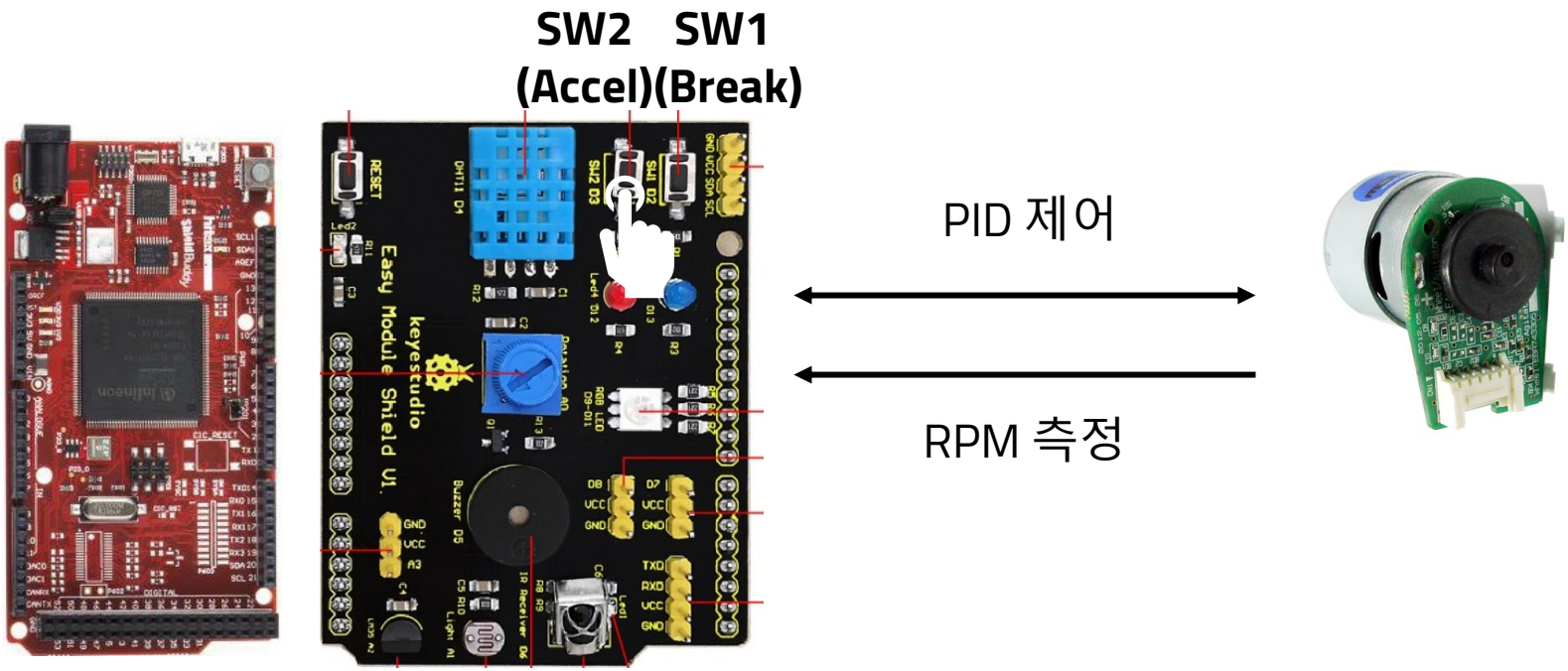
임베디드 기반 SW 개발 프로젝트

AURIX TC275 보드와 모터 사용
엑셀/브레이크의 가속/감속 PID 제어 구현

현대자동차 입문교육
박대진 교수

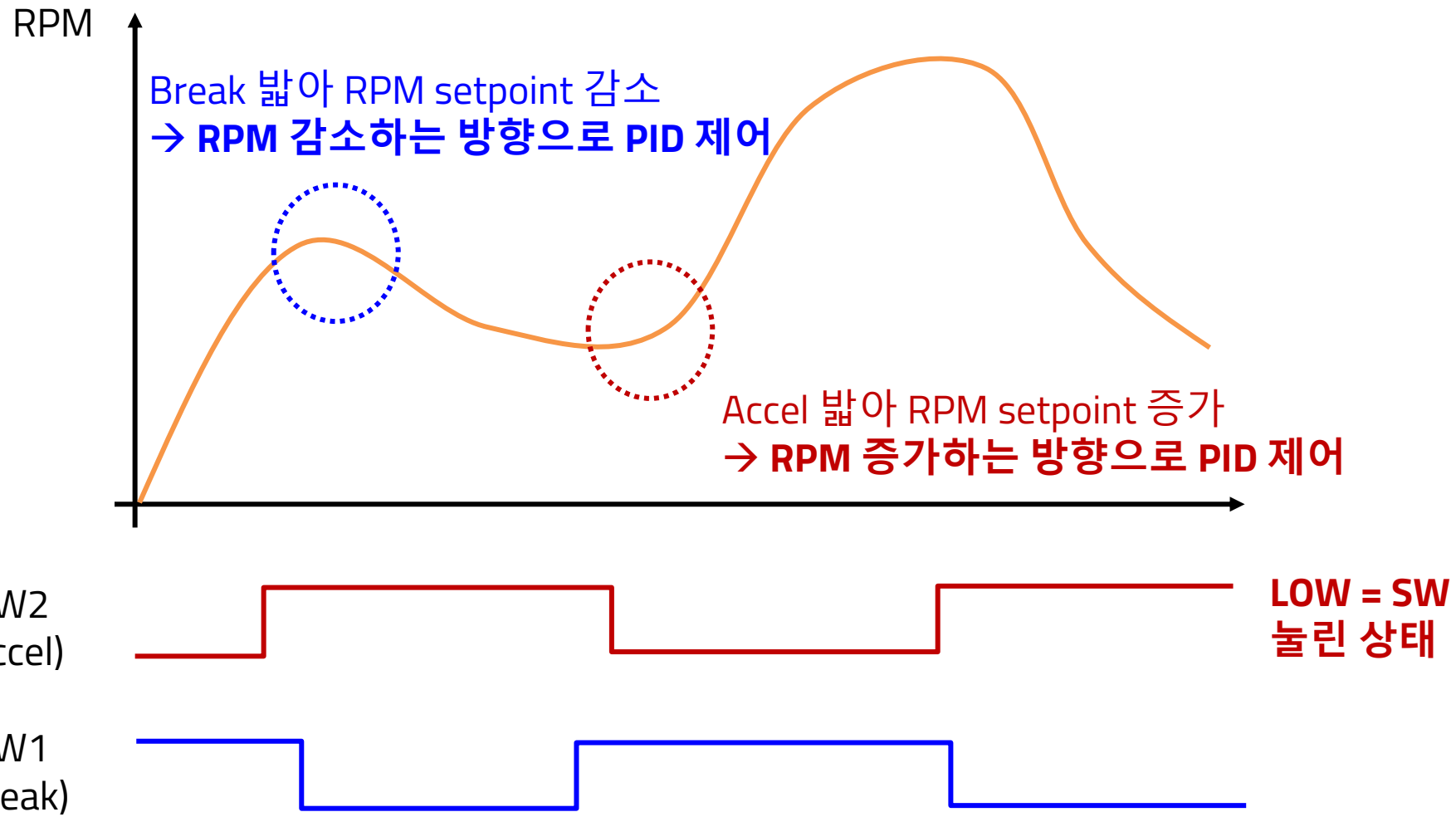
실습 목표

- **TC275 보드의 SW1을 자동차의 엑셀, SW2를 자동차의 브레이크로 가정**
- **SW1을 누르면 모터 RPM의 setpoint를 증가하는 방향으로 PID 제어 (엑셀 밟은 효과)**
- **SW2을 누르면 모터 RPM을 0으로 만드는 방향으로 PID 제어 (브레이크 밟은 효과)**
  1. SW1, SW2 입력 사용
  2. 모터 PID 제어 구현



