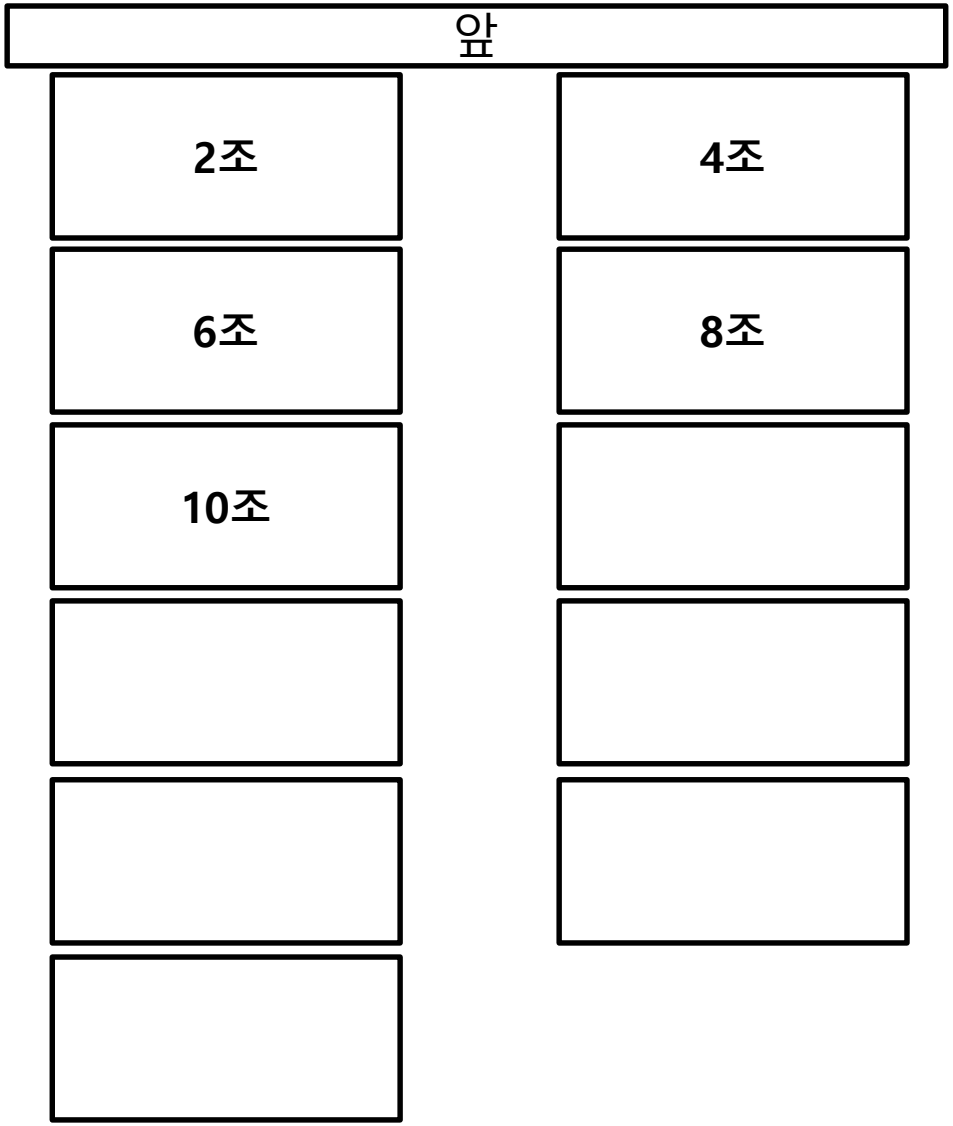
