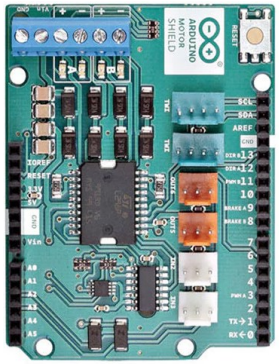
임베디드 기반 SW 개발 프로젝트

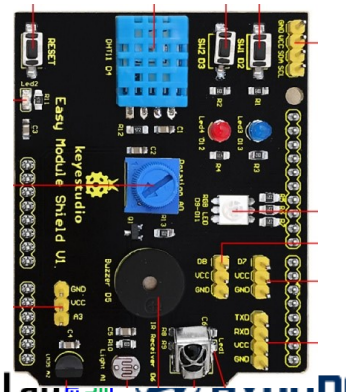
AURIX TC275 보드 / 모터 / 초음파 센서 사용
초음파 센서 거리 측정 기반 모터 RPM PID 제어하는
스마트 크루즈 기능 구현

현대자동차 교육
박대진 교수

실습 목표

- **TC275** 보드와 연결된 초음파 센서로부터 물체와의 거리를 측정
- 물체와의 거리가 가까울수록 모터의 **RPM**을 감소, 멀수록 **RPM**을 증가하는 방향으로 **setpoint** 설정
- 설정된 **setpoint**에 따라 모터의 **RPM**을 **PID** 제어





