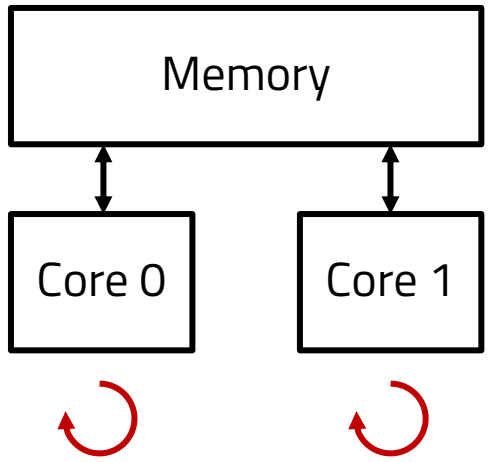
임베디드 기반 SW 개발 프로젝트

AURIX TC275 보드 멀티 코어 사용

현대자동차 입문교육
박대진 교수

실습 목표

- **TC275 보드 MCU에 내장된 2개의 core를 동시에 사용해본다. 여러 개의 core가 공유하는 변수에 접근하여 서로 다른 core가 독립적으로 프로그램을 수행하는 것을 확인해본다.**
  1. **Core 0 에서는 일정한 시간 간격마다 LED Blue 점등하되, 전역 변수의 값이 1일 경우에만 실행**
  2. **Core 1 에서는 Button이 눌렸을 때 전역 변수의 값을 1에서 0으로, 또는 0에서 1로 toggle**



독립적으로 core 에
할당된 프로그램 수행

Core 0 SW 프로그래밍 :LED, CCU60 함수

- Core 0 에서 수행할 동작
  - LED 초기화 (Blue general output 설정)
  - CCU60 타이머 및 인터럽트 초기화

```
14
15  void initLED(void)
16  {
17      P10_IOCR0.U &= ~(0x1F << PC1_BIT_LSB_IDX);      // reset P10_IOCR0 PC1
18      P10_IOCR0.U &= ~(0x1F << PC2_BIT_LSB_IDX);      // reset P10_IOCR0 PC2
19
20      P10_IOCR0.U |= 0x10 << PC1_BIT_LSB_IDX;         // set P10.1 push-pull general output
21      P10_IOCR0.U |= 0x10 << PC2_BIT_LSB_IDX;         // set P10.2 push-pull general output
22  }
23
```

0x10 write

```
124
125 void initCCU60(void)
126 {
127     // Password Access to unlock SCU_WDTS
128     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CO
129     while((SCU_WDTCPU0_CON0.U & (1 << LCK
130 //
131     // Modify Access to clear ENDINIT
```

initCCU60() 함수는 동일

Core 0 SW 프로그래밍
:CCU60 ISR 함수

- 일정한 시간 간격으로 발생하는 CCU60 인터럽트
  - 전역 변수 `glb_flag` 의 값에 따라 LED Blue 상태 toggle

Core 0 에서 처리

```
82
83  __interrupt(0x0B) __vector_table(0)
84  void CCU60_T12_ISR(void)
85  {
86      if( glb_flag == 1 )
87          P10_OUT.U ^= 0x1 << P2_BIT_LSB_IDX;  // toggle P10.2 (LED D13 BLUE)
88  }
89
```

glb_flag 의 값이 1일때만 LED Blue 점멸
그렇지 않다면 동일한 상태 유지

Core 0 SW 프로그래밍
:main 함수

- Core 0 에서 수행할 동작
  - LED 초기화 (Red, Blue general output 설정)
  - CCU60 타이머 및 인터럽트 초기화

```
88
89  unsigned int glb_flag = 0;
90
91
92  int core0_main(void)
93  {
94      IfxCpu_enableInterrupts();
95
96      /* !!WATCHDOG0 AND SAFETY WATCHDOG ARE DISABLED HERE!!
97       * Enable the watchdogs and service them periodically if it is required
98       */
99      IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());
100     IfxScuWdt_disableSafetyWatchdog(IfxScuWdt_getSafetyWatchdogPassword());
101
102     /* Wait for CPU sync event */
103     IfxCpu_emitEvent(&g_cpuSyncEvent);
104     IfxCpu_waitEvent(&g_cpuSyncEvent, 1);
105
106     initCCU60();
107     initLED();
108
109     while(1)
110     {
111
112     }
113     return (1);
114 }
```

→ 여러 개의 Core(0, 1)가 공유할 전역 변수

→ 초기화 함수 호출

사용하지 않는
코드는 주석처리!!