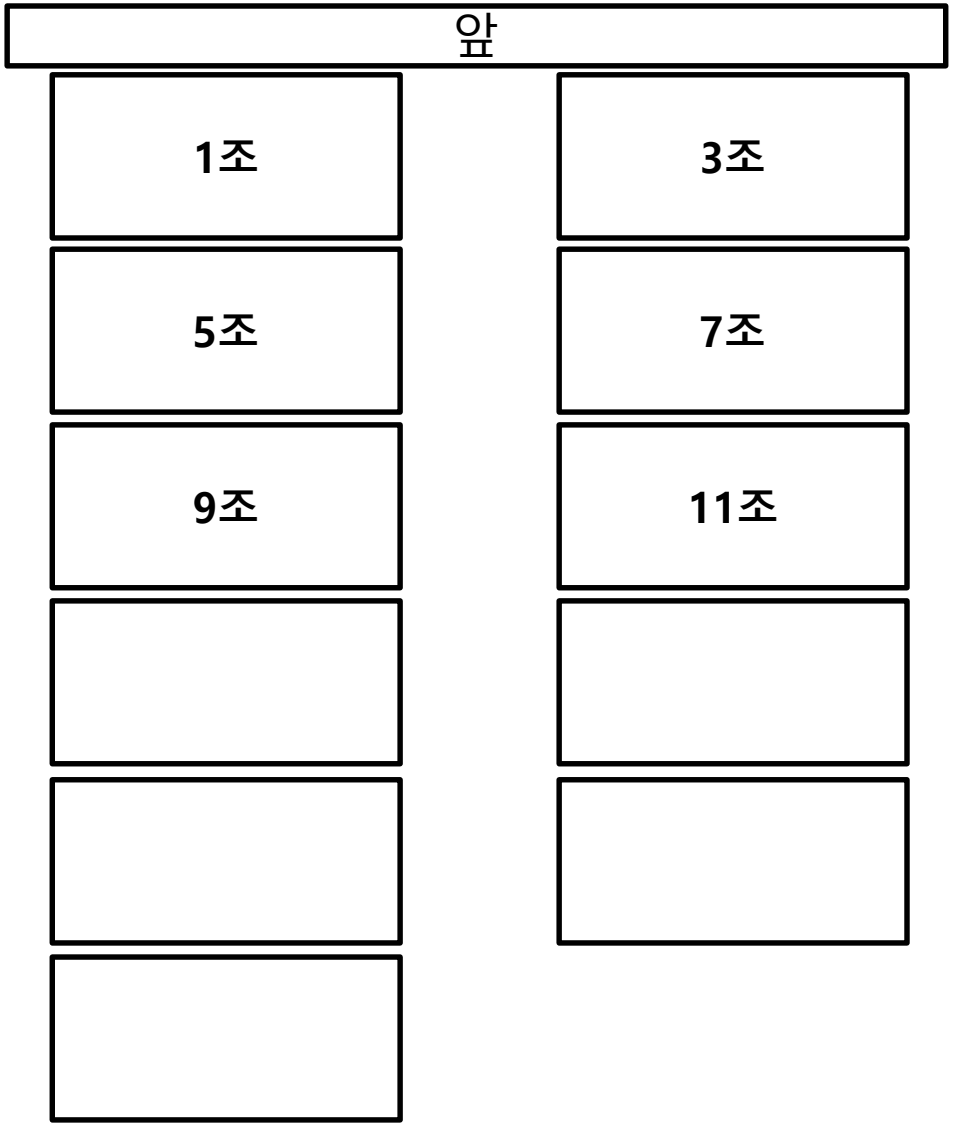
공지