모터 동작 개요

: SW 입력에 따른 PID 제어 setpoint 증감 방식

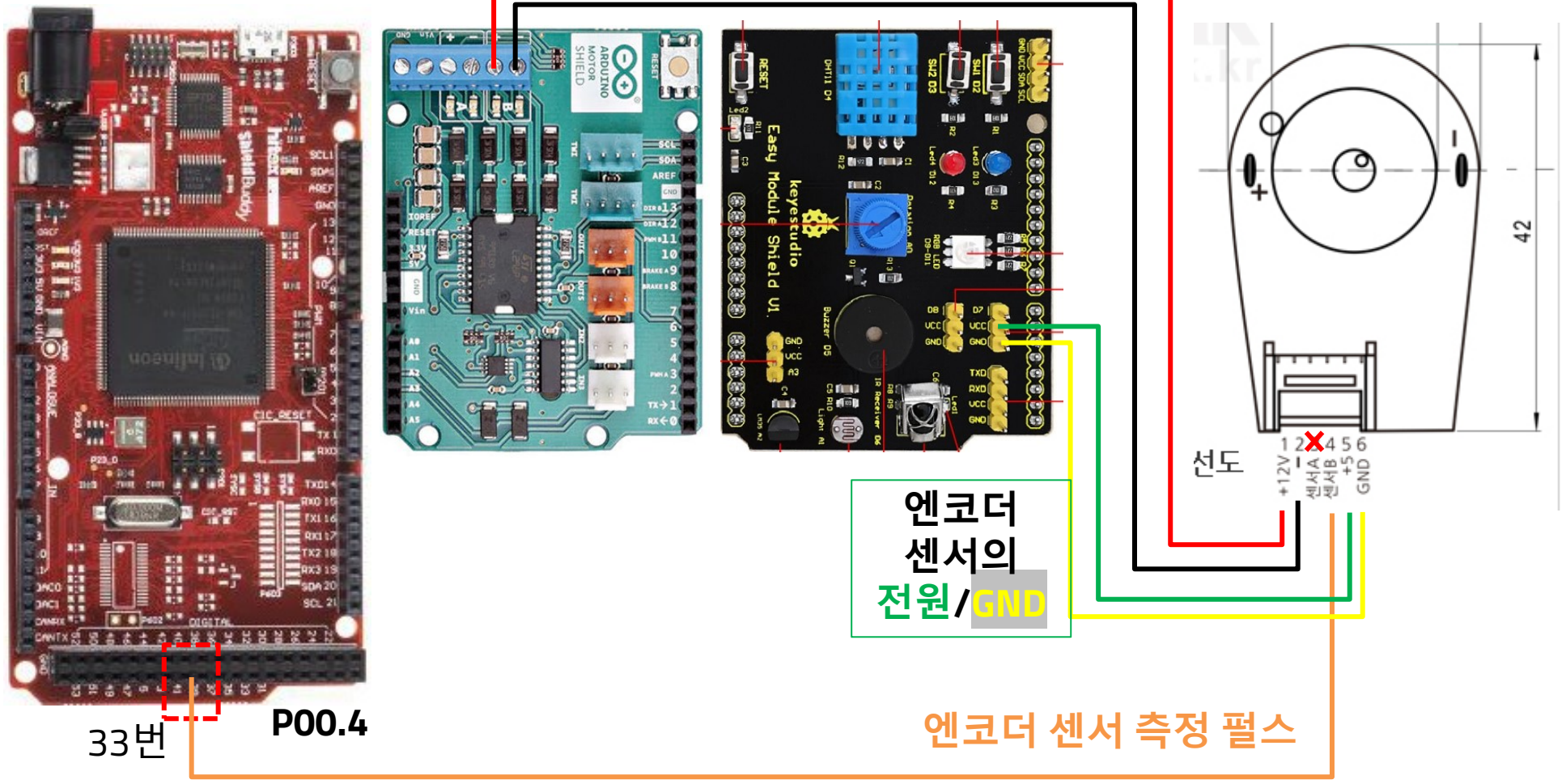


TC275 보드와 모터 연결

모터 배선 연결

- 모터의 5개 라인을 Easy Module 확장 보드 및 TC275 보드에 연결

DC모터의 전원/GND



SW를 통해 모터 RPM을 가속/감속하도록 PID 제어

Lab1

: 헤더 파일 / define 정의 / 전역 변수 / 함수 prototype

```
27
28 #include "Ifx_Types.h"
29 #include "IfxCpu.h"
30 #include "IfxScuWdt.h"
31
32 #include "IfxCcu6_reg.h"
33 #include "IfxGtm_reg.h"
34
35 #include "Ifx_Console.h"
36 #include <ASCLIN_UART.h>
37
38 IfxCpu_syncEvent g_cpuSyncEvent = 0;
39
40 // SCU registers
41 #define LCK_BIT_LSB_IDX      1
42 #define ENDINIT_BIT_LSB_IDX 0
43
44 //CCU60 registers
45 #define DISS_BIT_LSB_IDX     1
46 #define DISR_BIT_LSB_IDX     0
47
48 void initMotor(void);
49 void initGTM(void);
50 void initERU(void);
51 void initCCU60(void);
52 void initButton(void);
53 void initLED(void);
54
```

```
54
55 // PID Parameters (for acc)
56 float Kp = 0.005;
57 float Ki = 0.01;
58 float Kd = 0.05;
59
60
61 // Error variables
62 float err = 0;
63 float pre_err = 0;
64
65 // Motor speed from sensor
66 float RPM;  // MAX = 6800 RPM
67 float setpoint = 400.0;
68 float _P = 0;
69 float _I = 0;
70 float _D = 0;
71 int motor_duty = 0;
72
73 volatile unsigned int rotate = 0;
74
```

PID 제어 구현을 위해 사용되는 전역 변수들

Lab2

: ERU (외부 인터럽트) ISR

```
75  __interrupt(0x0A) __vector_table(0)
76  void ERU2_ISR(void)
77  {
78      //P10_OUT.B.P1 ^= 0x1;
79      rotate++;
80  }
81
```

