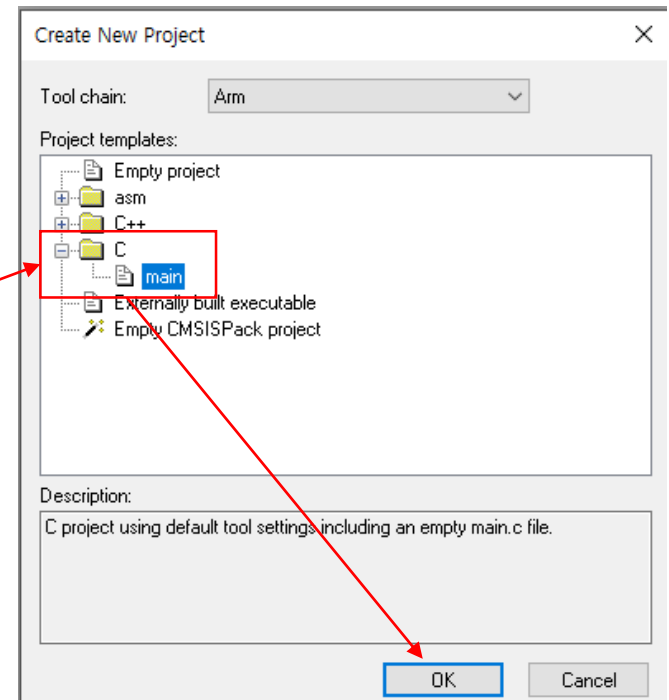
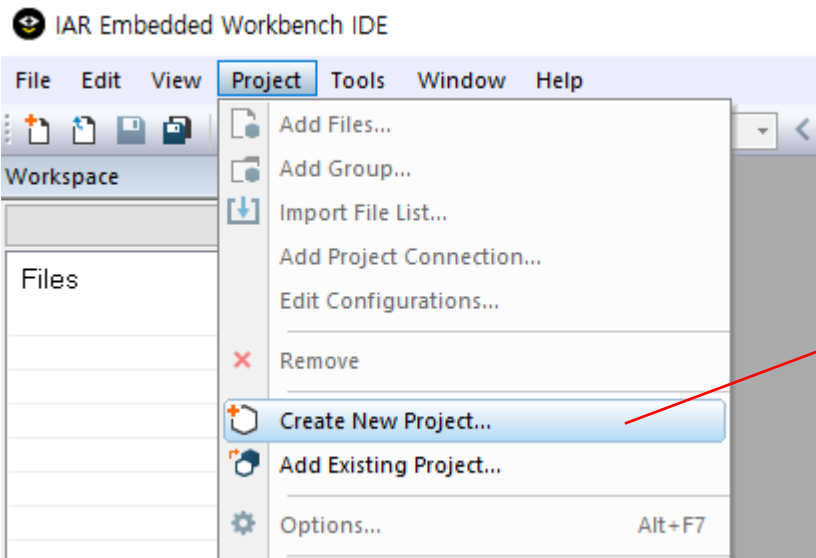
- EWARM 실행
 - File → New Workspace



실습환경 구축

• EARM 실행

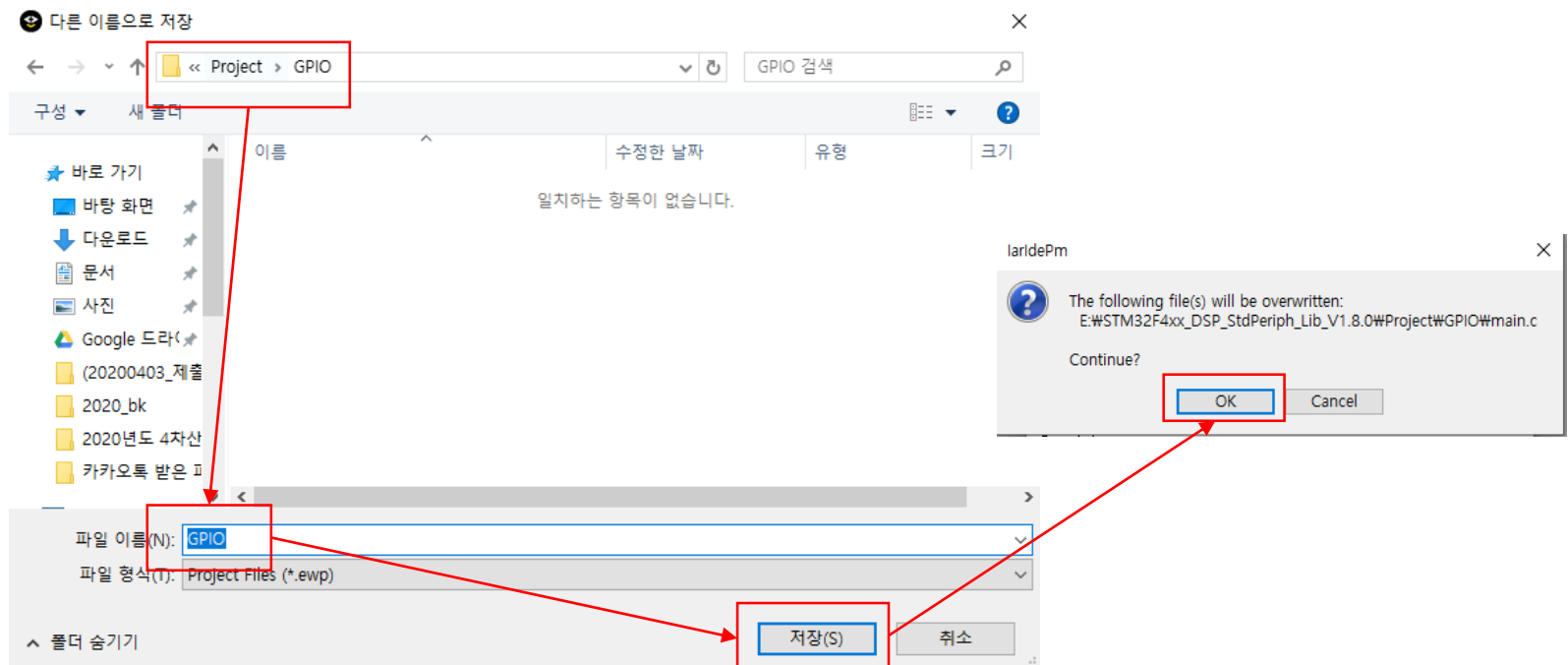
- Project → Create New Project
- Create New Project 창에서 C 아래 main을 클릭하고 "OK"



실습환경 구축

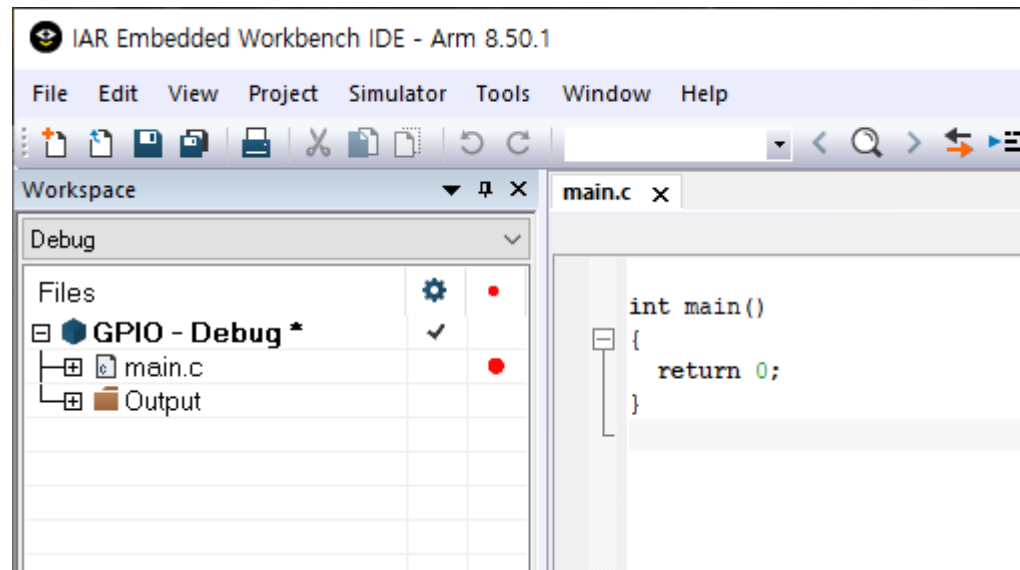
프로젝트 이름 생성

- 만들어진 "GPIO" 폴더를 선택하고 프로젝트 이름을 "GPIO"로 설정 후 저장
- GPIO.ewp 파일 생성
- 기존에 있던 main.c파일은 덮어쓰