PID 제어

RPM 측정

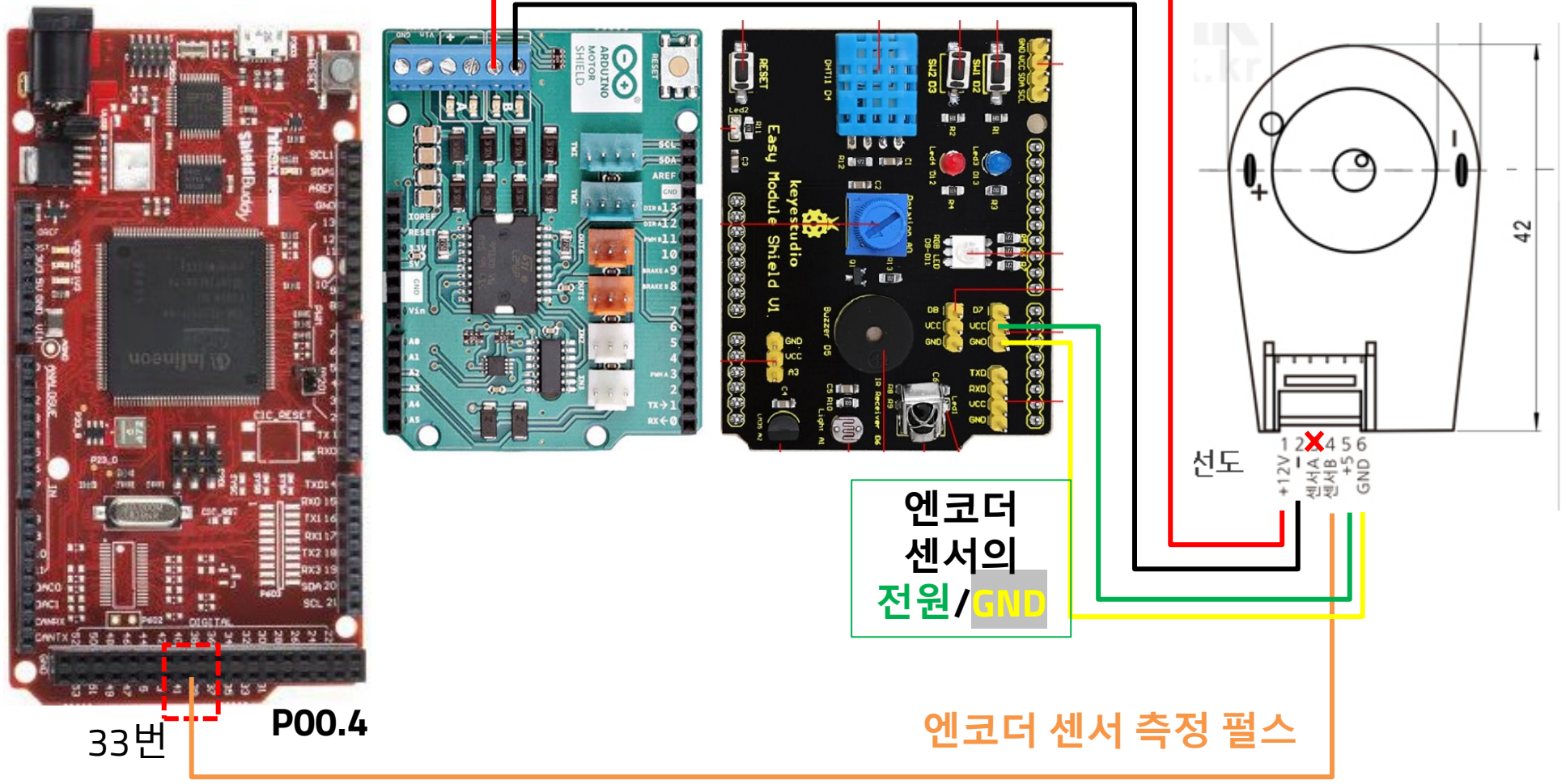
거리 측정

TC275 보드와 모터 / 초음파 센서 연결

모터 배선 연결

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

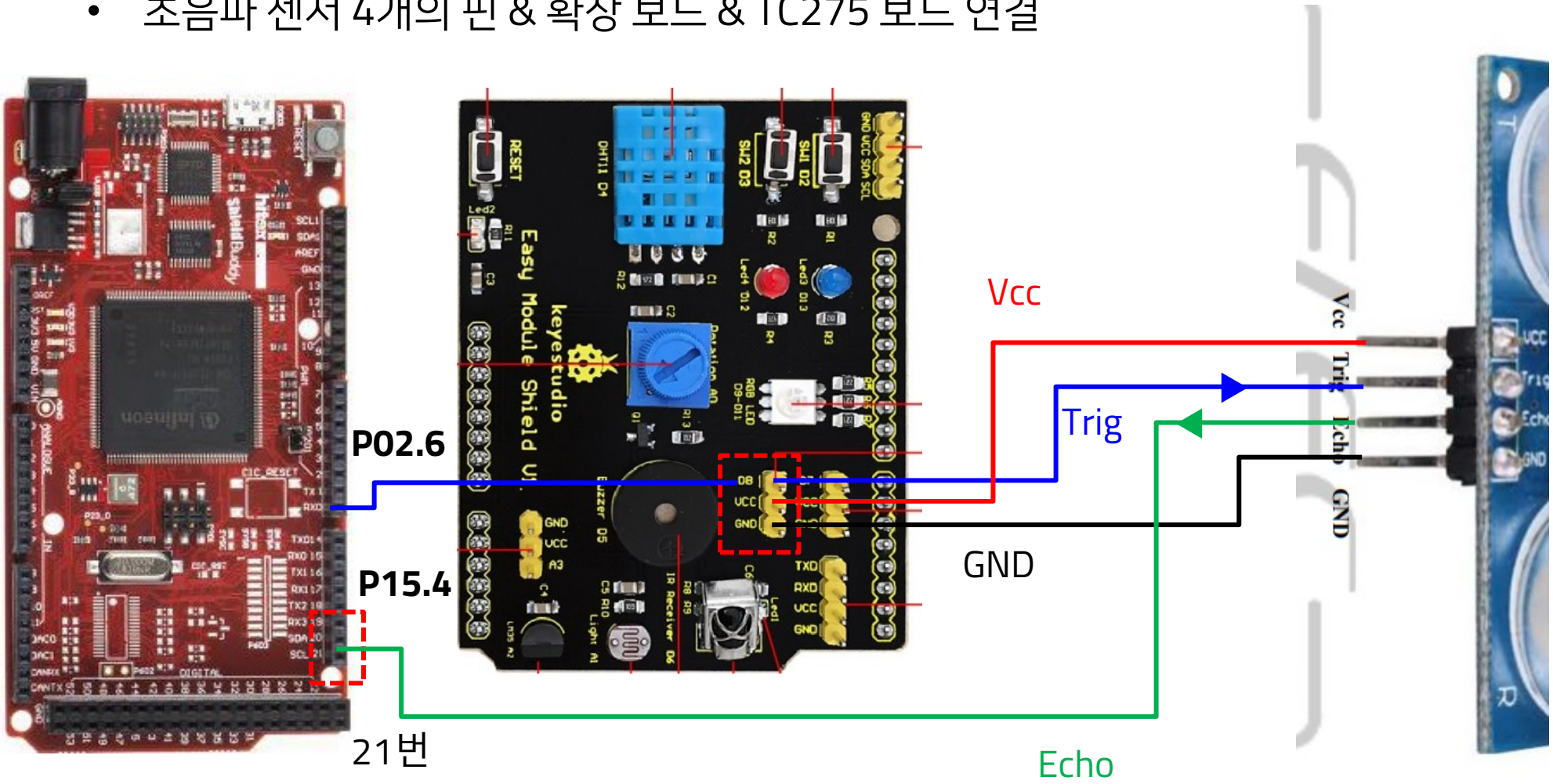
DC모터의 전원/GND



TC275 보드와 초음파 센서 연결

:확장 보드의 핀과 초음파 센서 연결 – 모든 핀 연결한 모습

- 초음파 센서 4개의 핀 & 확장 보드 & TC275 보드 연결



초음파 센서로 측정한 거리 기반 모터 RPM을 가속/감속하도록 PID 제어

Lab1

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

```
26  ****************************************
27  #include "Ifx_Types.h"
28  #include "IfxCpu.h"
29  #include "IfxScuWdt.h"
30
31  #include "IfxCcu6_reg.h"
32  #include "IfxGtm_reg.h"
33
34  #include "Ifx_Console.h"
35  #include <ASCLIN_UART.h>
36
37  // SCU registers
38  #define LCK_BIT_LSB_IDX             1
39  #define ENDINIT_BIT_LSB_IDX         0
40
41  // GTM registers
42  #define DISS_BIT_LSB_IDX            1
43  #define DISR_BIT_LSB_IDX            0
44
45  IfxCpu_syncEvent g_cpuSyncEvent = 0;
46
47  void initLED(void);
48  void initERU(void);
49  void initCCU60(void);
50  void initCCU61(void);
51  void usonicTrigger(void);
52  void initUSonic(void);
53  void initMotor(void);
54  void initGTM(void);
```

```
55
56  // PID Parameters (for acc)
57  float Kp = 0.005;
58  float Ki = 0.01;
59  float Kd = 0.05;
60
61  // Error variables
62  float err = 0;
63  float pre_err = 0;
64
65  // Motor speed from sensor