공지



주차별 실험 계획

주차	날짜	실험내용
1	09.02	분반편성 및 Overview
2	09.08	개발 환경 구축 및 개발 장비 교육
3	09.15	디버깅 툴(J-Link) 및 레지스터와 주소 제어를 통한 임베디드 펌웨어 개발
4	09.22	휴강 (추석)
5	09.29	스캐터 로딩 파일 및 플래시 메모리 이해
6	10.06	Polling 방식을 이용한 UART 통신 및 Clock control
7	10.13	Interrupt 방식을 활용한 GPIO 제어 및 UART 통신
8	10.20	휴강 (중간고사 없음)
9	10.27	Bluetooth 및 납땜
10	11.03	TFT-LCD 제어 및 ADC 구현
11	11.10	Timer 및 PWM 구현
12	11.17	DMA 구현
13~	11.24~12.24	텀 프로젝트 진행 (최종 검사일은 미정)
15~16	12.08~12.19	**기말고사 (시험일은 미정)**

차후에 변경 가능

공지

예비 발표 방법 변경

- **발표 영상을 녹화하여 조교 이메일 (chrismail@naver.com) 로 제출**
- **모든 학생은 조교가 PLATO에 업로드한 동영상 각자 시청**

제출 기한

11월 16일
3

Pusan National University

임베디드 시스템 설계 및 실험
수요일 분반

Nov 9, 2020

11주차
Timer & PWM

조교
김준명

INS LAB
IoT NETWORK & SYSTEMS LAB

Contents

11주차 실험 내용

실험 목적

- 타이머란
- 타이머의 종류와 특징
- 분주 계산
- PWM 및 예제

타이머란

- 주기적 시간 처리에 사용하는 디지털 카운터 회로 모듈
- 펄스폭 계측, 주기적인 interrupt 발생 등에 사용
- 주파수가 높기 때문에 우선 prescaler를 사용하여 주파수를 낮춘 후 낮아진 주파수로 8,16비트 등의 카운터 회로를 사용하여 주기를 얻는다.
- STM32 타이머 종류
  - SysTick Timer
  - Watchdog Timer
  - Advanced-control Timer (TIM1, TIM8)
  - General-purpose Timer (TIM2 ~ TIM5)
  - Basic Timer (TIM6, TIM7)

Systick Timer

Real-time operating system 전용이지만 standard down counter로 사용 할 수도 있음

- 24bit down counter
- Autoreload capability
- Counter가 0에 도달하면 설정에 따라 인터럽트가 발생
- Programmable clock source

IWDG/WWDG Timer-Watching timer

WATCHDOG(WDG)은 임베디드 시스템 등 특수 상황에서 CPU가 올바르게 작동하지 않을 시 강제로 리셋시키는 기능을 의미한다

- IWDG/WWDG는 모두 소프트웨어 고장으로 인한 오작동을 감지하고 해결하는 역할을 한다.
- 카운터가 주어진 시간 초과 값에 도달했을 때 시스템 재설정 또는 인터럽트(only WWDG)를 트리거한다.
- IWDG
  - IWDG는 LSI 클럭 기반으로 메인 클럭 고장에도 활성 상태 유지 가능
  - 타이밍 정확도 제약이 낮은 애플리케이션에 적합하다.
  - 카운터가 0이 되면 Reset
- WWDG
  - 7-bit down counter
  - WWDG의 클럭은 APB1 클럭을 프리스케일해서 정의 가능
  - 비정상적 어플리케이션 동작 감지를 위해 설정 가능한 time-window가 있다.
  - Time-window 내에서 반응하도록 요구하는 애플리케이션에 적합하다.
  - 카운터가 0x40 보다 작을 경우 또는 카운터가 Time-window 밖에 Reload 됐을 경우 Reset 가능하다.
  - Early wakeup interrupt (EWI) : 카운터가 0x40과 같을 때, EWI 인터럽트 발생하게 설정 가능하다.

Advanced-control timers (TIM1 & TIM8)

- The advanced-control timers는 prescaler를 이용해 설정 가능한 16-bit auto-reload counter를 포함하고있다.
- 입력 신호 펄스 길이 측정(input capture) 또는 출력 파형 생성(output compare, PWM, complementary PWM with dead-time insertion) 등에 사용 가능하다.
- The advanced-control (TIM1&TIM8) and general-purpose (TIMx)는 자원을 공유하지 않는 독립적인 구조이며, 동기화 시키는 것도 가능하다

Basic timer (TIM6 & TIM7)

- 16-bit auto-reload 업카운터
- 설정 가능한 16-bit prescaler를 이용해 the counter clock 주파수를 나눠서 설정 가능
- DAC 트리거에 사용
- 카운터 오버플로우 발생 시 인터럽트/DMA 생성

General-purpose timers (TIM2 to TIM5)

- General-purpose timer은 prescaler를 이용해 설정 가능한 16-bit up, down, up/down auto-reload counter를 포함하고있다.
- 입력 신호의 펄스 길이 측정(input capture) 또는 출력 파형 발생(output compare and PWM) 등 다양한 용도로 사용할 수 있다.
- 펄스 길이와 파형 주기는 timer prescaler와 the RCC clock controller prescaler를 사용하여 몇 μs에서 몇 ms까지 변조할 수 있다. 타이머들은 완전히 독립적이며, 어떤 자원도 공유하지 않으나 동기화 가능하다.

