**REPORT**

**| Embedded System**

**| Programming on an embedded systems 1**

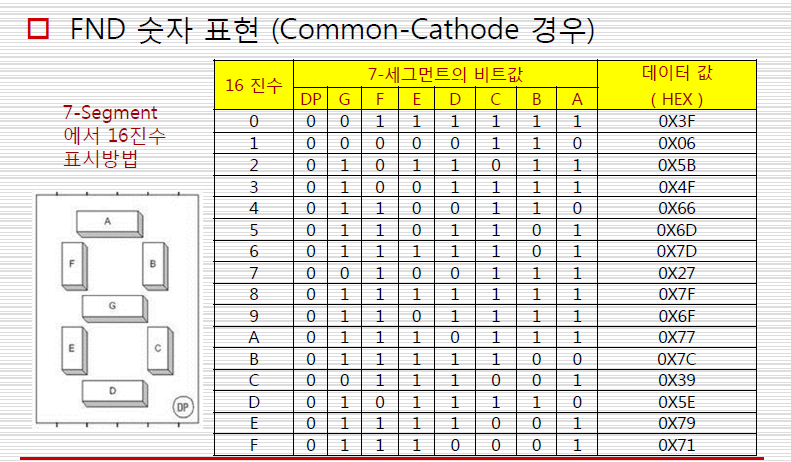
**| 컴퓨터정보공학과**

**| 12141540 박영창**

# 내용

9주차 실습에서는 임베디드 보드의 개발 환경 설정 및 LED/FND의 사용에 대해 배웠습니다.

* 개관
  + 임베디드 보드 : 기본적으로 교차 개발 환경이다 ( 보드에서 실행하는 환경과 컴퓨터로 개발하는 환경이 다르다 )
  + Atmega 128
    - RISC 구조 / 16Mhz ( 16 MIPS 성능 ) / 133종 명령어 세트 / 32개의 8bit 범용 레지스터 등의 특성을 가지고 있다.
    - 프로그램 메모리 : 프로그램 코드를 저장하고 실행하기 위해 필요한 메모리
    - 데이터 메모리 : 프로그램이 실행될 때 임시로 데이터를 저장하는 메모리
  + GPIO ( General Purpose Input Output ) : 범용 입출력 포트로 설계자가 입출력을 마음대로 선택
    - 출력으로는 0 / 1 의 신호, 입력으로는 인터럽트를 처리하는 기능/구조를 가짐
    - GPIO 레지스터 : 입출력 방향 전환 / 출력용 / 입력용 의 세 가지 레지스터가 필요
  + Atmega 128 GPIO 관련 레지스터
    - DDRx ( Data Direction Register ) : 각 포트에 대한 입출력 방향 지정용 레지스터
    - PORTx ( Port Output Register ) : 데이터 출력 레지스터
    - PINx ( Port Input Register ) : 데이터 입력 레지스터
    - SFIOR ( Special Function IO Register ) : 풀업 저항 활성화/비활성화
  + 시간 지연 함수
    - AVR 개발 환경에서 <util/delay.h>라는 헤더 내에 정의되어 있음
    - \_delay\_ms(unsigned int I )
    - \_delay\_us(unsigned int I )
    - 비교적 정확한 시간 지연을 얻을 수 있다.
  + Volatile
    - Volatile로 선언된 변수/포인터/코드를 컴파일러가 임의로 최적화하지 않는다.
    - 사용하는 경우
      * 인터럽트 루틴과 메인 루틴에서 같은 변수를 사용하는 경우
      * 자동변수의 경우 일정 시간 CPU에서 operation을 수행해 주기를 원하는 경우
      * 코드의 순서를 지키고 싶은 경우
      * I/O 레지스터 등을 메모리 매핑으로 사용하는 경우
* LED
  + 작동 원리 : LED 신호 bit를 set해서 LED를 켠다. Bit clear 시 LED를 끈다.
  + 출력 방법 ( ATmega 128 GPIO PA 포트 )
    - DDRA 레지스터의 해당 비트에 1을 write하여 방향을 출력 상태로 만듬
    - PORTA 레지스터의 해당 비트에 1을 write하면 LED on
* FND ( Flexible Numeric Device )
  + LED 7개와 점 1개를 숫자를 표시하기 쉽도록 배열



* + 출력 방법
    - DDRC / DDRG 레지스터의 해당 비트에 1을 write하여 방향을 출력 상태로 만듬
    - PORTC / PORTG 레지스터에 각각 FND 데이터 신호 / 선택 신호를 write해 출력