66  float RPM;  // MAX = 4500 RPM
67  float setpoint = 1000.0;
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
75
76  unsigned int range;
77  unsigned char range_valid_flag = 0;
78
```

PID 제어 구현을 위해 사용되는 전역 변수들

초음파 센서에서 측정한 거리를 저장할 변수

Lab2

: ERU (외부 인터럽트) ISR

```
79
80  __interrupt(0x0A) __vector_table(0)
81  void ERU0_ISR(void)
82  {
83      if( P15_IN.B.P4 != 0x0 )     // rising edge of echo
84      {
85          //                  _________
86          //        echo ______|
87          //                  ^
88          CCU61_TCTR4.B.T12RS = 0x1;       // start CCU61 T12 counter
89      }
90      else    // falling edge of echo
91      {
92          //                  _____
93          //        echo       |_____
94          //                  ^
95          CCU61_TCTR4.B.T12RR = 0x1;       // stop CCU61 T12 counter
96
97          // (1 / t_freq) * counter * 1000000 / 58  =  centimeter
98          range = ((CCU61_T12.B.T12CV * 1000000) / 48828) / 58;
99          range_valid_flag = 1;
100
101         CCU61_TCTR4.B.T12RES = 0x1;       // reset CCU61 T12 counter
102     }
103 }
104
105 __interrupt(0x0E) __vector_table(0)
106 void ERU2_ISR(void)
107 {
108 //    P10_OUT.B.P1 ^= 0x1;
109     rotate++;
110 }
111
```

ERU0_ISR
→ 초음파 센서의 echo 핀을 입력 받음

ERU2_ISR
→ 모터의 엔코더 펄스를 입력 받음 (회전 수 및 RPM 측정에 사용)