분주 계산

General-purpose timers (TIM2 to TIM5)

$$\frac{1}{f_{clk}} \times prescaler \times period$$

$$\frac{1}{72Mhz} \times 7200 \times 10000 = 1[s]$$

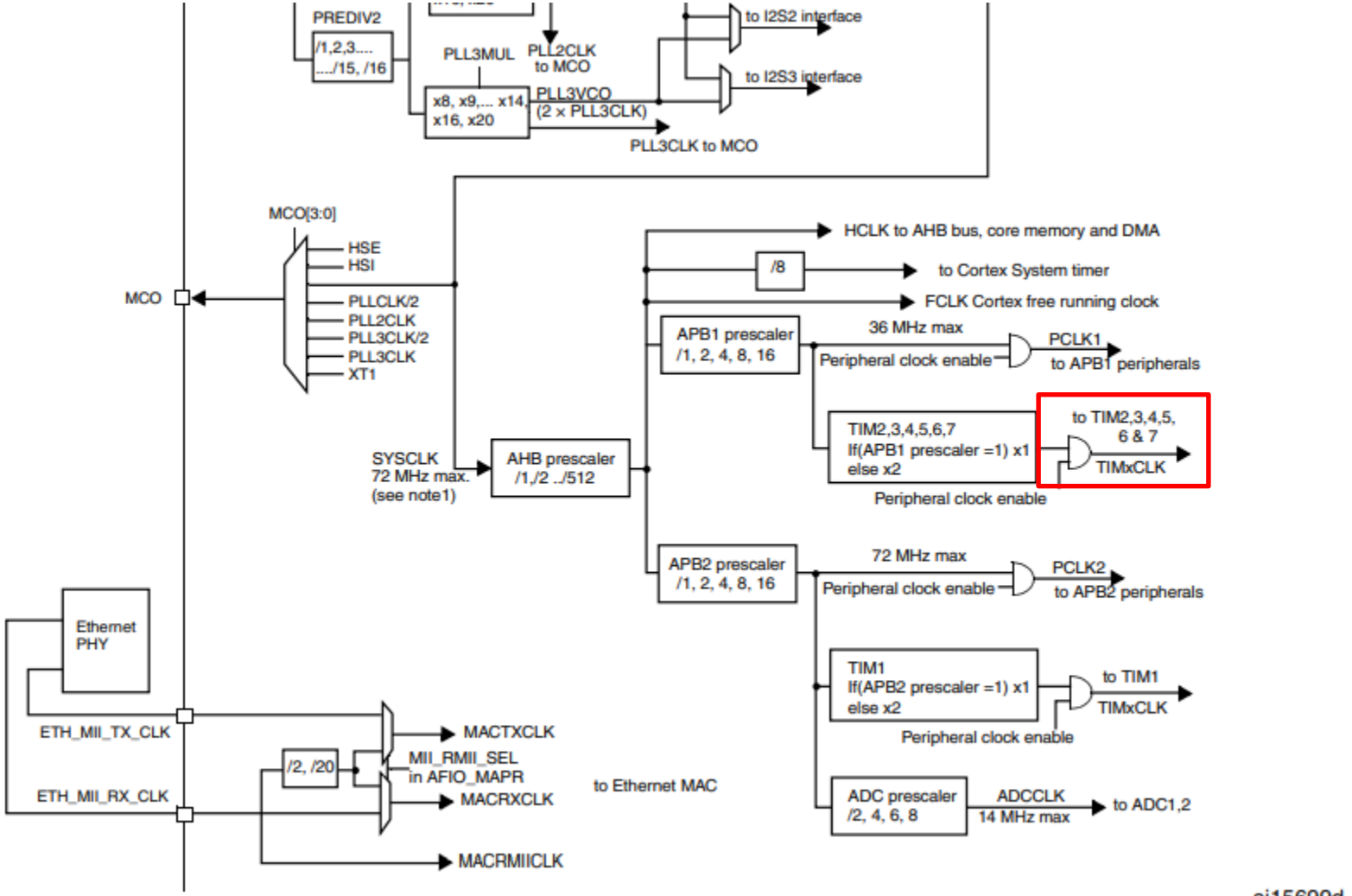
- 분주란 MCU에서 제공하는 Frequency를 우리가 사용하기 쉬운 값으로 바꾸어 주는 것을 말합니다.
- Counter clock frequency 를 1~65536의 값으로 나누기 위해 16-bit programmable prescaler 사용
- period 로 몇 번 count하는지 설정