# 과제 목표

1. < LED 실습 과제 >

좌우로 LED를 번갈아 켜는 프로그램 작성

1. < FND 실습 과제 >

1/100초 단위 초시계 만들기

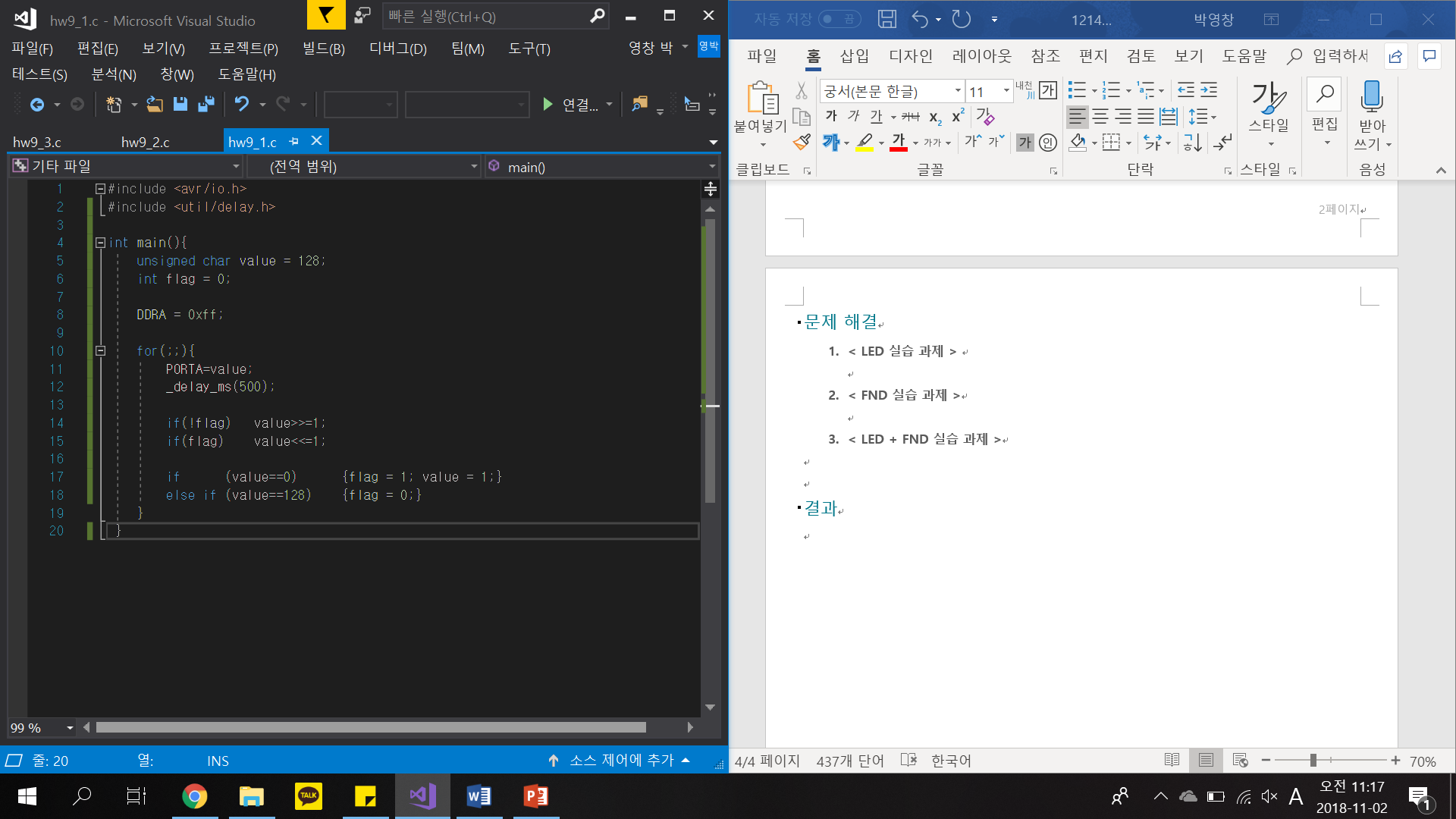
1. < LED + FND 실습 과제 >

0 ~ 7인 랜덤 값 생성해 해당 LED를 켠다

켜진 LED 위치에 매핑되는 숫자를 FND로 출력

# 문제 해결

1. **< LED 실습 과제 >**

**value : 출력할 LED 신호 bit**

**flag : LED 신호의 진행 방향**

**DDRA - 모든 LED를 출력 설정**

**PORTA - 출력할 LED 신호**

**flag 0 : LED 오른쪽 방향으로 이동**

**flag 1 : LED 왼쪽 방향으로 이동**

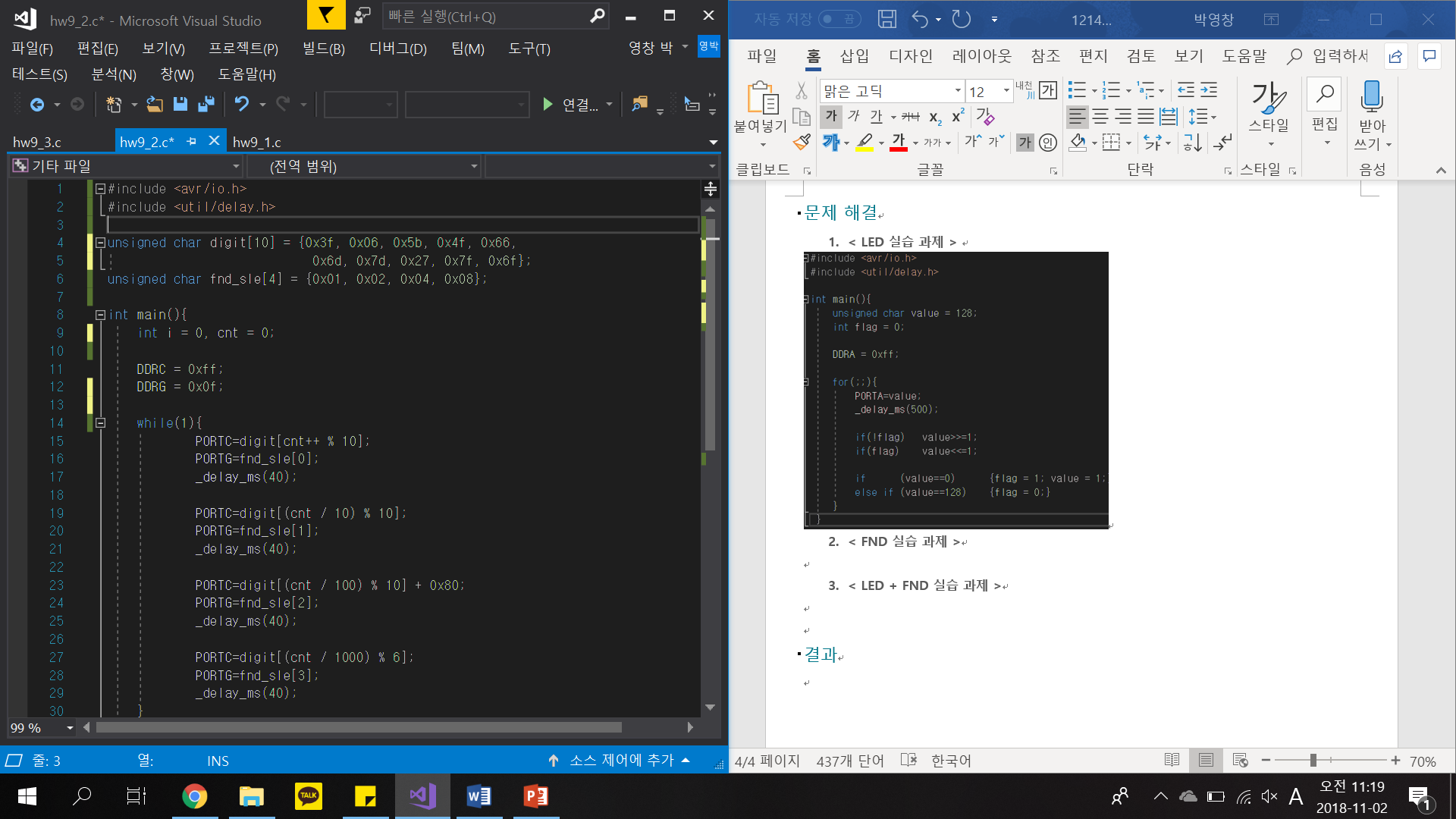