엔코더 모터로부터 회전 수 카운트
→ RPM 계산을 위해 사용

Lab3
: CCU6 타이머 인터럽트

일정 시간마다 PWM duty 변경하여 모터 RPM에 PID 제어 적용하기 위한 타이머

```
// 0.25 sec = 250 ms
__interrupt(0x0D) __vector_table(0)
void CCU60_T12_ISR(void)
{
    P10_OUT.B.P1 ^= 0x1;

    RPM = (float)(rotate*30);

    if( P02_IN.B.P0 == 0x0 ) {        // break (sw1)
        setpoint -= 10;
    }
    else if( P02_IN.B.P1 == 0x0 ) {   // acc (sw2)
        setpoint += 5;
    }
    else {
        setpoint = 400;
    }

    if( setpoint > 2000 )
        setpoint = 2000;
    else if( setpoint < 0 )
        setpoint = 0;

    err = setpoint - RPM;

    _P = Kp * err;
    _I += Ki * err * 0.25;
    _D = Kd * ( (err - pre_err) / 0.25 );

    motor_duty += (int)(_P + _I + _D);

    if( motor_duty >= 12500 ) {
        motor_duty = 12500 - 1;
    } else if( motor_duty <= 0 ) {
        motor_duty = 0;
    }
    GTM_TOM0_CH3_SR1.B.SR1 = motor_duty;

    Ifx_Console_print("%d %d\n\r", (unsigned int)RPM, motor_duty);

    rotate = 0;
}
```

Lab4

: main 함수

PWM 출력 시작

RPM, duty를 로그로 확인하기 위한 UART 출력 설정

```
123
126 int core0_main(void)
127 {
128     IfxCpu_enableInterrupts();
129
130     /* !!WATCHDOG0 AND SAFETY WATCHDOG ARE DISABLED HERE!!
131      * Enable the watchdogs and service them periodically if it is required
132      */
133     IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());
134     IfxScuWdt_disableSafetyWatchdog(IfxScuWdt_getSafetyWatchdogPassword());
135
136     /* Wait for CPU sync event */
137     IfxCpu_emitEvent(&g_cpuSyncEvent);
138     IfxCpu_waitEvent(&g_cpuSyncEvent, 1);
139
140
141     initMotor();
142     initButton();
143     initLED();
144     initERU();
145     initGTM();
146     initCCU60();
147
148
149     GTM_TOM0_TGC0_GLB_CTRL.B.HOST_TRIG = 0x1;
150
151     initUart();
152
153     Ifx_Console_print("RPM Duty\n\r");
154
155     while(1)
156     {
157     }
158
159     return (1);
160 }
161
```

Lab5

: ERU configuration

엔코더로부터 회전할 때마다 입력되는 펄스를
카운트 하기 위한 ERU 인터럽트 설정

```
162
163 void initERU(void)
164 {
165     // Motor Encoder
166     // ERU2 -> IN22 -> OGU0
167     SCU_EICR1.B.EXIS0 = 0x2;     // ERS2 - Input 2 (P00.4)
168     SCU_EICR1.B.FEN0  = 0x1;     // falling edge
169     SCU_EICR1.B.REN0  = 0x0;
170     SCU_EICR1.B.EIEN0 = 0x1;
171     SCU_EICR1.B.INP0  = 0x0;     // OGU0
172
173     SCU_IGCR0.B.IGP0  = 0x1;
174
175     // SRC Interrupt setting
176     SRC_SCU_SCU_ERU0.B.SRPN = 0x0A;
177     SRC_SCU_SCU_ERU0.B.TOS  = 0x0;
178     SRC_SCU_SCU_ERU0.B.SRE  = 0x1;
179 }
```

Lab6

: 모터 속도 결정하는 PWM 생성 위한 GTM configuration

```
void initGTM(void)
{
    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);

    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);

    GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX); // enable VADC

    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);

    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);

    while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0);

    GTM_CMU_FXCLK_CTRL.B.FXCLK_SEL = 0x0;
    GTM_CMU_CLK_EN.B.EN_FXCLK = 0x2;

    // P10.3 -> TOM0_3
    GTM_TOM0_TGC0_GLB_CTRL.B.UPEN_CTRL3 = 0x2;
    GTM_TOM0_TGC0_ENDIS_CTRL.B.ENDIS_CTRL3 = 0x2;
    GTM_TOM0_TGC0_OUTEN_CTRL.B.OUTEN_CTRL3 = 0x2;

    GTM_TOM0_CH3_CTRL.B.SL = 0x1;
    GTM_TOM0_CH3_CTRL.B.CLK_SRC_SR = 0x1;
    GTM_TOM0_CH3_SR0.U = 12500 - 1;

    GTM_TOUTSEL7.B.SEL1 = 0x0;
}
```

Lab7

: CCU60 타이머 configuration

```
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

    // f_T12 = f_CCU6 / prescaler
    CCU60_TCTR0.B.T12CLK = 0x2;     // f_CCU6 = 50 MHz, prescaler = 1024
    CCU60_TCTR0.B.T12PRE = 0x1;     // f_T12 = 48,828 Hz

    CCU60_TCTR0.B.CTM = 0x0;         // T12 auto reset when period match (PM) occur

    CCU60_T12PR.B.T12PV = 12207 - 1;                    // PM interrupt freq. = f_T12 / (T12PR + 1)
    CCU60_TCTR4.B.T12STR = 0x1;         // load T12PR from shadow register

    CCU60_T12.B.T12CV = 0;                              // clear T12 counter register

    // CCU60 T12 PM interrupt setting
    CCU60_INP.B.INPT12 = 0x0;        // service request output SR0 selected
    CCU60_IEN.B.ENT12PM = 0x1;          // enable T12 PM interrupt
    // SRC setting for CCU60
    SRC_CCU6_CCU60_SR0.B.SRPN = 0x0D;    // set priority 0x0B
    SRC_CCU6_CCU60_SR0.B.TOS = 0x0;      // CPU0 service T12 PM interrupt
    SRC_CCU6_CCU60_SR0.B.SRE = 0x1;      // SR0 enabled
    // CCU60 T12 counting start
    CCU60_TCTR4.B.T12RS = 0x1;           // T12 start counting
}
```