Lab3

: CCU60 타이머 인터럽트

CCU60_T12_ISR
→ 초음파 센서 trigger 신호 발생을 위해 10us 길이 펄스 생성

CCU60_T13_ISR
→ 일정 시간 간격으로 PID 제어를 위한 타이머

```
112 __interrupt(0x0B) __vector_table(0)
113 void CCU60_T12_ISR(void)
114 {
115     // end of 10us Trig
116     // GPIO P02.6 --> LOW
117     P02_OUT.B.P6 = 0x0;
118 }
119
120 // 0.1 sec = 100 ms
121 __interrupt(0x0D) __vector_table(0)
122 void CCU60_T13_ISR(void)
123 {
124     P10_OUT.B.P1 ^= 0x1;
125
126     RPM = (float)(rotate*30);
127
128     err = setpoint - RPM;
129
130     _P = Kp * err;
131     _I += Ki * err * 0.25;
132     _D = Kd * ( (err - pre_err) / 0.25 );
133
134     motor_duty += (int)(_P + _I + _D);
135
136     if( motor_duty >= 12500 ) {
137         motor_duty = 12500 - 1;
138     } else if( motor_duty <= 0 ) {
139         motor_duty = 0;
140     }
141     GTM_TOM0_CH3_SR1.B.SR1 = motor_duty;
142
143     Ifx_Console_print("%d %d\n\r", (unsigned int)RPM, motor_duty);
144
145     rotate = 0;
146 }
147
```

Lab4

: main 함수

PWM 출력 시작

RPM, duty를 로그로 확인하기 위한 UART 출력 설정

초음파 센서 거리 측정 시작하는 trigger 함수 호출

측정한 거리에 따라서 setpoint 변경

```
148
149 int core0_main(void)
150 {
151     IfxCpu_enableInterrupts();
152
153     /* !!WATCHDOG0 AND SAFETY WATCHDOG ARE DISABLED HERE!!
154      * Enable the watchdogs and service them periodically if it is required
155      */
156     IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());
157     IfxScuWdt_disableSafetyWatchdog(IfxScuWdt_getSafetyWatchdogPassword());
158
159     /* Wait for CPU sync event */
160     IfxCpu_emitEvent(&g_cpuSyncEvent);
161     IfxCpu_waitEvent(&g_cpuSyncEvent, 1);
162
163     initMotor();
164     initGTM();
165     initERU();
166     initCCU60();
167     initCCU61();
168     initLED();
169     initUSonic();
170
171     GTM_TOM0_TGC0_GLB_CTRL.B.HOST_TRIG = 0x1;
172
173     initUart();
174
175     Ifx_Console_print("RPM Duty\n\r");
176
177     while(1)
178     {
179         for(unsigned int i = 0; i < 10000000; i++);
180         usonicTrigger();
181         while( range_valid_flag == 0 );
182
183         // 2000 = less than max RPM
184         // 125 = max ultrasonic cm
185         setpoint = 2000 * range / 150;
186
187     }
188     return (1);
189 }
190
```

Lab5

: ERU configuration

초음파 센서 echo 입력으로부터 ERU 인터럽트를 발생

모터의 엔코더 펄스로부터 ERU 인터럽트를 발생

```
191
192 void initERU(void)
193 {
194     // Ultrasonic Echo
195     SCU_EICR0.B.EXIS0 = 0x0;     // ERS0 - In00
196     SCU_EICR0.B.FEN0  = 0x1;     // falling edge
197     SCU_EICR0.B.REN0  = 0x1;     // rising edge
198
199     SCU_EICR0.B.EIEN0 = 0x1;
200     SCU_EICR0.B.INP0  = 0x0;
201
202     SCU_IGCR0.B.IGP0  = 0x1;
203
204     // SRC Interrupt setting
205     SRC_SCU_SCU_ERU0.B.SRPN = 0x0A;
206     SRC_SCU_SCU_ERU0.B.TOS = 0x00;
207     SRC_SCU_SCU_ERU0.B.SRE = 0x01;
208
209     // Motor Encoder
210     // ERU2 -> IN22 -> OGU0
211     SCU_EICR1.B.EXIS0 = 0x2;     // ERS2 - Input 2 (P00.4)
212     SCU_EICR1.B.FEN0  = 0x1;     // falling edge
213     SCU_EICR1.B.REN0  = 0x0;
214     SCU_EICR1.B.EIEN0 = 0x1;
215     SCU_EICR1.B.INP0  = 0x1;     // OGU0
216
217     SCU_IGCR0.B.IGP1  = 0x1;
218
219     // SRC Interrupt setting
220     SRC_SCU_SCU_ERU1.B.SRPN = 0x0E;
221     SRC_SCU_SCU_ERU1.B.TOS  = 0x0;
222     SRC_SCU_SCU_ERU1.B.SRE  = 0x1;
223 }
```