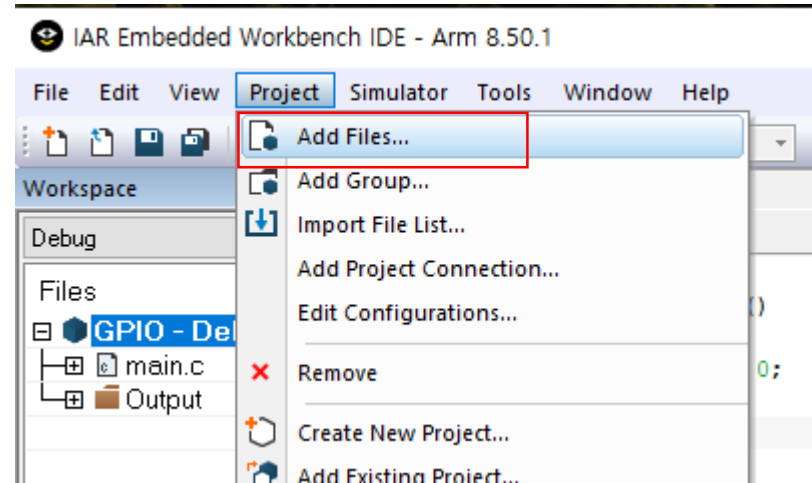
실습환경 구축

- 프로젝트 생성
 - 아래와 같이 GPIO프로젝트가 생성됨



실습환경 구축

- 프로젝트 파일 추가
 - Project → Add Files 클릭

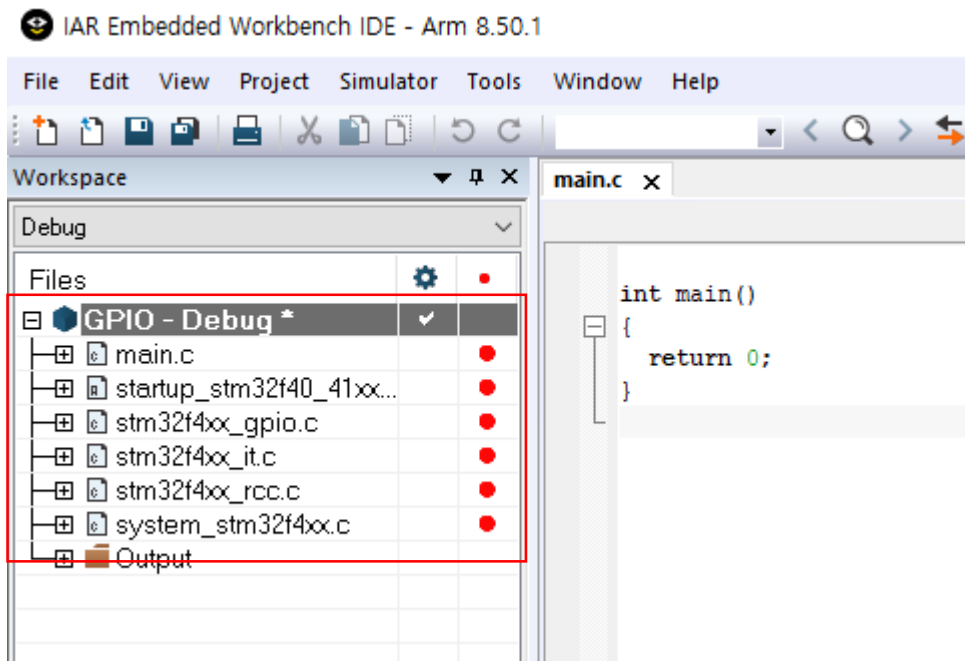


- 아래 파일 추가

경로	파일명
STM32F4xx_DSP_StdPeriph_Lib_V1.8.0\Project\GPIO	system_stm32f4xx.c
STM32F4xx_DSP_StdPeriph_Lib_V1.8.0\Project\GPIO	stm32f4xx_it.c
STM32F4xx_DSP_StdPeriph_Lib_V1.8.0\Libraries\CMSIS\Device\ST\STM32F4xx\Source\Templates\iar	startup_stm32f40_41xxx.s
STM32F4xx_DSP_StdPeriph_Lib_V1.8.0\Libraries\STM32F4xx_StdPeriph_Driver\src	stm32f4xx_gpio.c
STM32F4xx_DSP_StdPeriph_Lib_V1.8.0\Libraries\STM32F4xx_StdPeriph_Driver\src	stm32f4xx_rcc.c

실습환경 구축

- 프로젝트 파일 추가
 - 추가된 파일 확인

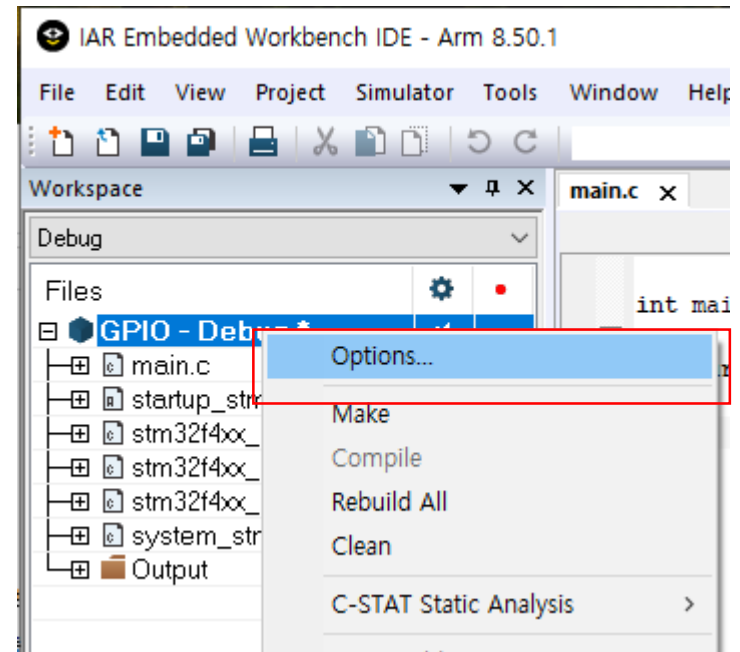
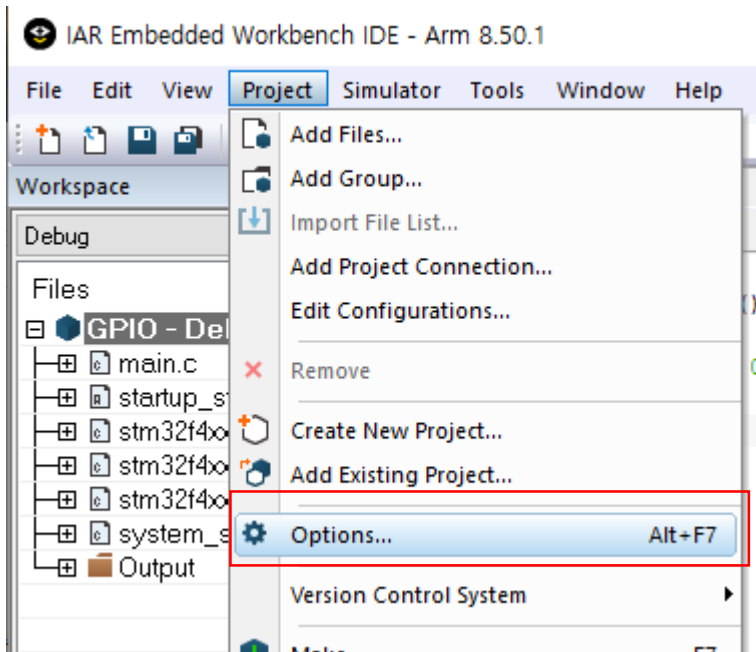