Lab8

: 모터, SW버튼, LED configuration

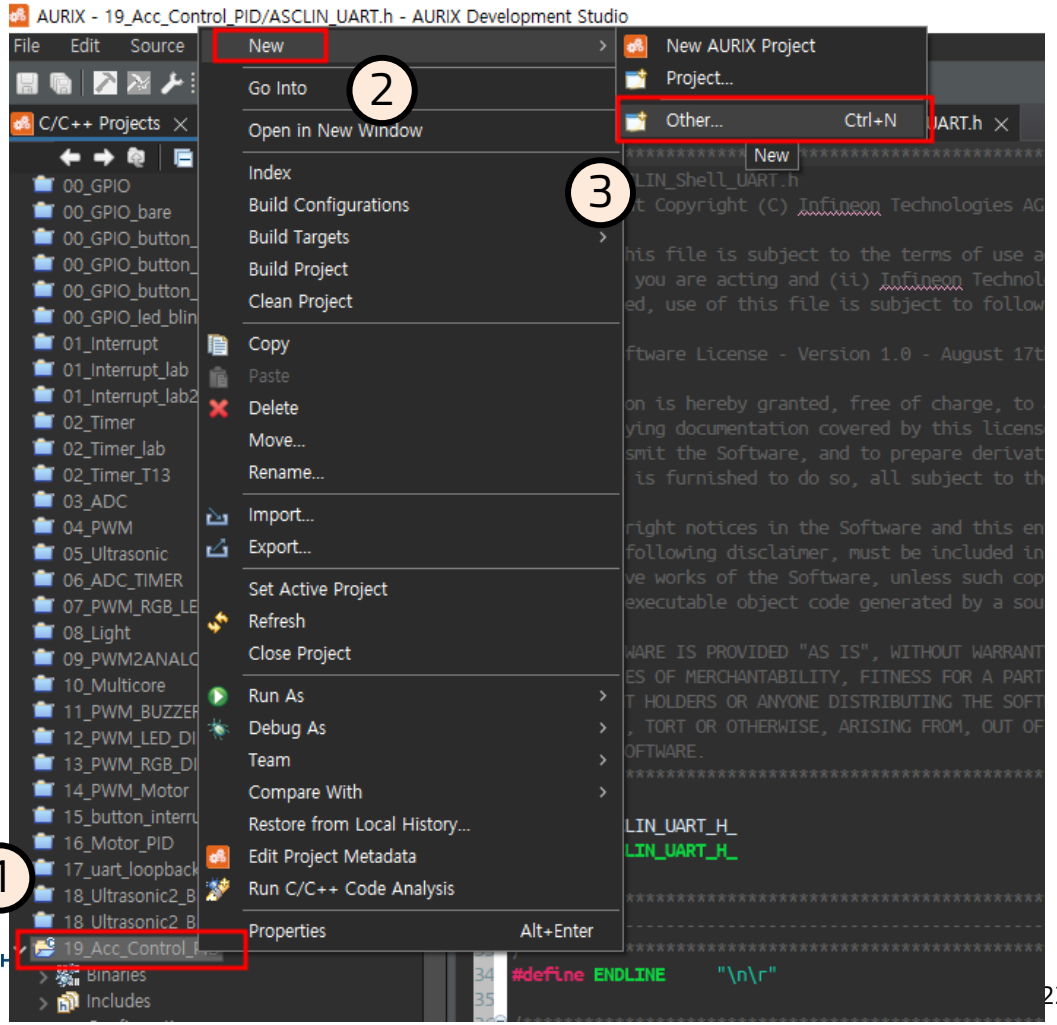
```
262 void initMotor(void)
263 {
264     P10_IOCR0.B.PC2 = 0x10;      // Direction (P10.2)
265     P10_IOCR0.B.PC3 = 0x11;      // PWM (P10.3)
266
267     // encoder input
268     P00_IOCR4.B.PC4 = 0x01;
269 }
270
271 void initButton(void)
272 {
273     P02_IOCR0.B.PC1 = 0x02;      // set P02.1 general input
274     P02_IOCR0.B.PC0 = 0x02;      // set P02.0 general input
275 }
276
277 void initLED(void)
278 {
279     P10_IOCR0.B.PC1 = 0x10;
280 }
281
```