$$f_{clk} * \frac{1}{prescaler} * \frac{1}{period} = \text{주파수}[Hz]$$

#현재 버그 있습니다

Clock frequency

라이브러리에서 설정된 timer clock frequency 확인하기



Clock frequency

라이브러리에서 설정된 timer clock frequency 확인하기

main

```
SystemInit();
RccInit();
GpioInit();
TIM_Configure();
NvicInit();

LCD_Init();
Touch_Configuration();
Touch_Adjust();
```

```
/* Configure the System clock frequency, HCLK, PCLK2 and PCLK1 prescalers */
/* Configure the Flash Latency cycles and enable prefetch buffer */
SetSysClock();
```

```
static void SetSysClock(void)
{
#ifdef SYSCLK_FREQ_HSE
  SetSysClockToHSE();
#elif defined SYSCLK_FREQ_24MHz
  SetSysClockTo24();
#elif defined SYSCLK_FREQ_36MHz
  SetSysClockTo36();
#elif defined SYSCLK_FREQ_48MHz
  SetSysClockTo48();
#elif defined SYSCLK_FREQ_56MHz
  SetSysClockTo56();
#elif defined SYSCLK_FREQ_72MHz
  SetSysClockTo72();
#endif
```

에서 확인

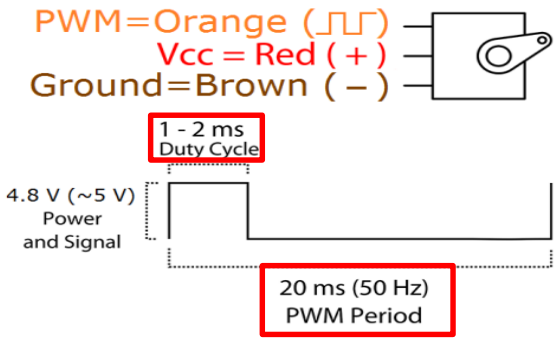
```
#if defined (STM32F10X_LD_VL) || (defined STM3
/* #define SYSCLK_FREQ_HSE    HSE_VALUE */
 #define SYSCLK_FREQ_24MHz  24000000
#else
/* #define SYSCLK_FREQ_HSE    HSE_VALUE */
/* #define SYSCLK_FREQ_24MHz  24000000 */
/* #define SYSCLK_FREQ_36MHz  36000000 */
/* #define SYSCLK_FREQ_48MHz  48000000 */
/* #define SYSCLK_FREQ_56MHz  56000000 */
#define SYSCLK_FREQ_72MHz  72000000
#endif
```

PWM(Pulse Width Modulation)

일정한 주기 내에서 Duty ratio를 변화 시켜서 평균 전압을 제어하는 방법
ex) 0~5V의 전력 범위에서 2.5V 전압을 가하고 싶다면, 50% 듀티 사이클 적용

대부분의 서보모터는 50Hz ~ 1000Hz 의 주파수를 요구하고
데이터 시트를 반드시 확인해서 사용해야함.

SG90 서보모터



Position "0" (1.5 ms pulse) is middle, "90" (~2 ms pulse) is all the way to the right, "-90" (~1 ms pulse) is all the way to the left.

STM32보드 PWM 신호 예시

TIM3의 채널 3 사용 예시
- PB0를 사용해야하며, Alternate function사용 (다른 채널 사용시 다른 핀을 사용해야함.)

```
prescale = (uint16_t) (SystemCoreClock / TODO);

TIM_TimeBaseStructure.TIM_Period = TODO;
TIM_TimeBaseStructure.TIM_Prescaler = prescale;
TIM_TimeBaseStructure.TIM_ClockDivision = 0;
TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Down;

TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_Pulse = TODO; // us
TIM_OC3Init(TIM3, &TIM_OCInitStructure);

TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
TIM_OC3PreloadConfig(TIM3, TIM_OCPreload_Disable);
TIM_ARRPreloadConfig(TIM3, ENABLE);
TIM_Cmd(TIM3, ENABLE);
```