실습환경 구축

- 프로젝트 옵션 설정

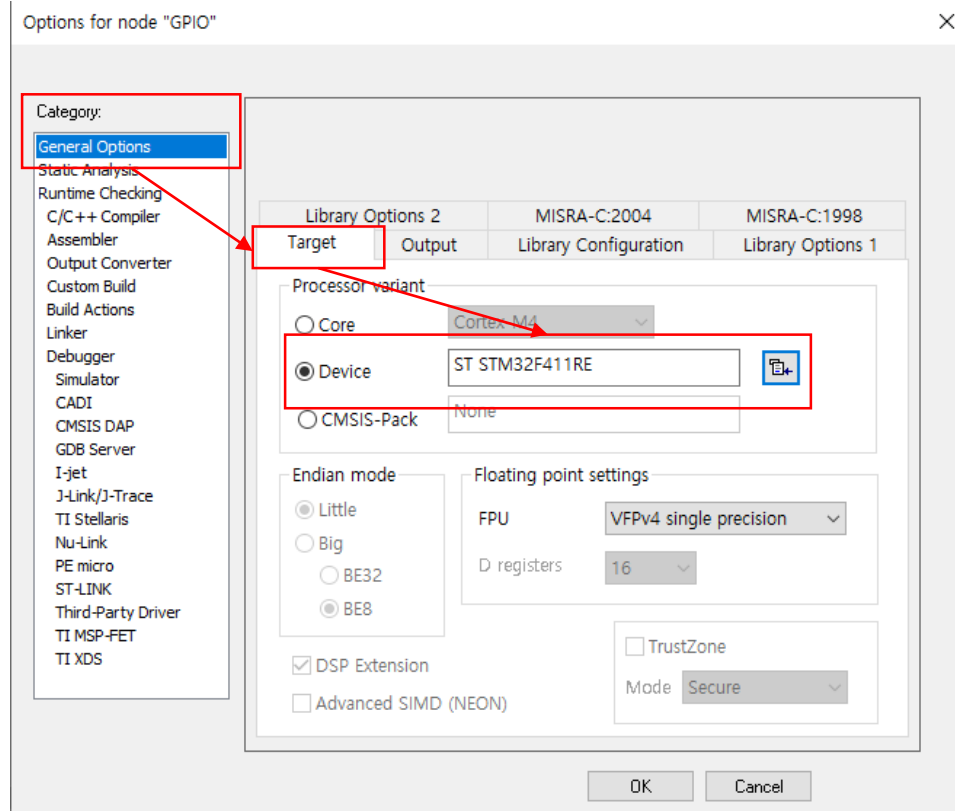
- Project → Options

- 또는 Workspace 상의 프로젝트명 위에서 우 클릭 후 Options 클릭



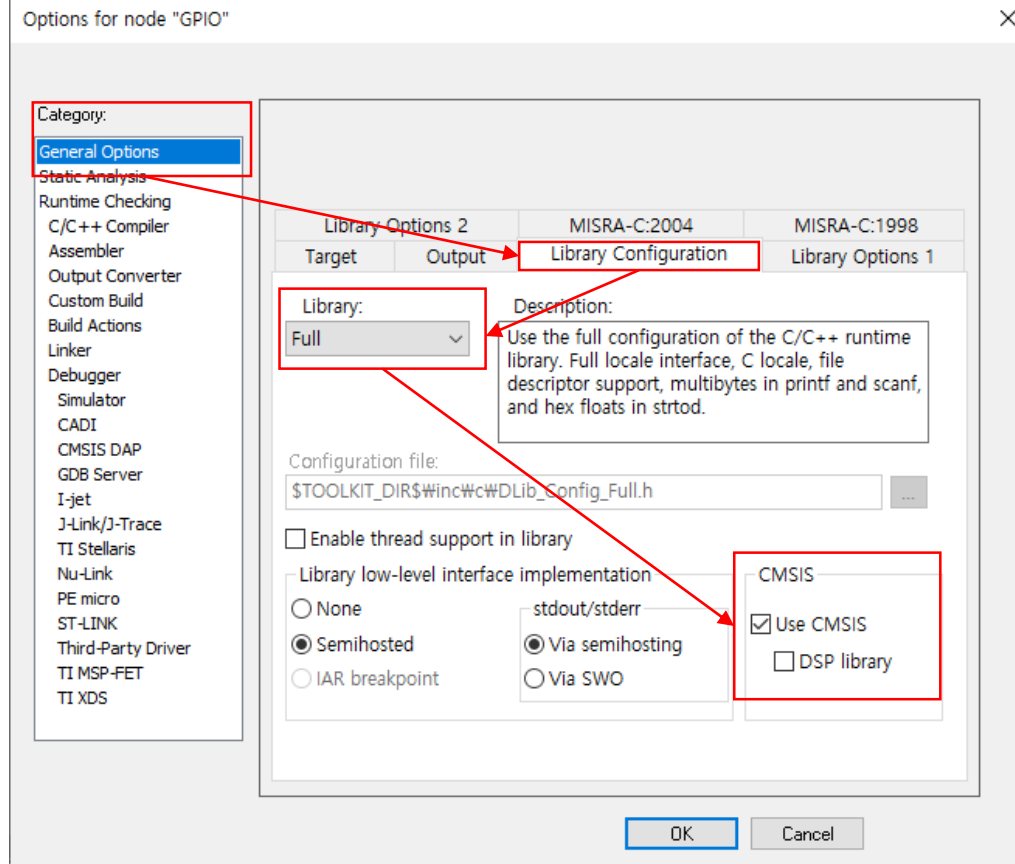
실습환경 구축

- 프로젝트 옵션 설정
 - Category: General Options
 - Tab: Target
 - 사용할 MCU를 설정
 - 오른쪽 MCU선택 버튼을 누르고
 - ST → STM32F4
 - STM32F411 → STM32F411RE



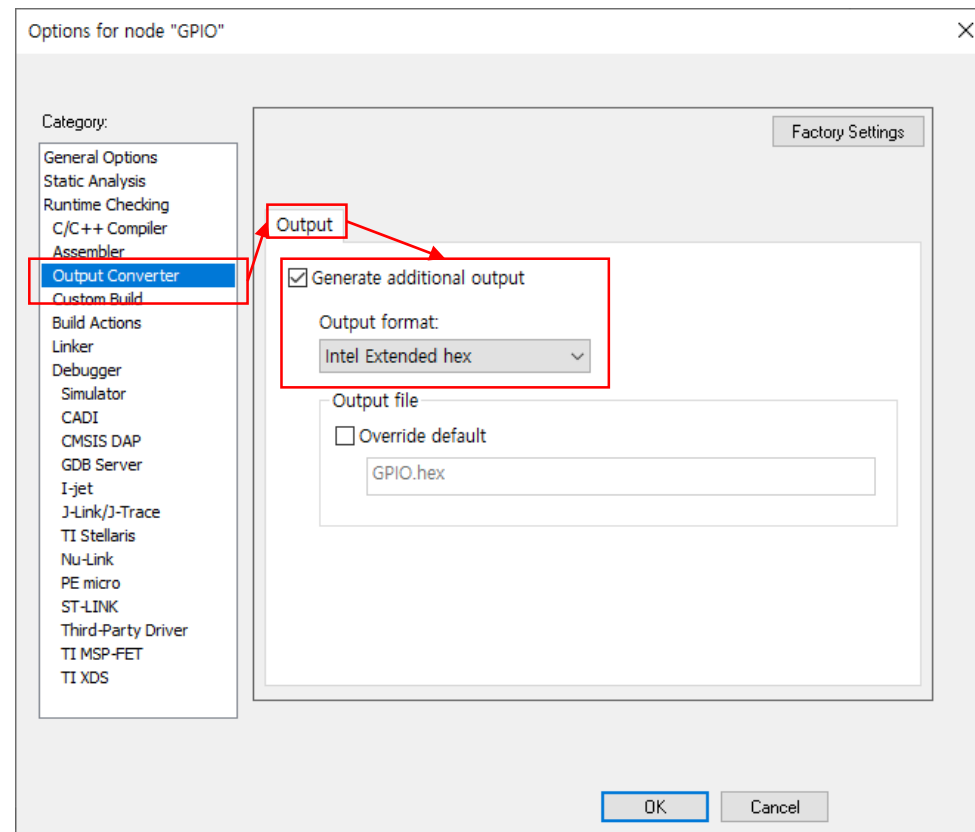
실습환경 구축

- 프로젝트 옵션 설정
 - Category: General Options
 - Tab: Library Configuration
 - Library: Full
 - CMSIS: Use CMSIS



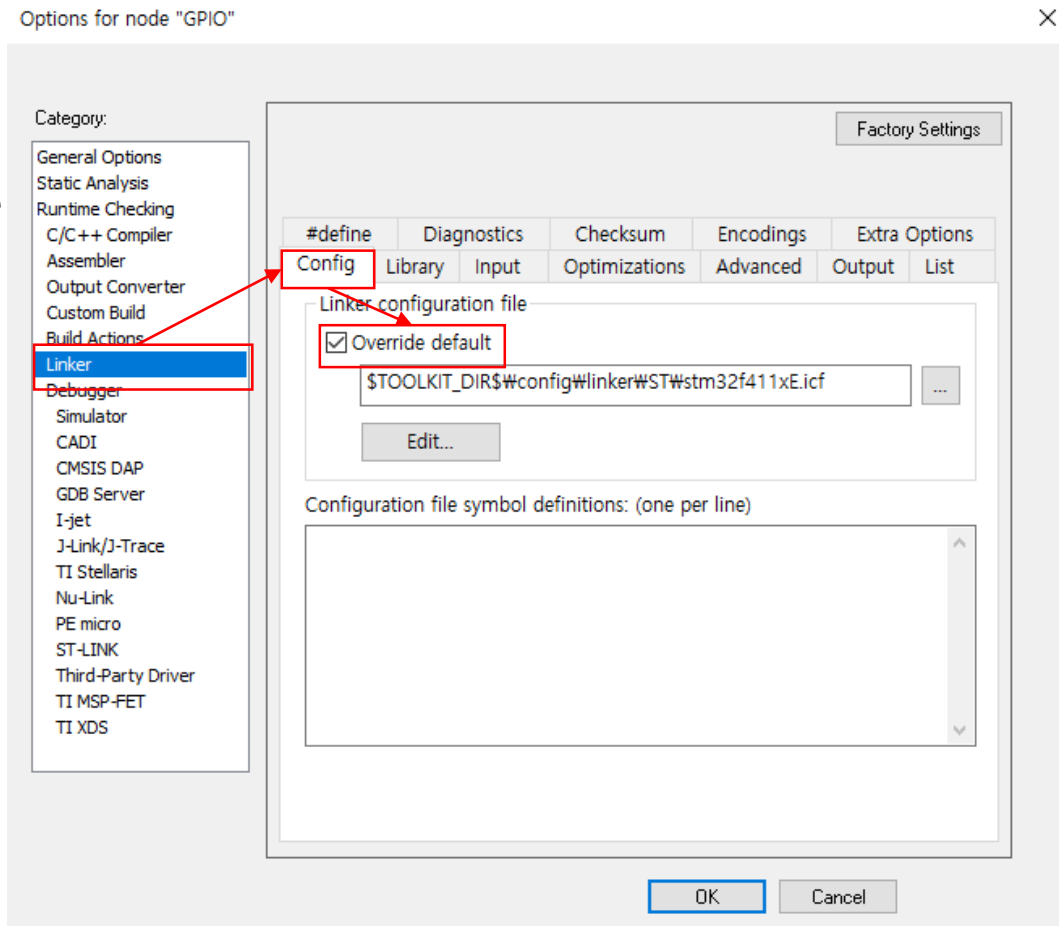
실습환경 구축

- 프로젝트 옵션 설정
 - Category: Output Converter
 - Tab: Output
 - Generate additional output
 - Output format:
Intel Extended hex