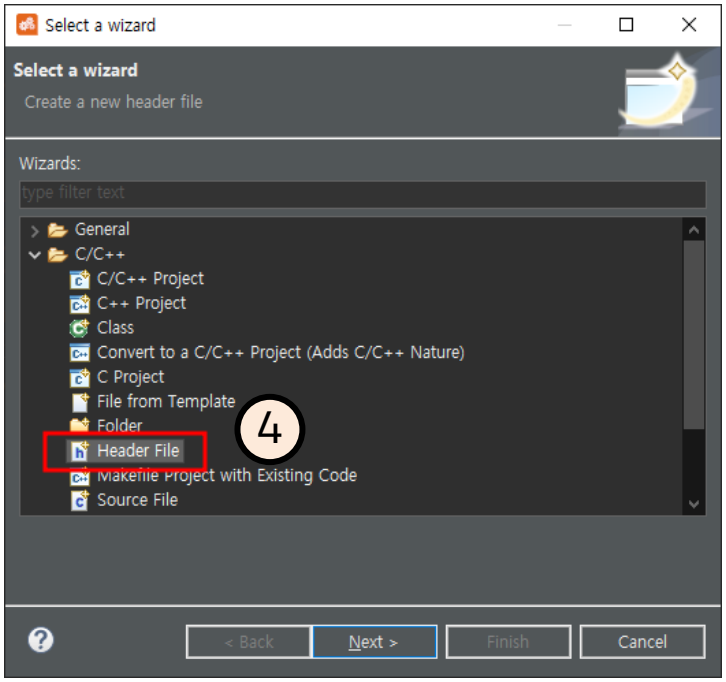
Lab9

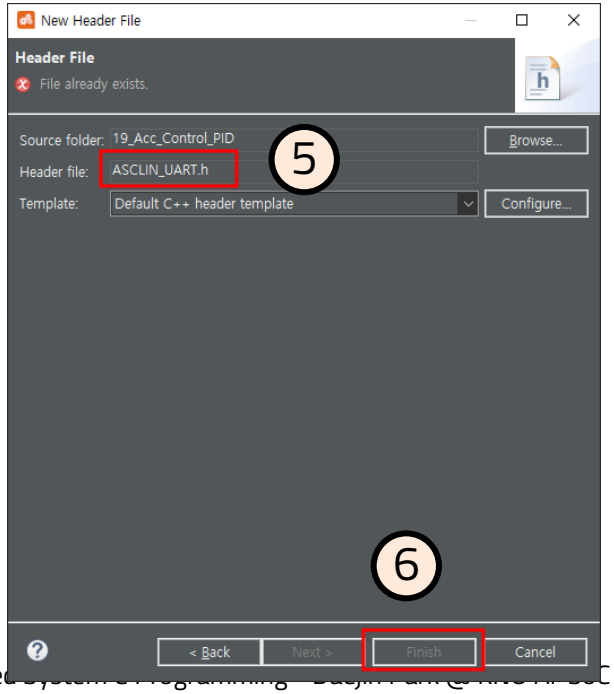
: UART configuration 헤더 파일

새로운 파일 생성

ASCLIN_UART.h

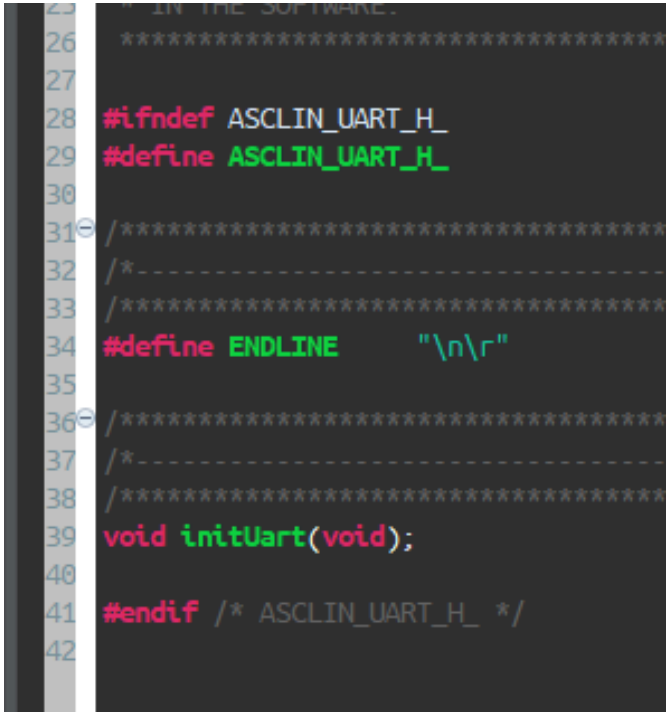






Lab9

: UART configuration 헤더 파일



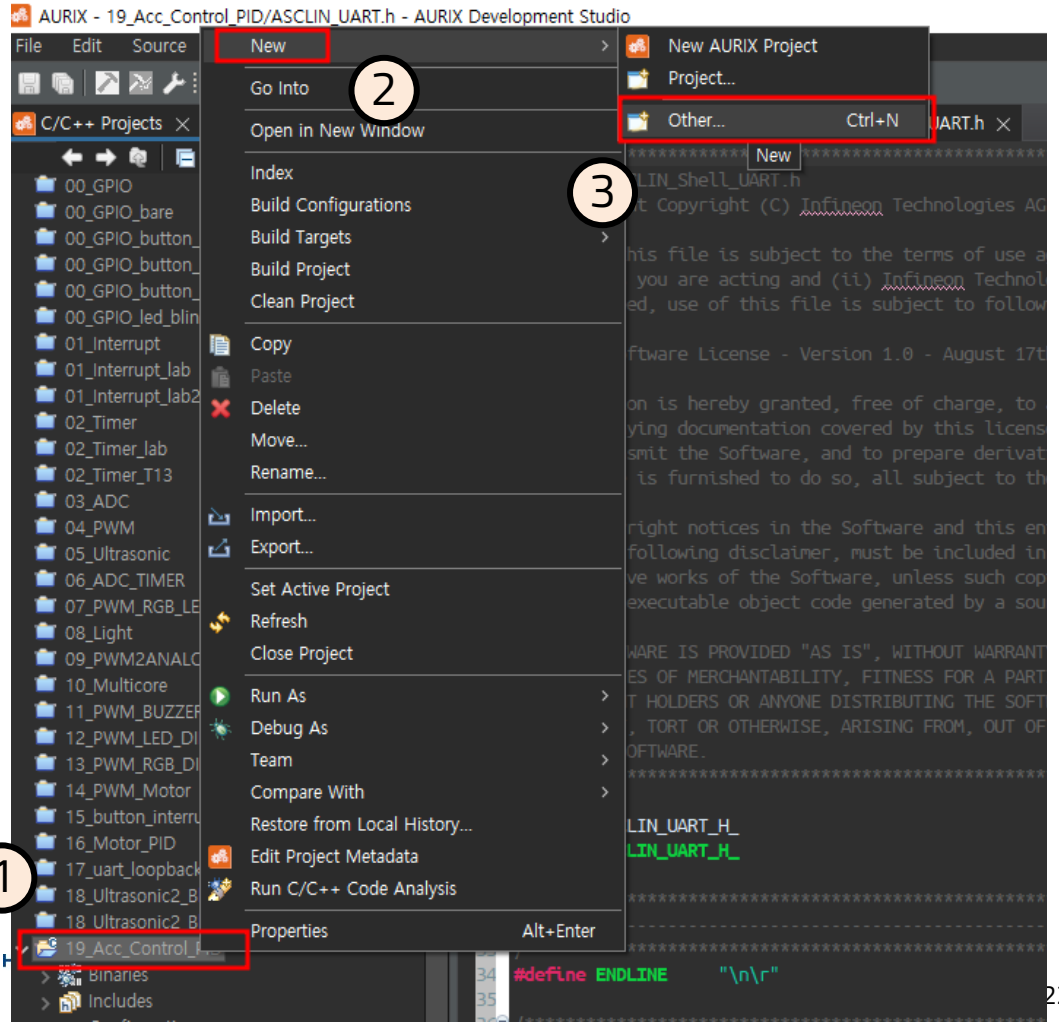
```
 * IN THE SOFTWARE.
 *********************************************

#ifndef ASCLIN_UART_H_
#define ASCLIN_UART_H_

/*********************************************
 *-------------------------------------------
 *********************************************
#define ENDLINE     "\n\r"

/*********************************************
 *-------------------------------------------
 *********************************************
void initUart(void);

#endif /* ASCLIN_UART_H_ */