Core 0 SW 프로그래밍 :전체 코드 check

```
76
77  void initLED(void);
78  void initCCU60(void);
79
80
81  __interrupt(0x0B) __vector_table(0)
82  void CCU60_T12_ISR(void)
83  {
84      if( glb_flag == 1 )
85          P10_OUT.U ^= 0x1 << P2_BIT_LSB_IDX;  // toggle P10.2 (LED D13 BLUE)
86  }
87
88
89  unsigned int glb_flag = 0;
90
91
92  int core0_main(void)
93  {
94      IfxCpu_enableInterrupts();
95
96      /* !!WATCHDOG0 AND SAFETY WATCHDOG ARE DISABLED HERE!!
97       * Enable the watchdogs and service them periodically if it is required
98       */
99      IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());
100     IfxScuWdt_disableSafetyWatchdog(IfxScuWdt_getSafetyWatchdogPassword());
101
102     /* Wait for CPU sync event */
103     IfxCpu_emitEvent(&g_cpuSyncEvent);
104     IfxCpu_waitEvent(&g_cpuSyncEvent, 1);
105
106     initCCU60();
107     initLED();
108
109     while(1)
110     {
111
112     }
113     return (1);
114 }
115
```

```
6  void initLED(void)
7  {
8      P10_IOCR0.U &= ~(0x1F << PC1_BIT_LSB_IDX);      // reset P10_IOCR0 PC1
9      P10_IOCR0.U &= ~(0x1F << PC2_BIT_LSB_IDX);      // reset P10_IOCR0 PC2
0
1      P10_IOCR0.U |= 0x10 << PC1_BIT_LSB_IDX;         // set P10.1 push-pull general output
2      P10_IOCR0.U |= 0x10 << PC2_BIT_LSB_IDX;         // set P10.2 push-pull general output
3  }
4
```

```
void initCCU60(void)
{
    // Password Access to unlock SCU_WDTSCON0
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked

    // Modify Access to clear ENDINIT
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked

    CCU60_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCY

    // Password Access to unlock SCU_WDTSCON0
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked

    // Modify Access to set ENDINIT
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked


    // CCU60 T12 configurations
    while((CCU60_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0);// wait until CCU60 module enabled

    CCU60_TCTR0.U &= ~(0x7 << T12CLK_BIT_LSB_IDX);      // f_T12 = f_CCU6 / prescaler
    CCU60_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX;         // f_CCU6 = 50 MHz, prescaler = 1024
    CCU60_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;         // f_T12 = 48,828 Hz

    CCU60_TCTR0.U &= ~(0x1 << CTM_BIT_LSB_IDX);         // T12 auto reset when period match (PM) occur


    CCU60_T12PR.U = 24414 -1;                       // PM interrupt freq. = f_T12 / (T12PR + 1)
    CCU60_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX;         // load T12PR from shadow register

    CCU60_T12.U = 0;                                // clear T12 counter register


    // CCU60 T12 PM interrupt setting
    CCU60_INP.U &= ~(0x3 << INPT12_BIT_LSB_IDX);        // service request output SR0 selected
    CCU60_IEN.U |= 0x1 << ENT12PM_BIT_LSB_IDX;          // enable T12 PM interrupt


    // SRC setting for CCU60
    SRC_CCU6_CCU60_SR0.U &= ~(0xFF << SRPN_BIT_LSB_IDX);
    SRC_CCU6_CCU60_SR0.U |= 0x0B << SRPN_BIT_LSB_IDX;   // set priority 0x0B

    SRC_CCU6_CCU60_SR0.U &= ~(0x3 << TOS_BIT_LSB_IDX);  // CPU0 service T12 PM interrupt

    SRC_CCU6_CCU60_SR0.U |= 0x1 << SRE_BIT_LSB_IDX;     // SR0 enabled


    // CCU60 T12 counting start
    CCU60_TCTR4.U = 0x1 << T12RS_BIT_LSB_IDX;           // T12 start counting
```