실습환경 구축

- 프로젝트 옵션 설정
 - Category: Linker
 - Tab: Config
 - Override default: Enable



실습환경 구축

- 프로젝트 옵션 설정

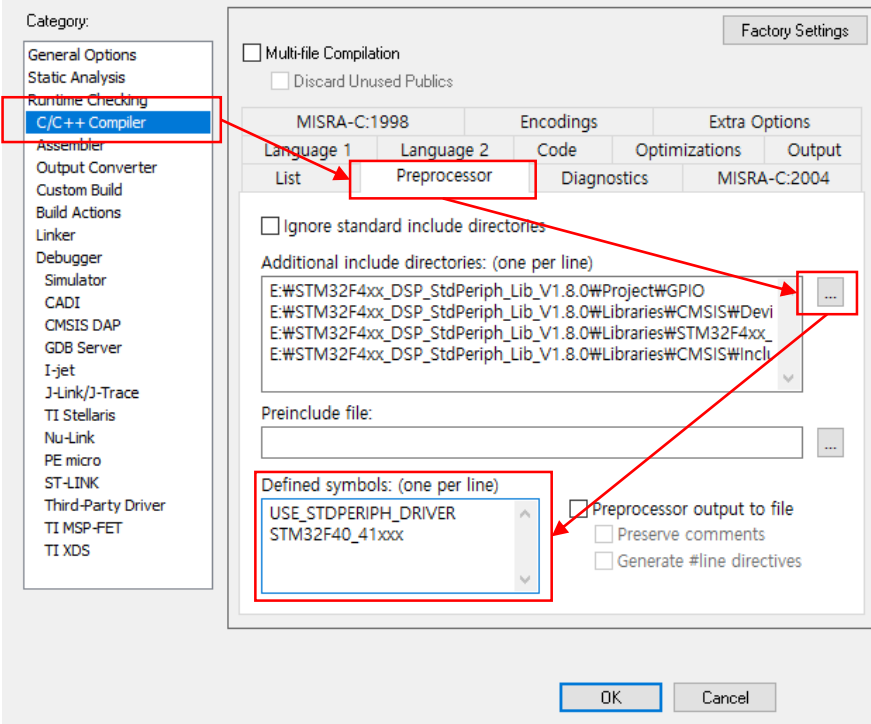
- Category: C/C++ Compiler
- Tab: Preprocessor
- Defined symbols: 아래 심벌 작성
- USE_STDPERIPH_DRIVER
- STM32F40_41xxx

- Additional include directories

- 아래 4개 경로 추가

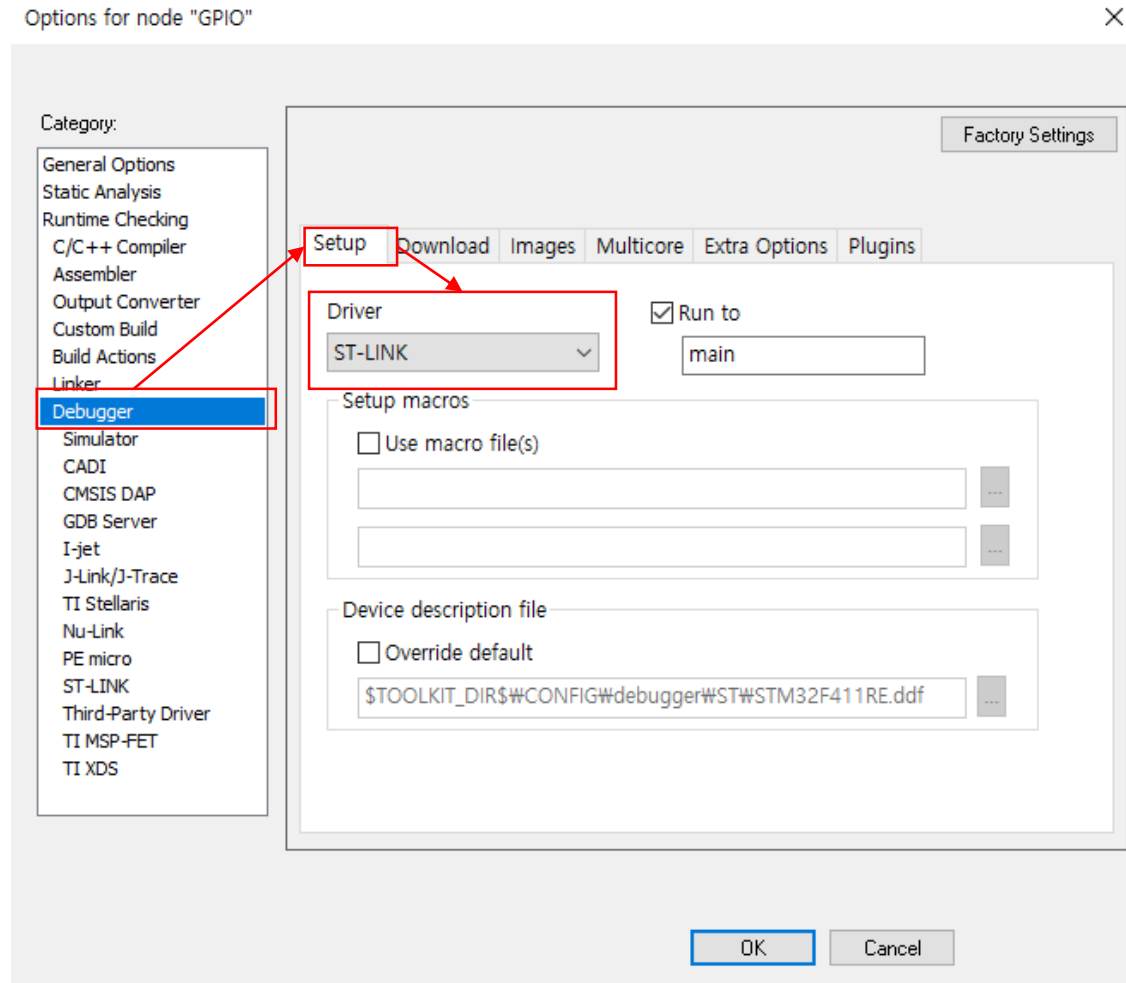
- STM32F4xx_DSP_StdPeriph_Lib_V1.8.0\Project\GPIO
- STM32F4xx_DSP_StdPeriph_Lib_V1.8.0\Libraries\CMSIS\Device\ST\STM32F4xx\Include
- STM32F4xx_DSP_StdPeriph_Lib_V1.8.0\Libraries\STM32F4xx_StdPeriph_Driver\Inc
- STM32F4xx_DSP_StdPeriph_Lib_V1.8.0\Libraries\CMSIS\Include

Options for node "GPIO"