```

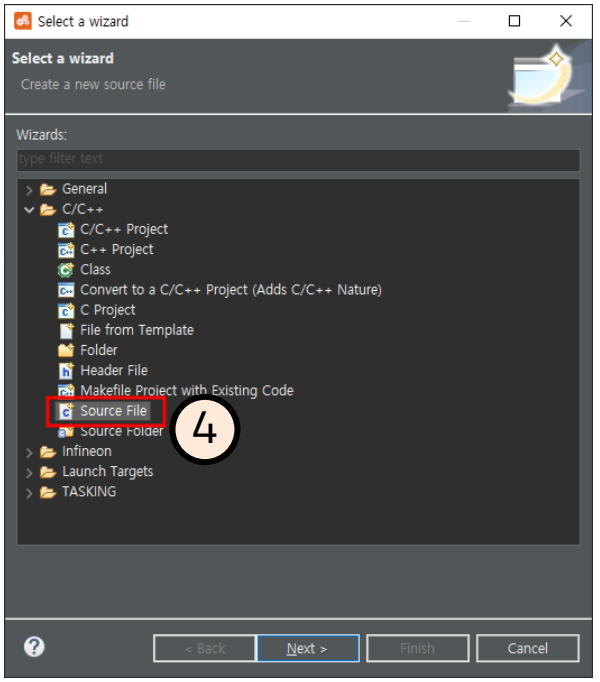
Lab10

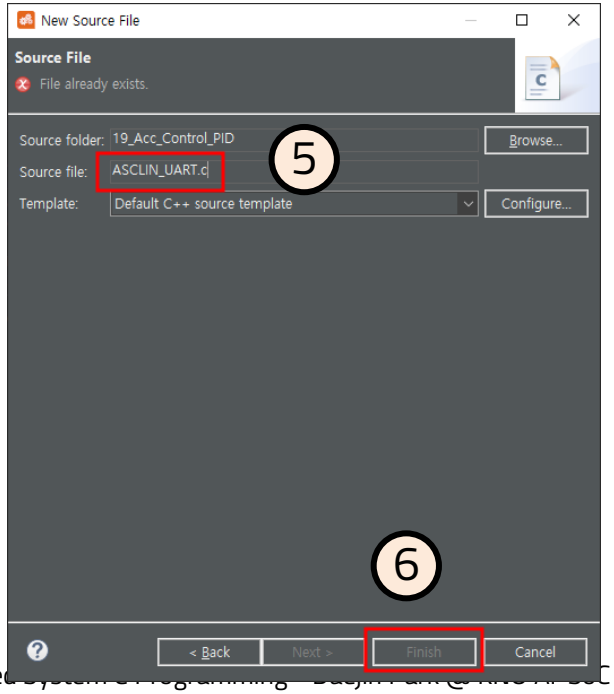
: UART configuration 소스 파일

새로운 파일 생성

ASCLIN_UART.c







Lab10

: UART configuration 소스 파일

```c
/*-----------------------------------------------------------Includes-----
/*****************************************************************************
#include <ASCLIN_UART.h>
#include "Ifx_Types.h"
#include "IfxAsclin_Asc.h"
#include "Ifx_Console.h"

/*****************************************************************************
/*-------------------------------------------------------Macros-------
/*****************************************************************************
/* Communication parameters */
#define ISR_PRIORITY_ASCLIN_TX      8
#define ISR_PRIORITY_ASCLIN_RX      4
#define ISR_PRIORITY_ASCLIN_ER      12
#define ASC_TX_BUFFER_SIZE          256
#define ASC_RX_BUFFER_SIZE          256
#define ASC_BAUDRATE                115200

/*****************************************************************************
/*-------------------------------------------------Function Prototypes
/*****************************************************************************
void initSerialInterface(void);

/*****************************************************************************
/*-------------------------------------------------Global variables--
/*****************************************************************************
IfxStdIf_DPipe  g_ascStandardInterface;
IfxAsclin_Asc   g_asclin;

/* The transfer buffers allocate memory for the data itself and for
 * 8 more bytes have to be added to ensure a proper circular buffer
 * the address to which the buffers have been located.
 */
uint8 g_uartTxBuffer[ASC_TX_BUFFER_SIZE + sizeof(Ifx_Fifo) + 8];
uint8 g_uartRxBuffer[ASC_RX_BUFFER_SIZE + sizeof(Ifx_Fifo) + 8];
```

```c
/*****************************************************************************
IFX_INTERRUPT(asc0TxISR, 0, ISR_PRIORITY_ASCLIN_TX);
void asc0TxISR(void)
{
    IfxStdIf_DPipe_onTransmit(&g_ascStandardInterface);
}

IFX_INTERRUPT(asc0RxISR, 0, ISR_PRIORITY_ASCLIN_RX);
void asc0RxISR(void)
{
    IfxStdIf_DPipe_onReceive(&g_ascStandardInterface);
}

IFX_INTERRUPT(asc0ErrISR, 0, ISR_PRIORITY_ASCLIN_ER);
void asc0ErrISR(void)
{
    IfxStdIf_DPipe_onError(&g_ascStandardInterface);
}
```

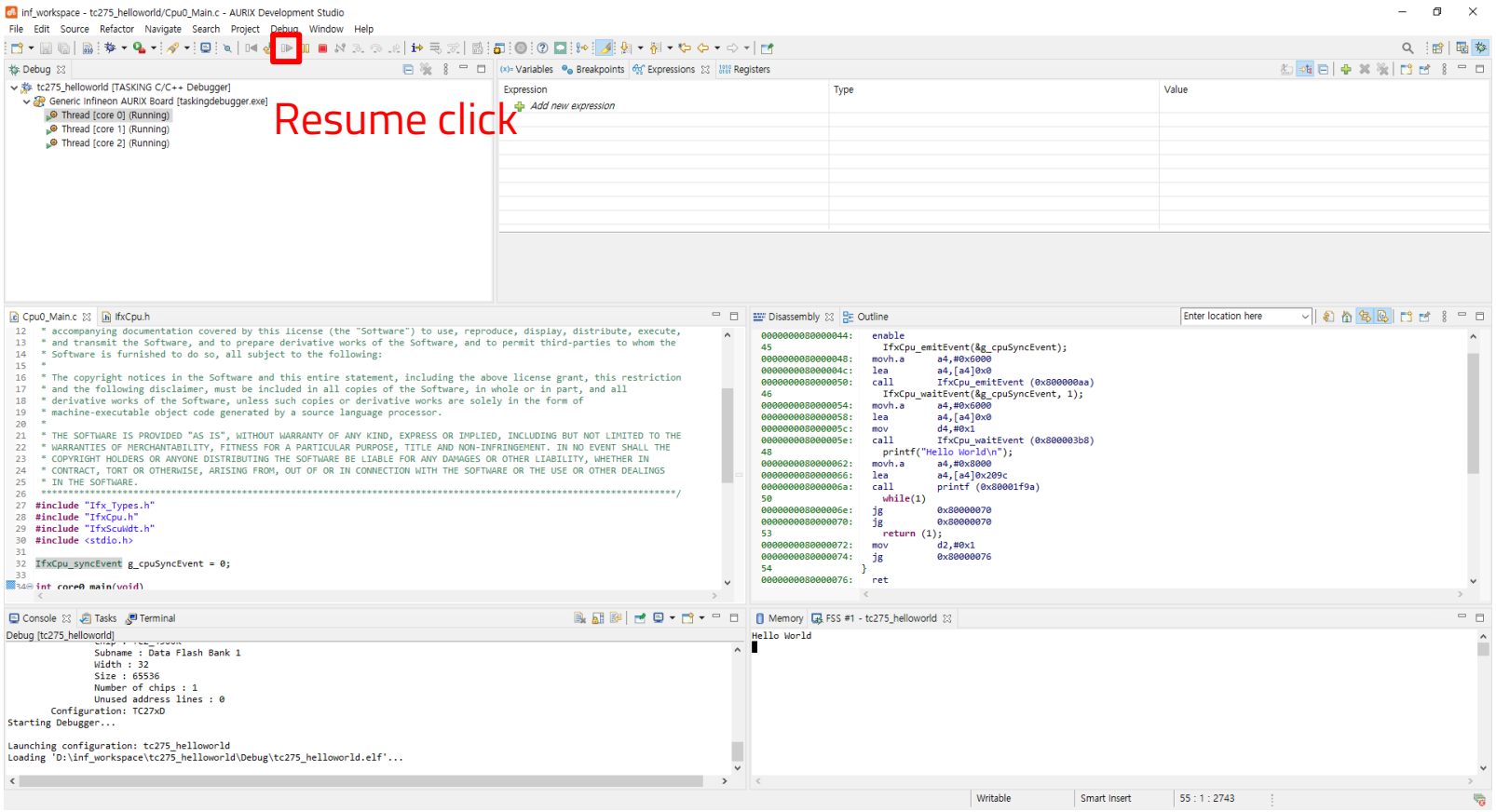
Lab10 (계속)