Lab6

: CCU60 타이머 configuration

10us 길이의 초음파 센서 trigger 신호 생성을 위한 타이머

PID 제어를 일정 시간 간격으로 수행하기 위한 타이머

```
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 CCU

    // 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

    // T12 configurations
    // f_T12 = f_CCU6 / prescaler
    CCU60_TCTR0.B.T12CLK = 0x2;                    // f_CCU6 = 50 MHz, prescaler = 4
    //CCU60_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;  // f_T12 --> 12.5 MHz
    CCU60_TCTR0.B.CTM = 0x0;                       // T12 auto reset when period match (PM) occur
    CCU60_T12PR.B.T12PV = 125 - 1;                 // PM interrupt freq. = f_T12 / (T12PR + 1)
    CCU60_TCTR4.B.T12STR = 0x1;                    // load T12PR from shadow register
    CCU60_TCTR2.B.T12SSC = 0x1;                    // Single Shot Control
    CCU60_T12.B.T12CV = 0x0;                       // clear T12 counter register
    CCU60_INP.B.INPT12 = 0x0;                      // service request output SR0 selected
    CCU60_IEN.B.ENT12PM = 0x1;                     // enable T12 PM interrupt
    SRC_CCU6_CCU60_SR0.B.SRPN = 0x0B;              // set priority 0x0B
    SRC_CCU6_CCU60_SR0.B.TOS = 0x0;                // CPU0 service T12 PM interrupt
    SRC_CCU6_CCU60_SR0.B.SRE = 0x1;                // SR0 enabled


    // T13 configurations
    CCU60_TCTR0.B.T13CLK = 0x2;     // f_CCU6 = 50 MHz, prescaler = 1024
    CCU60_TCTR0.B.T13PRE = 0x1;     // f_T13 --> 48.828 kHz
    CCU60_T13PR.B.T13PV = 4882 - 1;
    CCU60_TCTR4.B.T13STR = 0x1;
    CCU60_T13.B.T13CV = 0x0;
    CCU60_IEN.B.ENT13PM = 0x1;
    CCU60_INP.B.INPT13 = 0x1;
    SRC_CCU6_CCU60_SR1.B.SRPN = 0x0D;
    SRC_CCU6_CCU60_SR1.B.TOS = 0x0;
    SRC_CCU6_CCU60_SR1.B.SRE = 0x1;
    CCU60_TCTR4.B.T13RS = 0x1;
}
```

Lab7

: CCU60 타이머 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_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.B.SR0 = 12500 - 1;

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

모터 RPM 설정을 위한 PWM 생성하는 GTM

Lab8

: 모터, SW버튼, LED configuration

```
343 void initUSonic(void)
344 {
345     P15_IOCR4.B.PC4 = 0x01;         // set P00.4 general input (pull-down connected) [Echo]
346     P02_IOCR4.B.PC6 = 0x10;         // set P02.6 push-pull general output           [Trig]
347
348     P02_OUT.B.P6 = 0x0;
349 }
350
351 void usonicTrigger(void)
352 {
353     // start of 10us Trigger Pulse
354     // GPIO P02.6 --> HIGH
355     P02_OUT.B.P6 = 0x1;
356     range_valid_flag = 0;
357     CCU60_TCTR4.B.T12RS = 0x1;           // T12 start counting
358 }
359
360 void initMotor(void)
361 {
362     P10_IOCR0.B.PC2 = 0x10;      // Direction (P10.2)
363     P10_IOCR0.B.PC3 = 0x11;      // PWM (P10.3)
364
365     // encoder input
366     P00_IOCR4.B.PC4 = 0x01;
367 }
368
369 void initLED(void)
370 {
371     P10_IOCR0.B.PC1 = 0x10;         // set P10.1 push-pull general output
372 }
373
```