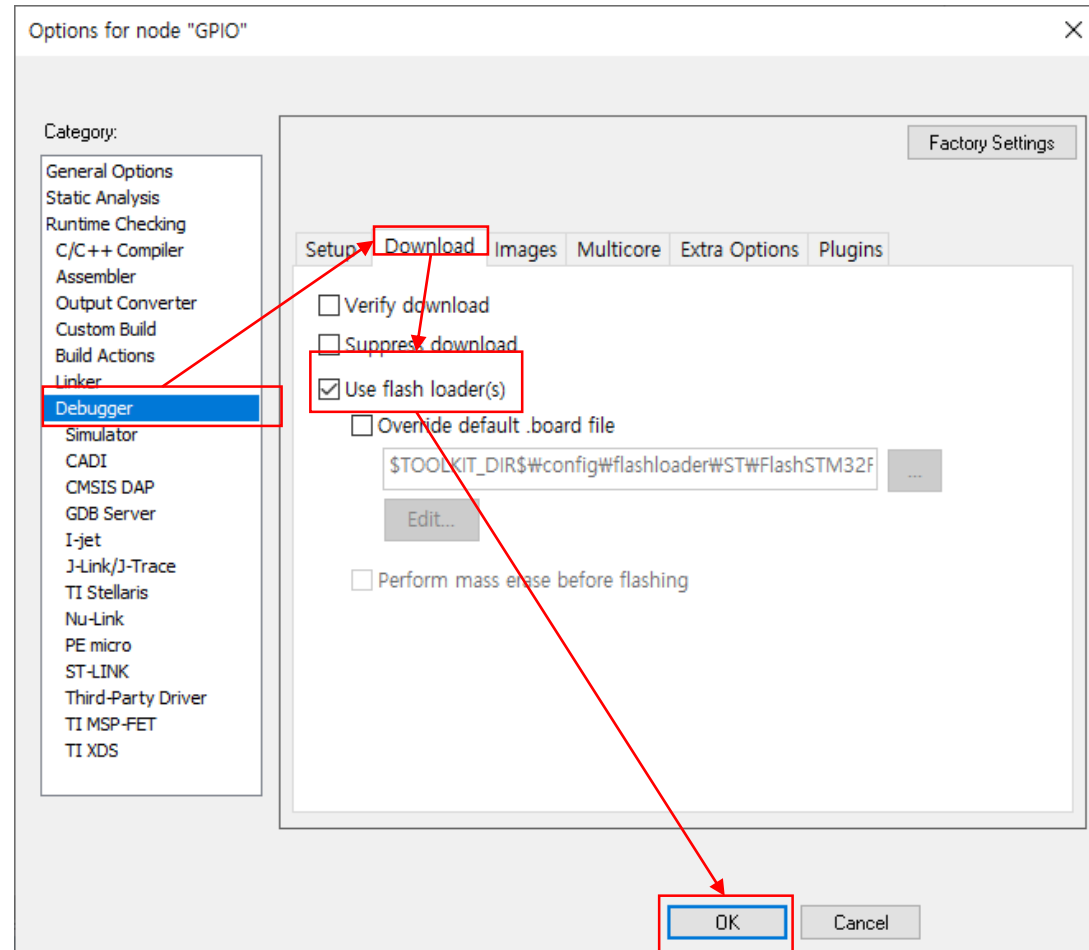
실습환경 구축

- 프로젝트 옵션 설정
 - Category: Debugger
 - Tab: Setup
 - Driver: ST-LINK



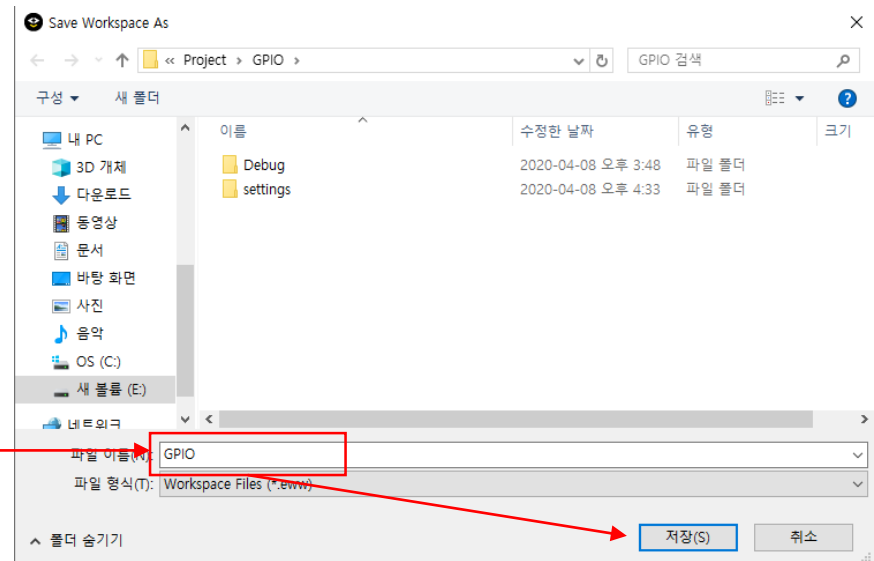
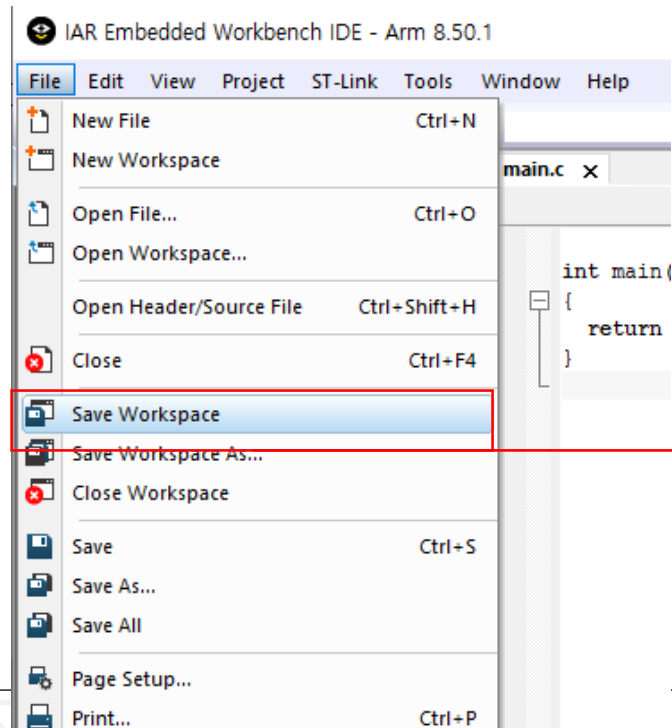
실습환경 구축

- 프로젝트 옵션 설정
 - Category: Debugger
 - Tab: Download
 - Use flash loader(s): Enable
 - 모든 옵션 설정 후 "OK"



실습환경 구축

- Workspace 저장
 - File → Save Workspace
 - workspace명 "GPIO" 설정 및 저장
 - GPIO.eww 파일 생성



- main.c 코드 작성

```
#include "stm32f4xx.h"

void Delay(__IO uint32_t nCount)
{
    for(; nCount != 0; nCount--);
}

int main()
{
    GPIO_InitTypeDef GPIO_InitStructure;

    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);

    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;

    GPIO_Init(GPIOA, &GPIO_InitStructure);

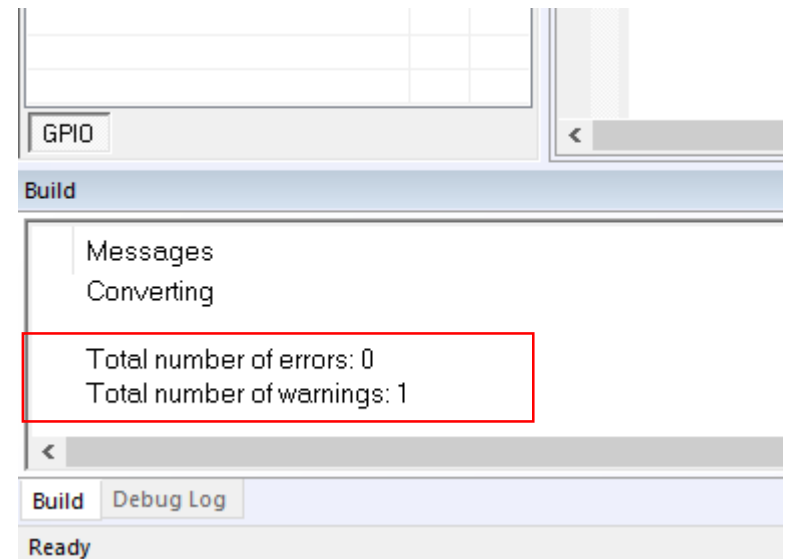
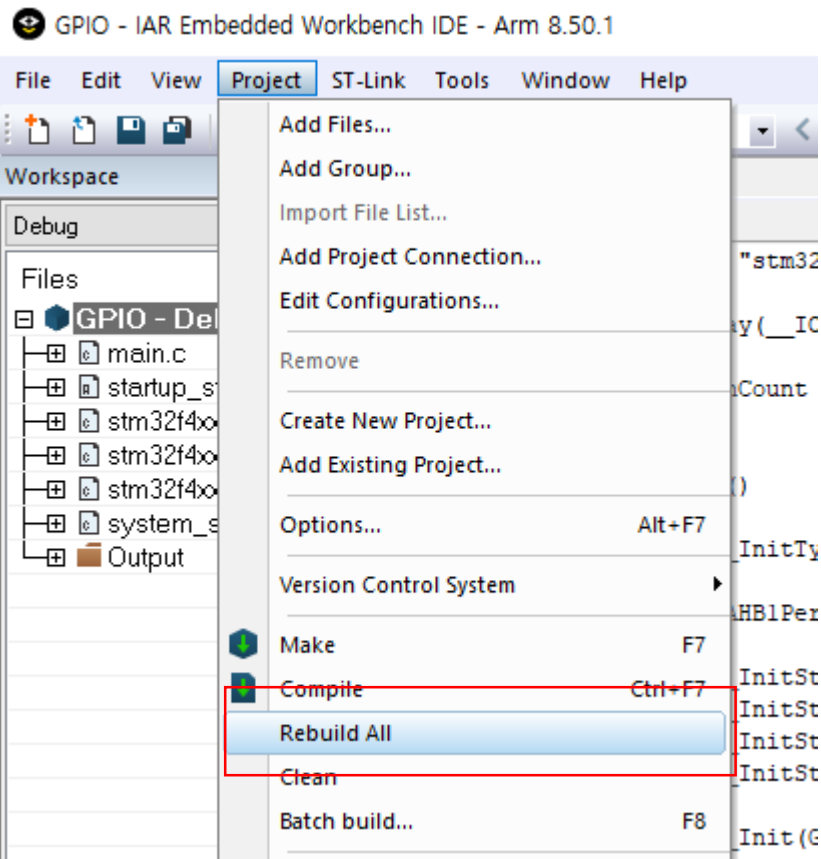
    while(1)
    {
        GPIO_SetBits(GPIOA, GPIO_Pin_5);
        Delay(1000000);
        GPIO_ResetBits(GPIOA, GPIO_Pin_5);
        Delay(1000000);
    }

    return 0;
}
```