: UART configuration 소스 파일

```
/* Function to initialize ASCLIN module */
void initSerialInterface(void)
{
    IfxAsclin_Asc_Config ascConf;

    /* Set default configurations */
    IfxAsclin_Asc_initModuleConfig(&ascConf, &MODULE_ASCLIN3); /* Initialize the structure with default values      */

    /* Set the desired baud rate */
    ascConf.baudrate.baudrate = ASC_BAUDRATE;                                   /* Set the baud rate in bit/s       */
    ascConf.baudrate.oversampling = IfxAsclin_OversamplingFactor_16;            /* Set the oversampling factor      */

    /* Configure the sampling mode */
    ascConf.bitTiming.medianFilter = IfxAsclin_SamplesPerBit_three;             /* Set the number of samples per bit*/
    ascConf.bitTiming.samplePointPosition = IfxAsclin_SamplePointPosition_8;    /* Set the first sample position    */

    /* ISR priorities and interrupt target */
    ascConf.interrupt.txPriority = ISR_PRIORITY_ASCLIN_TX;  /* Set the interrupt priority for TX events            */
    ascConf.interrupt.rxPriority = ISR_PRIORITY_ASCLIN_RX;  /* Set the interrupt priority for RX events            */
    ascConf.interrupt.erPriority = ISR_PRIORITY_ASCLIN_ER;  /* Set the interrupt priority for Error events         */
    ascConf.interrupt.typeOfService = IfxSrc_Tos_cpu0;

    /* Pin configuration */
    const IfxAsclin_Asc_Pins pins = {
            .cts        = NULL_PTR,                     /* CTS pin not used                                         */
            .ctsMode    = IfxPort_InputMode_pullUp,
            .rx         = &IfxAsclin3_RXD_P32_2_IN ,    /* Select the pin for RX connected to the USB port      */
            .rxMode     = IfxPort_InputMode_pullUp,     /* RX pin                                               */
            .rts        = NULL_PTR,                     /* RTS pin not used                                     */
            .rtsMode    = IfxPort_OutputMode_pushPull,
            .tx         = &IfxAsclin3_TX_P15_7_OUT,     /* Select the pin for TX connected to the USB port      */
            .txMode     = IfxPort_OutputMode_pushPull,  /* TX pin                                               */
            .pinDriver  = IfxPort_PadDriver_cmosAutomotiveSpeed1
    };
    ascConf.pins = &pins;

    /* FIFO buffers configuration */
    ascConf.txBuffer = g_uartTxBuffer;                  /* Set the transmission buffer                          */
    ascConf.txBufferSize = ASC_TX_BUFFER_SIZE;          /* Set the transmission buffer size                     */
    ascConf.rxBuffer = g_uartRxBuffer;                  /* Set the receiving buffer                             */
    ascConf.rxBufferSize = ASC_RX_BUFFER_SIZE;          /* Set the receiving buffer size                        */

    /* Init ASCLIN module */
    IfxAsclin_Asc_initModule(&g_asclin, &ascConf);      /* Initialize the module with the given configuration   */
}

/* Function to initialize the Shell */
void initUart(void)
{
    /* Initialize the hardware peripherals */
    initSerialInterface();

    /* Initialize the Standard Interface */
    IfxAsclin_Asc_stdIfDPipeInit(&g_ascStandardInterface, &g_asclin);

    /* Initialize the Console */
    Ifx_Console_init(&g_ascStandardInterface);
}
```

Build 및 Debug

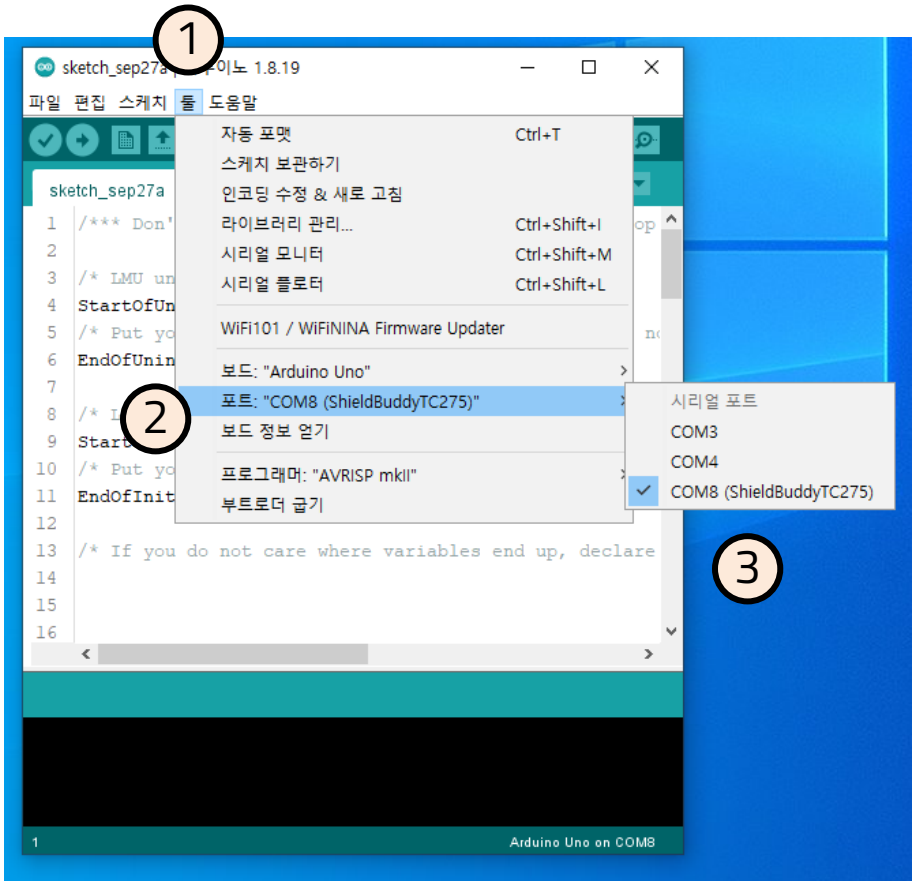
- 프로젝트 빌드 (ctrl + b)
- 디버그 수행하여 보드에 실행 파일 flash



TC275에서 출력되는 로그 확인

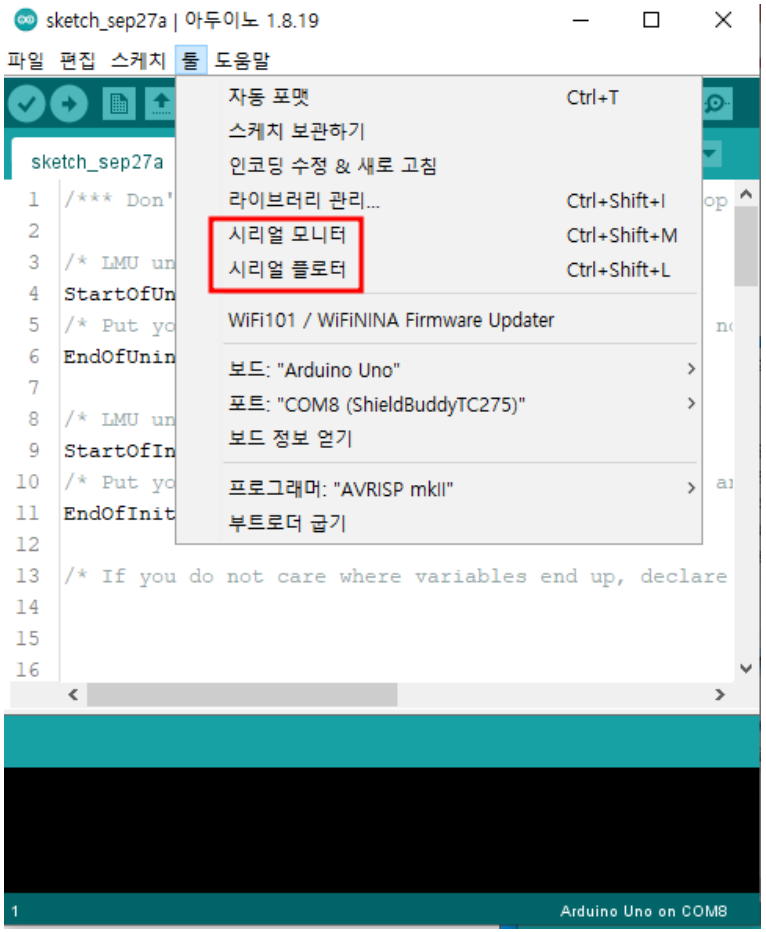
- 아두이노 IDE 설치

- TC275에 해당하는 포트 연결
  - COM* 중에서 TC275 선택



TC275에서 출력되는 로그 확인 (계속)