Core 1 SW 프로그래밍
:Button, ERU 함수

- Core 1 에서 수행할 동작
  - Button 초기화 (general input 설정)
  - Button 외부 GPIO 입력 인터럽트 (ERU) 초기화

Button 초기화 함수는 이전과 동일
(공용체를 사용한 레지스터 접근 방법)

```
43
44  void initButton(void)
45  {
46      P02_IOCR0.B.PC1 = 0x02;      // set P02.1 general input (pull-up connected)
47  }
48
49  void initERU(void)
50  {
51      // ERU setting
52      SCU_EICR1.B.EXIS0 = 0x1;
53      SCU_EICR1.B.FEN0 = 0x1;
54      SCU_EICR1.B.EIEN0 = 0x1;
55      SCU_EICR1.B.INP0 = 0x0;
56      SCU_IGCR0.B.IGP0 = 0x1;
57
58      // SRC Interrupt setting
59      SRC_SCU_SCU_ERU0.B.SRPN = 0x0A;
60      SRC_SCU_SCU_ERU0.B.TOS = 0x1;
61      SRC_SCU_SCU_ERU0.B.SRE = 0x1;
62  }
63
```

Core 1에서 ERU 인터럽트 처리
→ TOS 필드에 0x1 값 write

Core 1 SW 프로그래밍
:ERU ISR 함수

- Core 1 에서 Button이 눌렸을 때 발생하는 ERU 인터럽트를 처리하도록 할 것임.

→ 인터럽트가 발생하면 전역 변수의 값을 toggle

```
33  extern unsigned int glb_flag;
34
35  __interrupt(0x0A) __vector_table(1)
36  void ERU0_ISR(void)
37  {
38      if( glb_flag == 0 )
39          glb_flag = 1;
40      else
41          glb_flag = 0;
42  }
```

다른 파일의 전역 변수 사용 → extern

Core 1 에서 인터럽트를 처리 → 1

인터럽트가 발생하면 전역 변수 glb_flag 의 값 toggle

Core 1 SW 프로그래밍
:main 함수

- Core 1 에서 수행할 동작
  - Button 초기화 (general input 설정)
  - Button 외부 GPIO 입력 인터럽트 (ERU) 초기화

```
int core1_main(void)
{
    IfxCpu_enableInterrupts();

    /* !!WATCHDOG1 IS DISABLED HERE!!
     * Enable the watchdog and service it periodically if it is required
     */
    IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());

    /* Wait for CPU sync event */
    IfxCpu_emitEvent(&g_cpuSyncEvent);
    IfxCpu_waitEvent(&g_cpuSyncEvent, 1);

    initERU();
    initButton();

    while(1)
    {

    }
    return (1);
}
```