Pins			Pin name	Type(1)	I / O Level(2)	Main function(3) (after reset)	Alternate functions(4)	
BGA100	LQFP64	LQFP100					Default	Remap
H2	15	24	PA1	I/O	-	PA1	USART2_RTS(7)/ ADC12_IN1/ TIM5_CH2 /TIM2_CH2(7)/ ETH_MII_RX_CLK/ ETH_RMII_REF_CLK	-
J2	16	25	PA2	I/O	-	PA2	USART2_TX(7)/ TIM5_CH3/ADC12_IN2/ TIM2_CH3 (7)/ ETH_MII_MDIO/ ETH_RMII_MDIO	-
K2	17	26	PA3	I/O	-	PA3	USART2_RX(7)/ TIM5_CH4/ADC12_IN3 / TIM2_CH4(7)/ ETH_MII_COL	-
E4	18	27	V_{SS_4}	S	-	V_{SS_4}	-	-
F4	19	28	V_{DD_4}	S	-	V_{DD_4}	-	-
G3	20	29	PA4	I/O	-	PA4	SPI1_NSS(7)/DAC_OUT1 / USART2_CK(7) / ADC12_IN4	SPI3_NSS/I2S3_WS
H3	21	30	PA5	I/O	-	PA5	SPI1_SCK(7) / DAC_OUT2 / ADC12_IN5	-
J3	22	31	PA6	I/O	-	PA6	SPI1_MISO(7)/ADC12_IN6 / TIM3_CH1(7)	TIM1_BKIN
K3	23	32	PA7	I/O	-	PA7	SPI1_MOSI(7)/ADC12_IN7 / TIM3_CH2(7)/ ETH_MII_RX_DV(8)/ ETH_RMII_CRS_DV	TIM1_CH1N
G4	24	33	PC4	I/O	-	PC4	ADC12_IN14/ ETH_MII_RXD0(8)/ ETH_RMII_RXD0	-
H4	25	34	PC5	I/O	-	PC5	ADC12_IN15/ ETH_MII_RXD1(8)/ ETH_RMII_RXD1	-
J4	26	35	PB0	I/O	-	PB0	ADC12_IN8/TIM3_CH3/ ETH_MII_RXD2(8)	TIM1_CH2N
K4	27	36	PB1	I/O	-	PB1	ADC12_IN9/TIM3_CH4(7)/ ETH_MII_RXD3(8)	TIM1_CH3N
G5	28	37	PB2	I/O	FT	PB2/BOOT1	-	-
H5	-	38	PE7	I/O	FT	PE7	-	TIM1_ETR
J5	-	39	PE8	I/O	FT	PE8	-	TIM1_CH1N

Datasheet Table 5. Pin definitions

#현재 의도치않게 SG50용 50Hz생성시 prescale=72, Period=20000로 강제됨 원인파악중

실험 주의사항

- 실험 장비들을 연결 및 분리할 때 반드시 모든 전원을 끄고 연결해주세요.
- 장비사용시 충격이 가해지지 않도록 주의해주세요.
- 자리는 항상 깔끔하게 유지하고 반드시 정리 후 퇴실해주세요.
- 실험 소스 코드와 프로젝트 폴더는 백업 후 반드시 삭제해주세요.
- 장비 관리, 뒷정리가 제대로 되지 않을 경우 해당 조에게 감점이 주어집니다.

- 동작 중 케이블 절대 뽑지말것
- 보드는 전원으로 USBPort나 어댑터(5V,1A)를 사용할것 (5V 5A 어댑터(비슷하게 생김)와 혼동하지 말 것, 사용시 보드가 타버림 -> 감점)
- 디버깅 모드 중에 보드 전원을 끄거나 연결 케이블을 분리하지 말 것!!!

- ->지켜지지 않을 시 해당 조 감점

미션 ! 별도 미션지 참고

실험 검사

오늘은 2시간 내에 다 하고 검사 받을 수 있기를 바랍니다

이번 주 실험 결과 보고서 및 소스 코드 및 실험 동작 영상

A. **이론부터 실습까지 전반적인 내용을 포함하도록 작성 (실험 과정 사진 찍으시면 좋아요)**
B. **다음 실험전날 자정전까지 E-mail 제출**
C. **소스 코드는 직접 작성 및 수정한 파일만 제출**

예비 발표 조는 발표 자료(영상) 만들어서
일요일 24시까지 조교 이메일로 제출

나가실 때, 만드신 코드 및 프로젝트 폴더는 모두 백업하시고 삭제해주세요.
다른 분반 파일은 만지지 마시고 조교에게 알려주세요.
자리 정리정돈 안 되어 있으면 감점합니다!!!