실습환경 구축

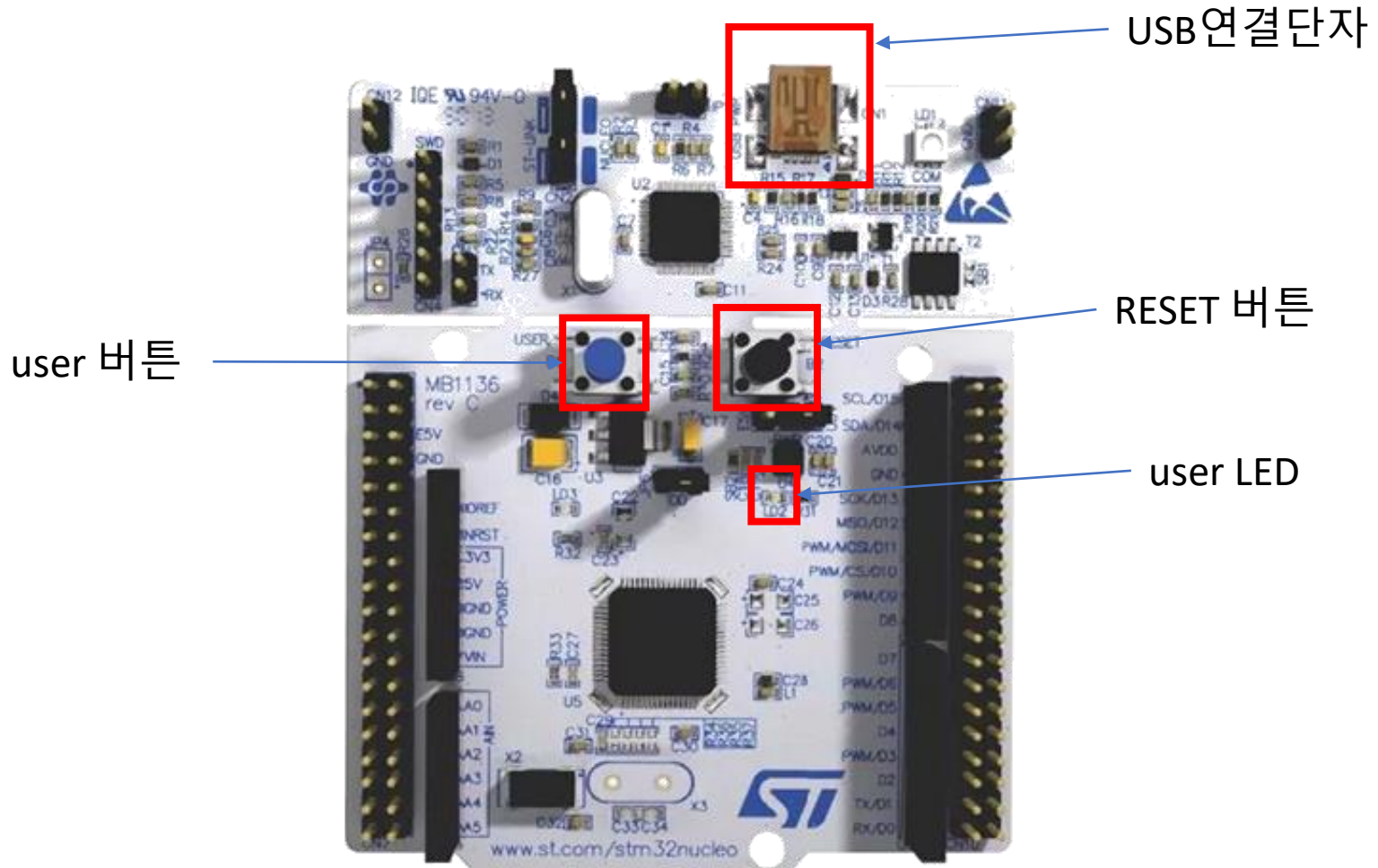
- 프로젝트 build

- Project → Rebuild All

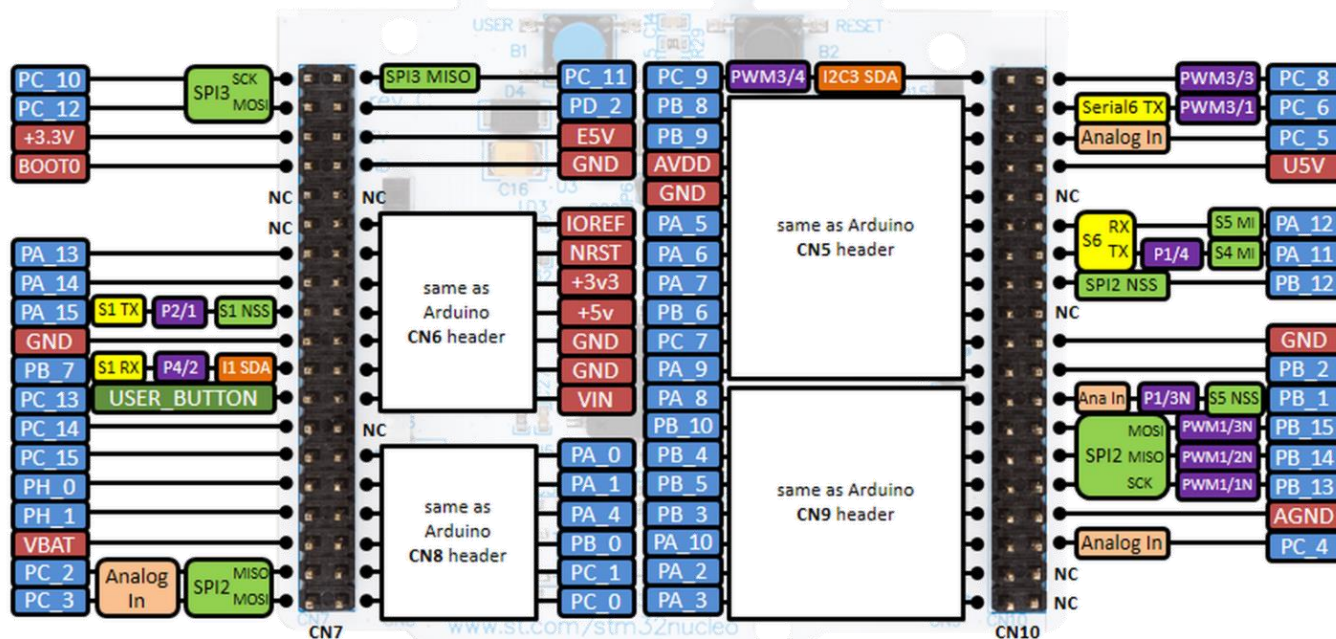


실습: HW 구성

NUCLEO-F411RE board



NUCLEO-F411RE board



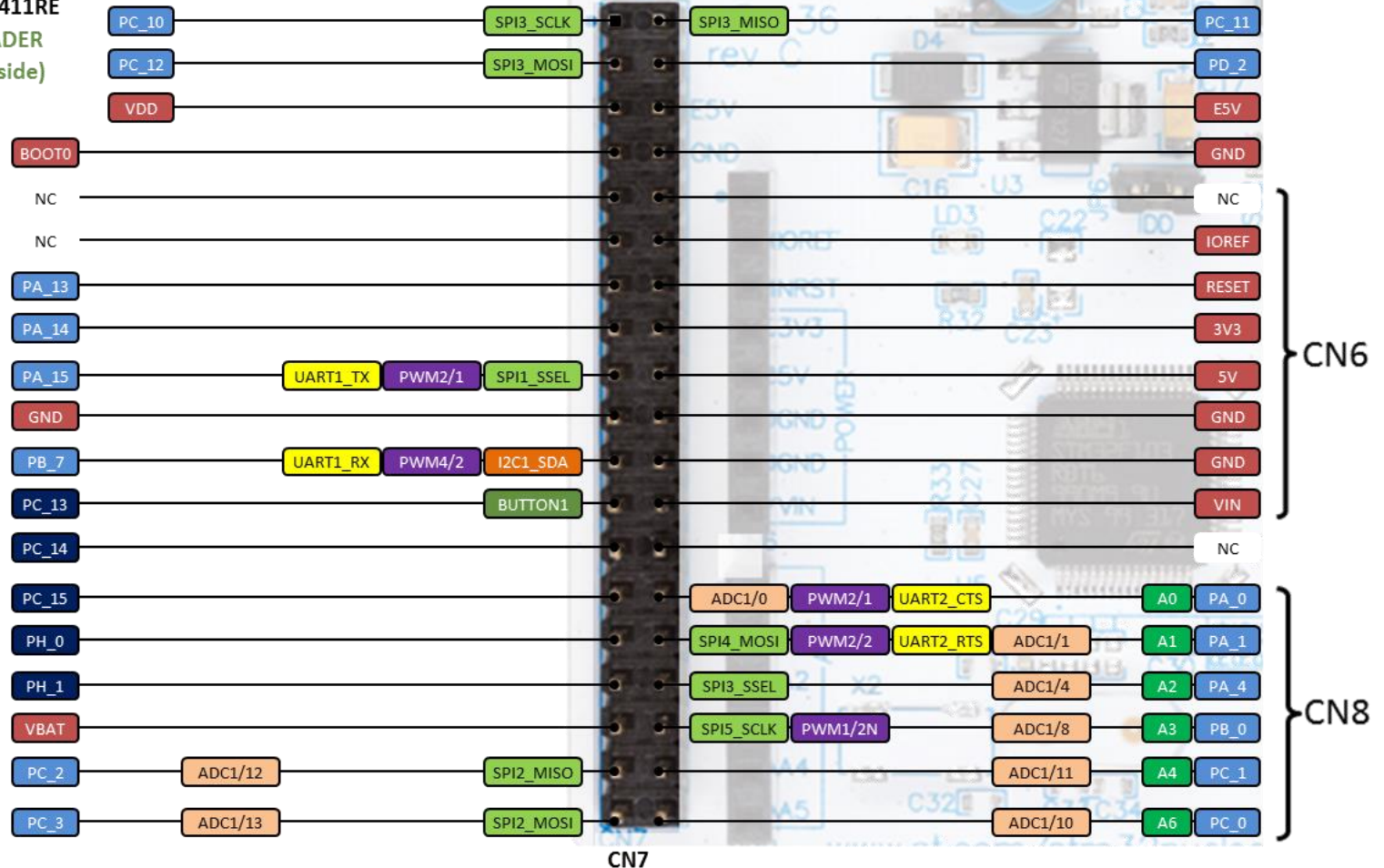
| 실습: HW 구성

NUCLEO-F411RE board

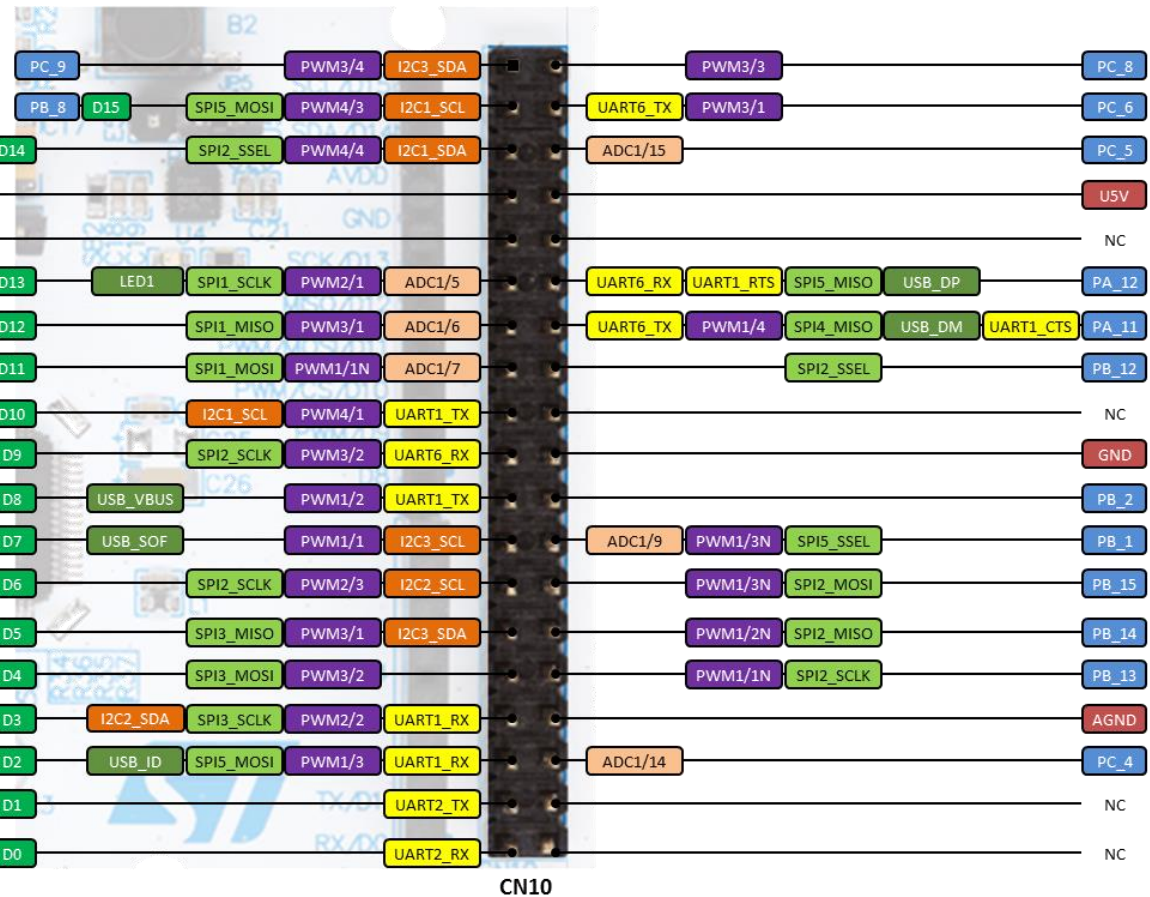
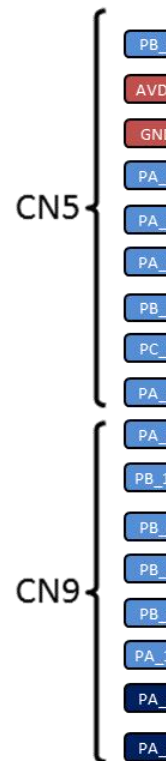
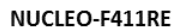


NUCLEO-F411RE

CN7 HEADER
(top left side)



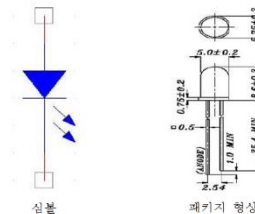
NUCLEO-F411RE board



실습: HW 구성

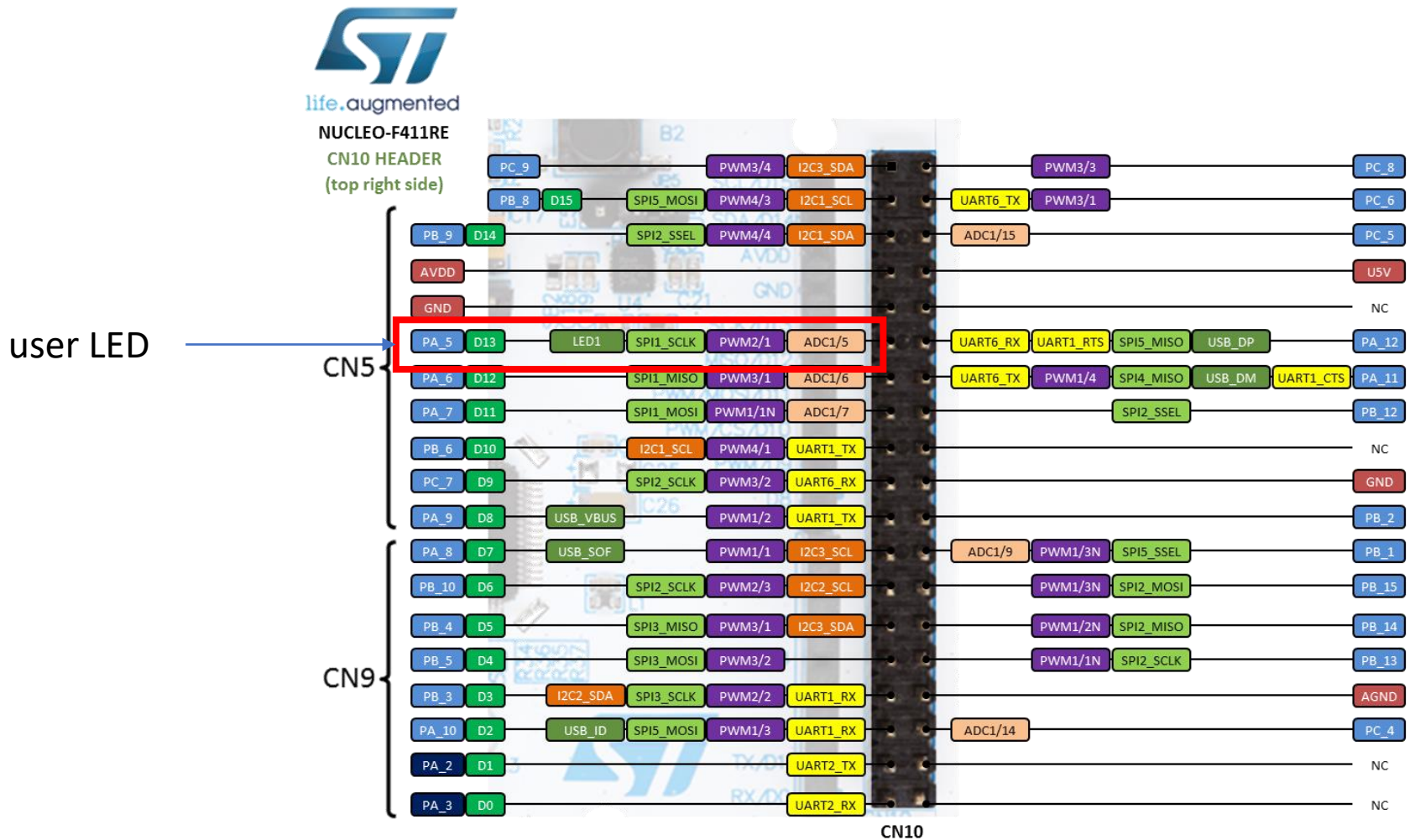
LED

- LED는 Light-emitting diode의 약자로서, 발광다이오드라 부름. 이는 빛을 발산하는 반도체 소자
- 백열전구에 비하여 소비전력은 1/5밖에 안되고, 반응시간은 백만 배나 빠르며, 수명은 반영구적으로 향상되어 정밀 반도체 장비 검사 기기, 자동차 계기판 등의 전자 표시 판, 전광판, 산업기기 표시기 및 각종 교통 안전 신호기 등등에 많이 사용
- LED 는 Pn 접합을 하는 다이오드이고 순방향에 전류를 흘리는 것에 따라 전자와 정공이 재결합하여 발광하는 소자
- 전기적 특성은 일반적으로 10 ~ 20[mA] 의 전류에서 1.5[V] ~ 2.5[V] 의 전압강하를 결정
- LED 의 모양은 다리가 긴 부분이 양극 (Anode), 짧은 쪽이 음극 (Cathode)



실습: HW 구성

User LED



실습: LED 제어 1

User LED

- user LED(PA_5)가 ON-OFF 되는 프로그램을 작성

```
#include "stm32f4xx.h"

void Delay(__IO uint32_t nCount)
{
    for(; nCount != 0; nCount--);
}

int main()
{
    GPIO_InitTypeDef GPIO_InitStructure;

    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);

    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;

    GPIO_Init(GPIOA, &GPIO_InitStructure);

    while(1)
    {
        GPIO_SetBits(GPIOA, GPIO_Pin_5);
        Delay(5000000);
        GPIO_ResetBits(GPIOA, GPIO_Pin_5);
        Delay(5000000);
    }

    return 0;
}
```