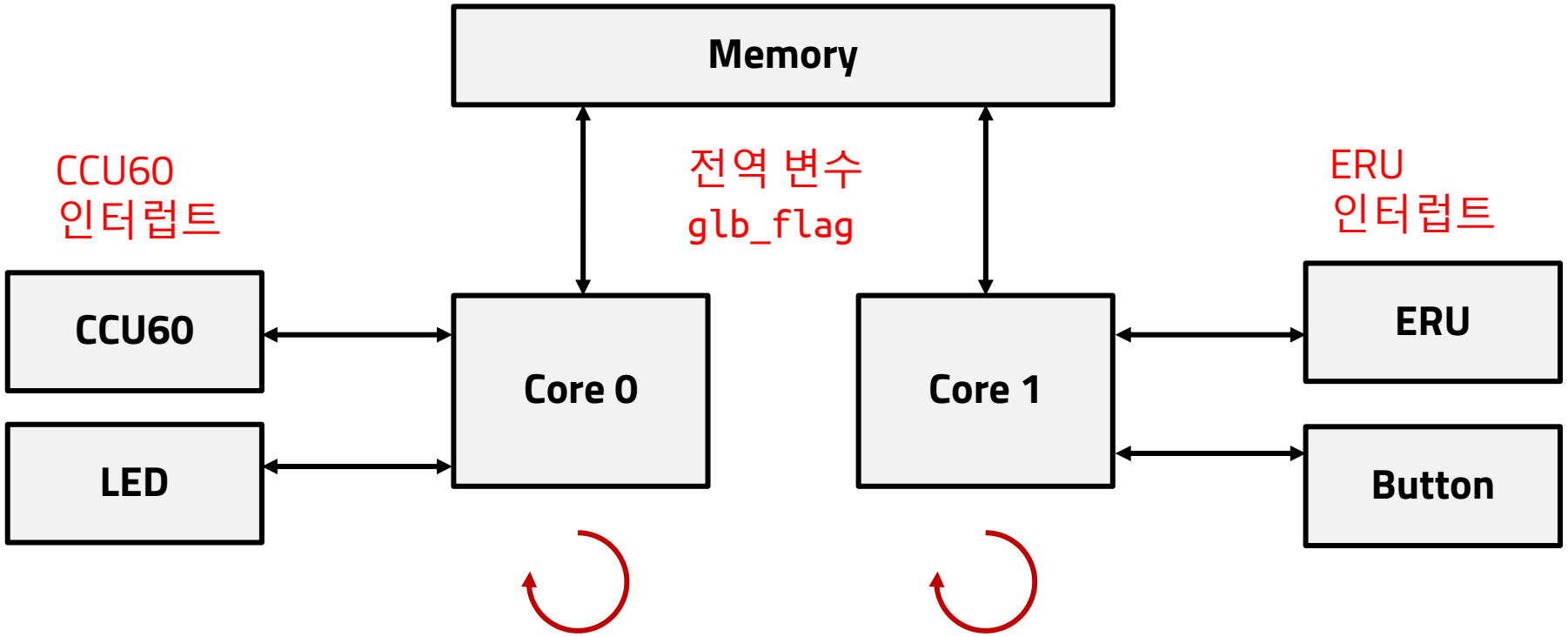
→ 초기화 함수 호출

Core 1 SW 프로그래밍
:전체 코드 check

```c
#include "Ifx_Types.h"
#include "IfxCpu.h"
#include "IfxScuWdt.h"

extern IfxCpu_syncEvent g_cpuSyncEvent;

extern unsigned int glb_flag;

__interrupt(0x0A) __vector_table(1)
void ERU0_ISR(void)
{
    if( glb_flag == 0 )
        glb_flag = 1;
    else
        glb_flag = 0;
}

void initButton(void)
{
    P02_IOCR0.B.PC1 = 0x02;     // set P02.1 general input (pull-up connected)
}

void initERU(void)
{
    // ERU setting
    SCU_EICR1.B.EXIS0 = 0x1;
    SCU_EICR1.B.FEN0 = 0x1;
    SCU_EICR1.B.EIEN0 = 0x1;
    SCU_EICR1.B.INP0 = 0x0;
    SCU_IGCR0.B.IGP0 = 0x1;

    // SRC Interrupt setting
    SRC_SCU_SCU_ERU0.B.SRPN = 0x0A;
    SRC_SCU_SCU_ERU0.B.TOS = 0x1;
    SRC_SCU_SCU_ERU0.B.SRE = 0x1;
}
```