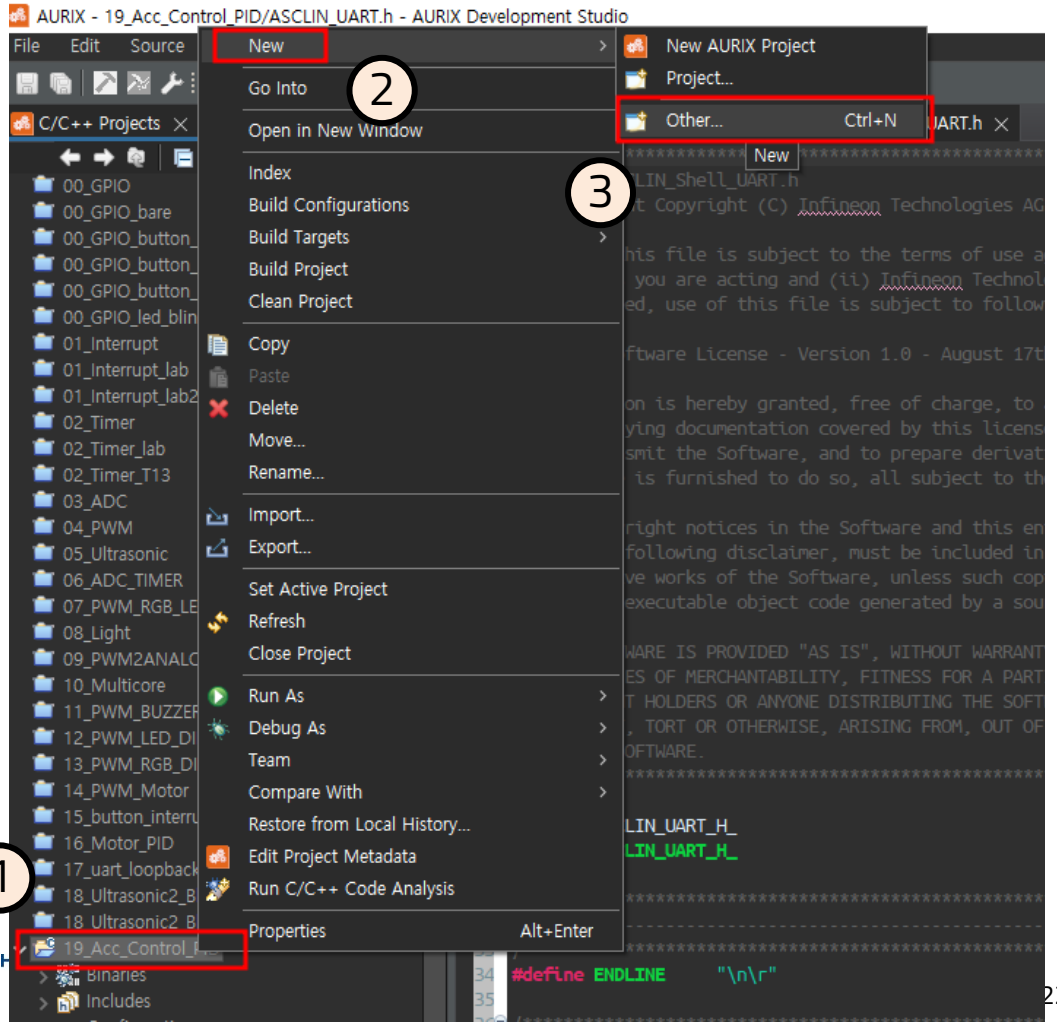
모터, 초음파 센서 등에서
사용하는 핀(포트) 설정

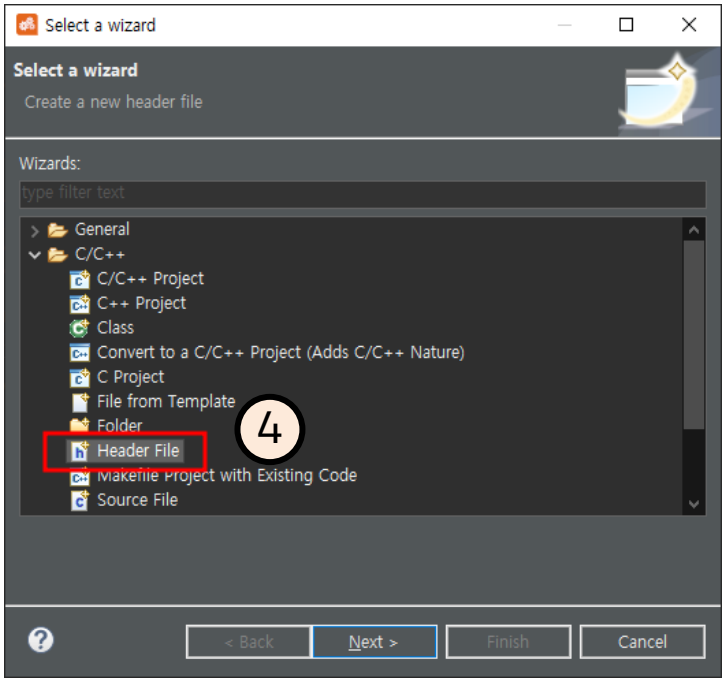
Lab9

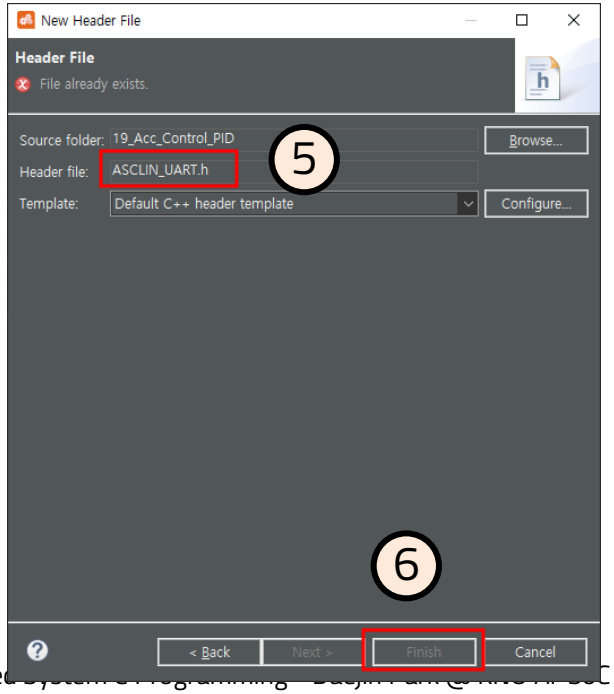