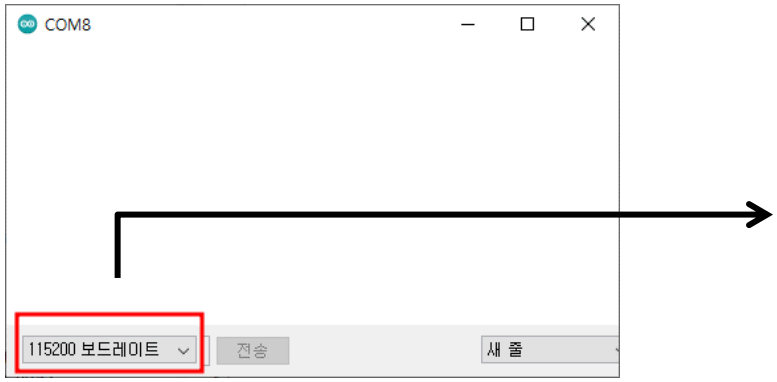
- 문자로 로그를 확인하고 싶으면
  - 시리얼 모니터
- 그래프 형태로 로그를 확인하고 싶으면
  - 시리얼 플로터



TC275에서 출력되는 로그 확인 (계속)

: RPM, Duty 변화 추이 그래프로 확인

- 시리얼 플로터 **open**



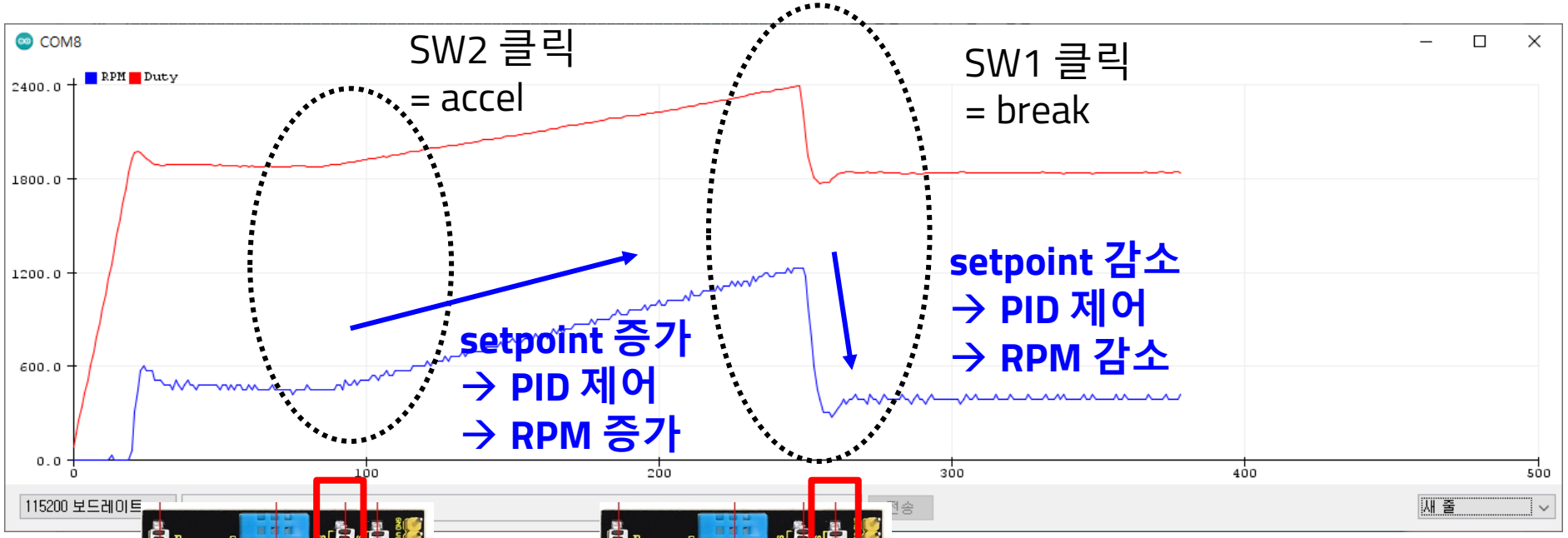
ASCLIN_UART.c 파일에서 정의됨

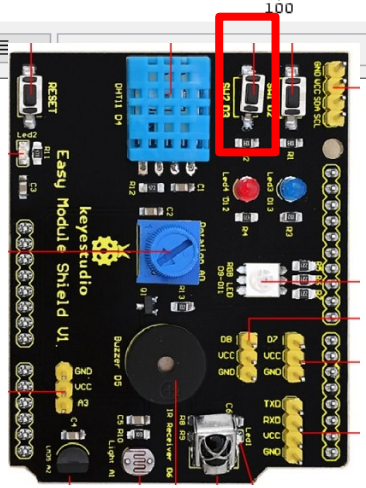
```
37  /*---------------------------------------------
38  /*********************************************
39  /* Communication parameters */
40  #define ISR_PRIORITY_ASCLIN_TX      8
41  #define ISR_PRIORITY_ASCLIN_RX      4
42  #define ISR_PRIORITY_ASCLIN_ER      12
43  #define ASC_TX_BUFFER_SIZE          256
44  #define ASC_RX_BUFFER_SIZE          256
45  #define ASC_BAUDRATE                115200
46
```

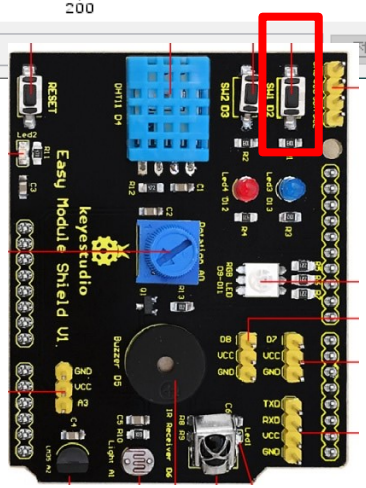
중요) UART 속도 설정 = 115200 보드레이트

TC275에서 출력되는 로그 확인 (계속)

: RPM, Duty 변화 추이 그래프로 확인







감사합니다. 휴식~~