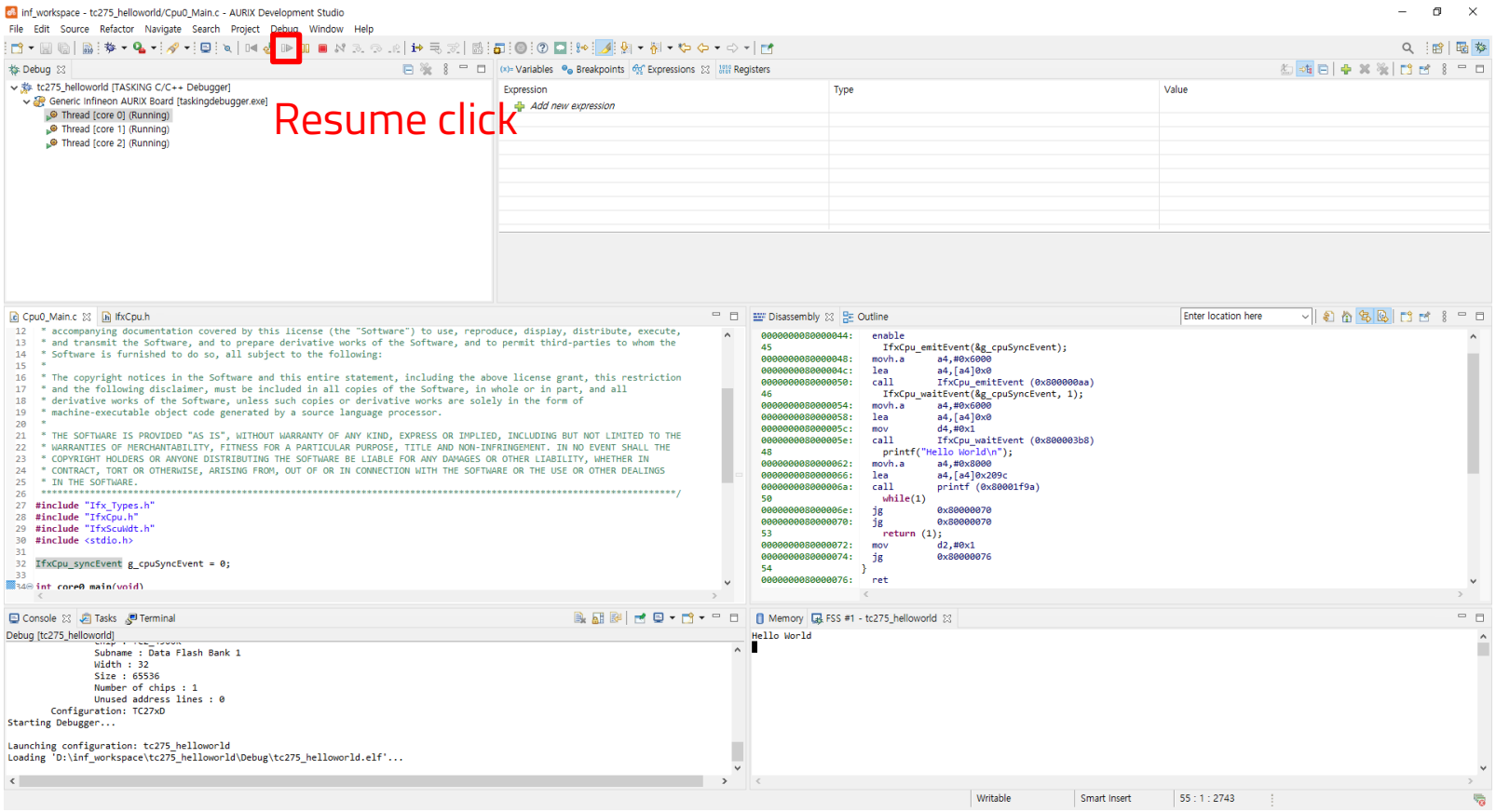
```c
int core1_main(void)
{
    IfxCpu_enableInterrupts();

    /* !!WATCHDOG1 IS DISABLED HERE!!
     * Enable the watchdog and service it periodically if it is required
     */
    IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());

    /* Wait for CPU sync event */
    IfxCpu_emitEvent(&g_cpuSyncEvent);
    IfxCpu_waitEvent(&g_cpuSyncEvent, 1);

    initERU();
    initButton();

    while(1)
    {

    }
    return (1);
}
```