**오른쪽 끝으로 간 경우**

**flag 1로 set 후 value에 1 넣어 줌**

**왼쪽 끝으로 간 경우**

**flag 0으로 set**

1. **< FND 실습 과제 >**

**digit : FND에 보여질 숫자 bit 조합**

**fnd\_sle : FND 출력 위치**

**cnt : 1/100초 단위로 증가할 수**

**DDRC – 모든 FND 데이터 신호 출력 설정**

**DDRG – 모든 FND 선택 신호 출력 설정**

**PORTC – 출력할 FND 데이터 신호**

**PORTG – 출력할 FND 선택 신호**

**0번째 FND - cnt의 1의 자릿수 출력**

**1번째 FND - cnt의 10의 자릿수 출력**

**2번째 FND - cnt의 100의 자릿수 출력**

**3번째 FND - cnt의 1000의 자릿수 출력**

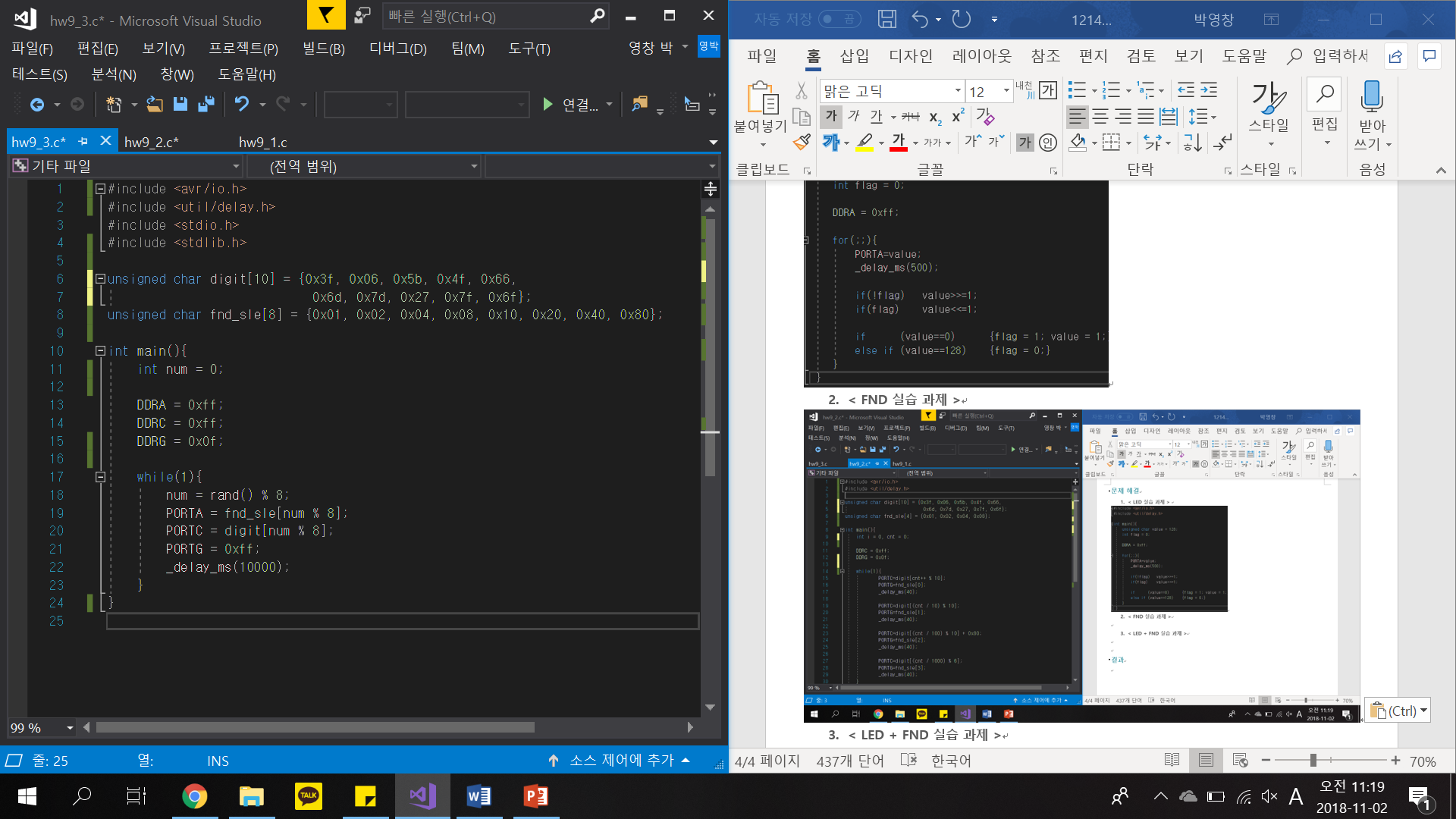
**5까지만 출력 후 0 출력**

**-> 60초를 넘어가지 않게**

**( cnt값은 초기화하지 않음 )**

* + - * **\_delay\_ms(40)으로 설정했을 때 실제 스톱워치와 가장 비슷했습니다**

1. **< LED + FND 실습 과제 >**



**digit : FND에 보여질 숫자 bit 조합**

**fnd\_sle : LED 출력 위치**

**num : rand()를 통해 생성할 난수 저장**

**DDRA – 모든 LED를 출력 설정**

**DDRC – 모든 FND 데이터 신호 출력 설정**

**DDRG – 모든 FND 선택 신호 출력 설정**

**PORTA - 출력할 LED 신호 num에 생성된 난수에 해당하는 LED 출력**

**PORTC – 출력할 FND 데이터 신호 num에 생성된 난수에 해당하는 숫자 출력**

**PORTG – 출력할 FND 선택 신호 0xff으로 전체 FND 선택**

# 결과

**동영상 첨부했습니다**