I 실습: LED 제어 1

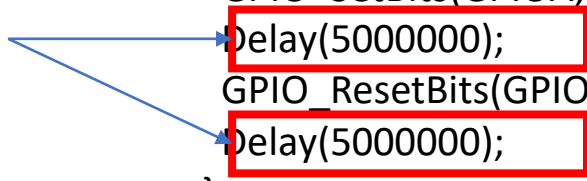
User LED

- Delay함수의 입력 값의 변화에 따른 LED의 변화를 확인

Delay함수

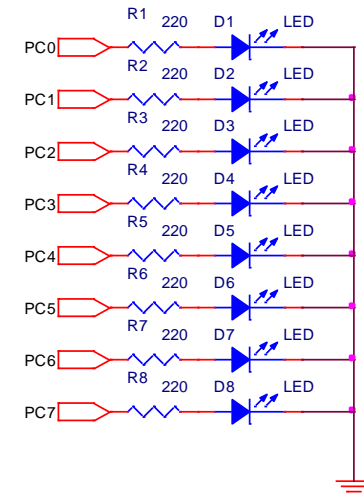
```
while(1)
{
    GPIO_SetBits(GPIOA, GPIO_Pin_5);
    Delay(5000000);
    GPIO_ResetBits(GPIOA, GPIO_Pin_5);
    Delay(5000000);
}

return 0;
}
```



외부 LED

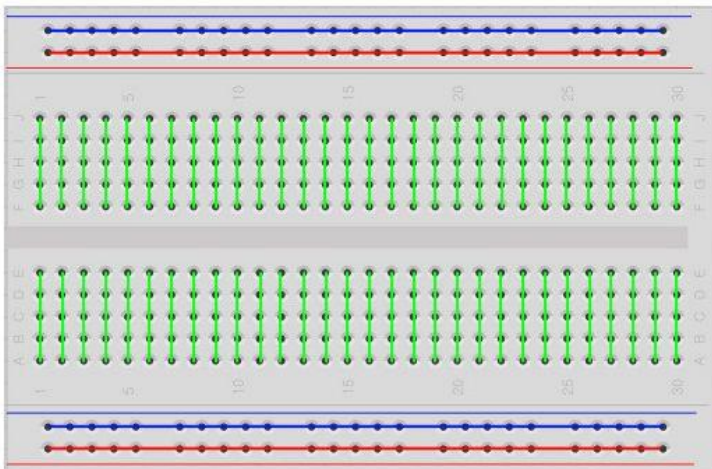
-
- The image displays the ST Nucleo F411RE Morpho Headers. At the top left is the ST logo with the text "life.augmented" and "Nucleo F411RE Morpho Headers". At the top right is the "mbed Enabled" logo. The center features a photograph of the Nucleo F411RE board. Below the photo is a detailed pinout diagram for headers CN7 and CN10. The diagram shows various pins connected to functions like SPI3, I2C3, UART, and analog inputs, with some pins labeled as "same as Arduino" headers.



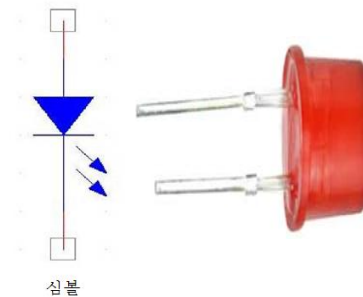
실습: LED 제어 2

외부 LED

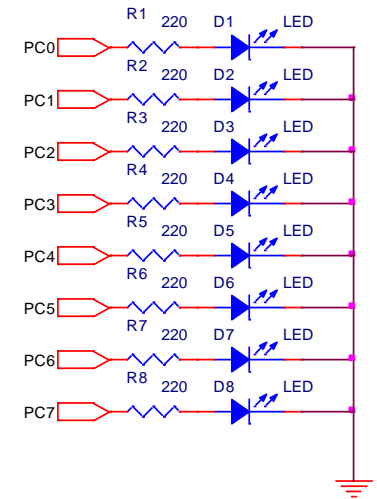
- HW 구성
 - 저항(220Ω), LED를 아래와 같이 연결
 - 포트 C PC0~PC7을 각각 LED와 연결



브레드 보드의 내부 연결도



LED의 심볼과 실물



I 실습: LED 제어 2

외부 LED

- 8개의 LED가 ON-OFF 되는 프로그램 작성

```
#include "stm32f4xx.h"

void Delay(__IO uint32_t nCount)
{
    for(; nCount != 0; nCount--);
}

int main()
{
    GPIO_InitTypeDef GPIO_InitStructure;

    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);

    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;

    GPIO_Init(GPIOC, &GPIO_InitStructure);

    while(1)
    {
        GPIO_Write(GPIOC, 0x0000);
        Delay(5000000);
        GPIO_Write(GPIOC, 0x00FF);
        Delay(5000000);
    }

    return 0;
}
```

I 실습: LED 제어 2

외부 LED

- 8개의 LED가 순차적으로 하나씩 켜지도록 프로그램 수정

```
while(1)
{
    GPIO_Write(GPIOC, GPIO_Pin_0);
    Delay(5000000);
    GPIO_Write(GPIOC, GPIO_Pin_1);
    Delay(5000000);
    GPIO_Write(GPIOC, GPIO_Pin_2);
    Delay(5000000);
    GPIO_Write(GPIOC, GPIO_Pin_3);
    Delay(5000000);
    GPIO_Write(GPIOC, GPIO_Pin_4);
    Delay(5000000);
    GPIO_Write(GPIOC, GPIO_Pin_5);
    Delay(5000000);
    GPIO_Write(GPIOC, GPIO_Pin_6);
    Delay(5000000);
    GPIO_Write(GPIOC, GPIO_Pin_7);
    Delay(5000000);
}
```

실습: LED 제어 2

외부 LED

- 8개의 LED가 순차적으로 하나씩 켜지도록 프로그램 수정
- 무엇이 문제인가?

```
int led = 0;
```

```
while(1)
```

```
{
```

```
    GPIO_Write(GPIOC, led);
```

```
    Delay(500000);
```

```
    led++;
```

```
    if(led > 255)
```

```
        led = 0;
```

```
}
```



0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---

led = 1



0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

led = 2



0	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---

led = 3



⋮

1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---

led = 255

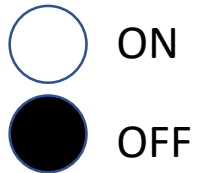
0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

| 실습: LED 제어 3

외부 LED

- 8개의 LED가 아래와 같은 순서로 ON 되도록 프로그램을 수정



Time : 0

LED0

LED7



Time : 1



Time : 2



Time : 3



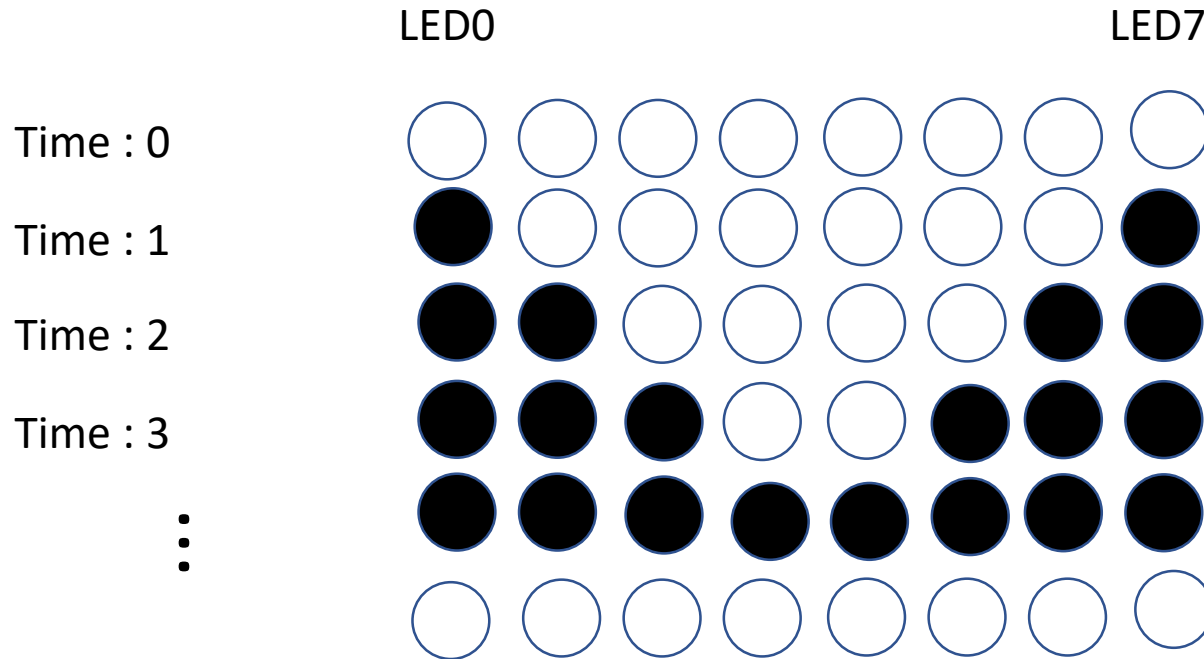
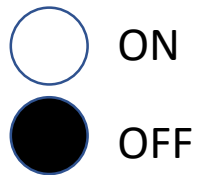
--	--	--	--	--	--	--	--

1	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

과제: LED 제어 4

외부 LED

- 8개의 LED가 아래와 같은 순서로 ON 되도록 프로그램을 수정



Thank you.