프로그램 수행 개요
:Core 0, 1 역할



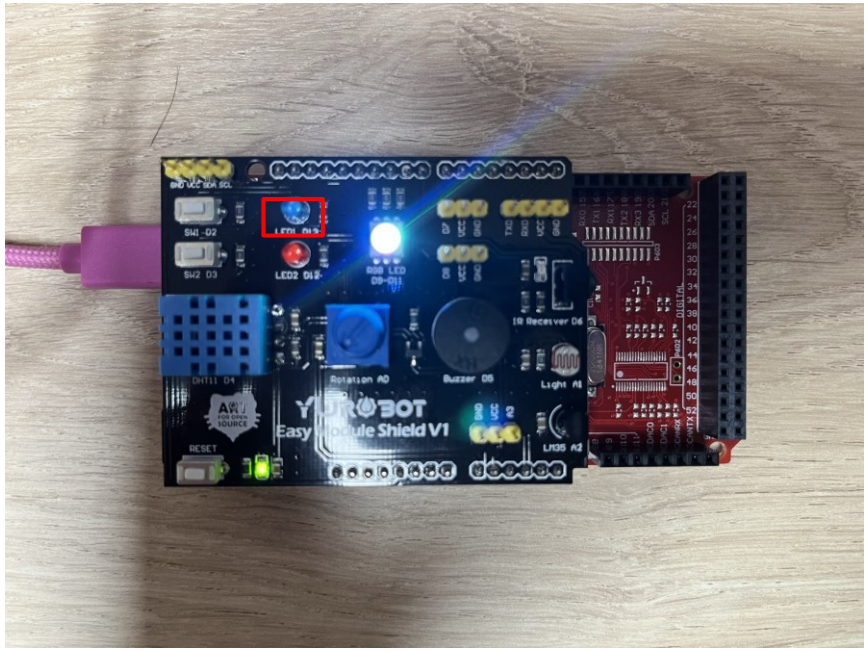
Build 및 Debug

- 프로젝트 빌드 (ctrl + b)
- 디버그 수행하여 보드에 실행 파일 flash



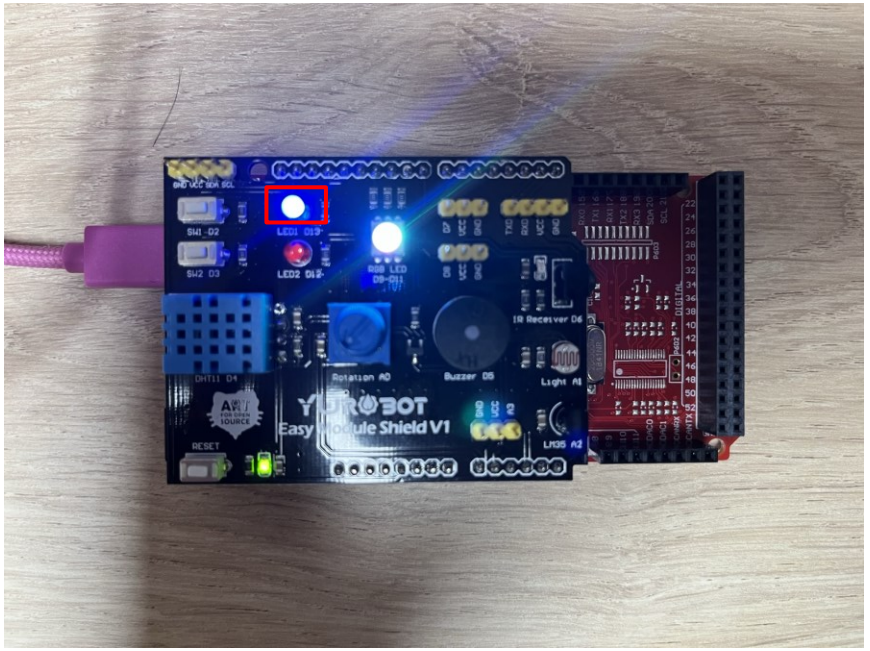
동작 확인

- Button을 1번 누를 때마다 glb_flag 변수의 값이 toggle
  → glb_flag 값이 0이면 LED Blue는 현재 상태 유지
  → glb_flag 값이 1이면 LED Blue는 점멸



glb_flag 변수의 값이 0인 상태

Button
push



glb_flag 변수의 값이 1인 상태

감사합니다. 휴식~~