: UART configuration 헤더 파일

새로운 파일 생성

ASCLIN_UART.h

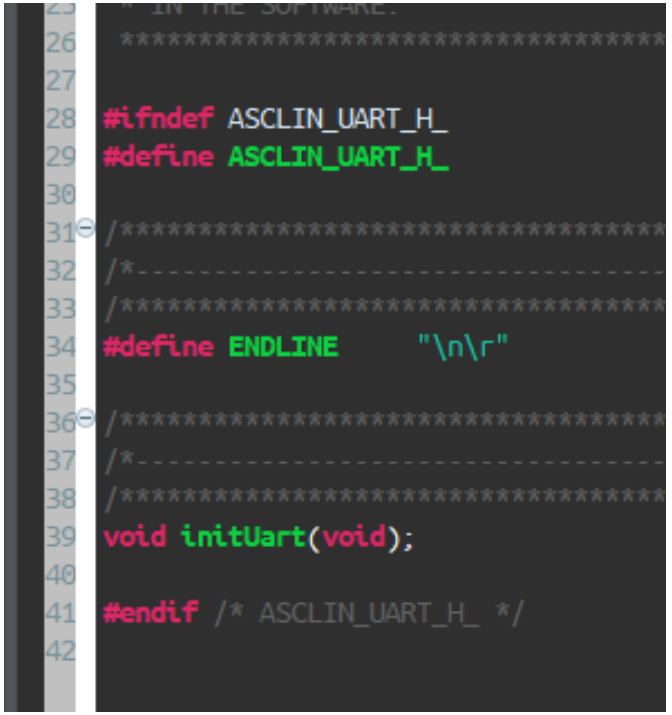






Lab9

: UART configuration 헤더 파일



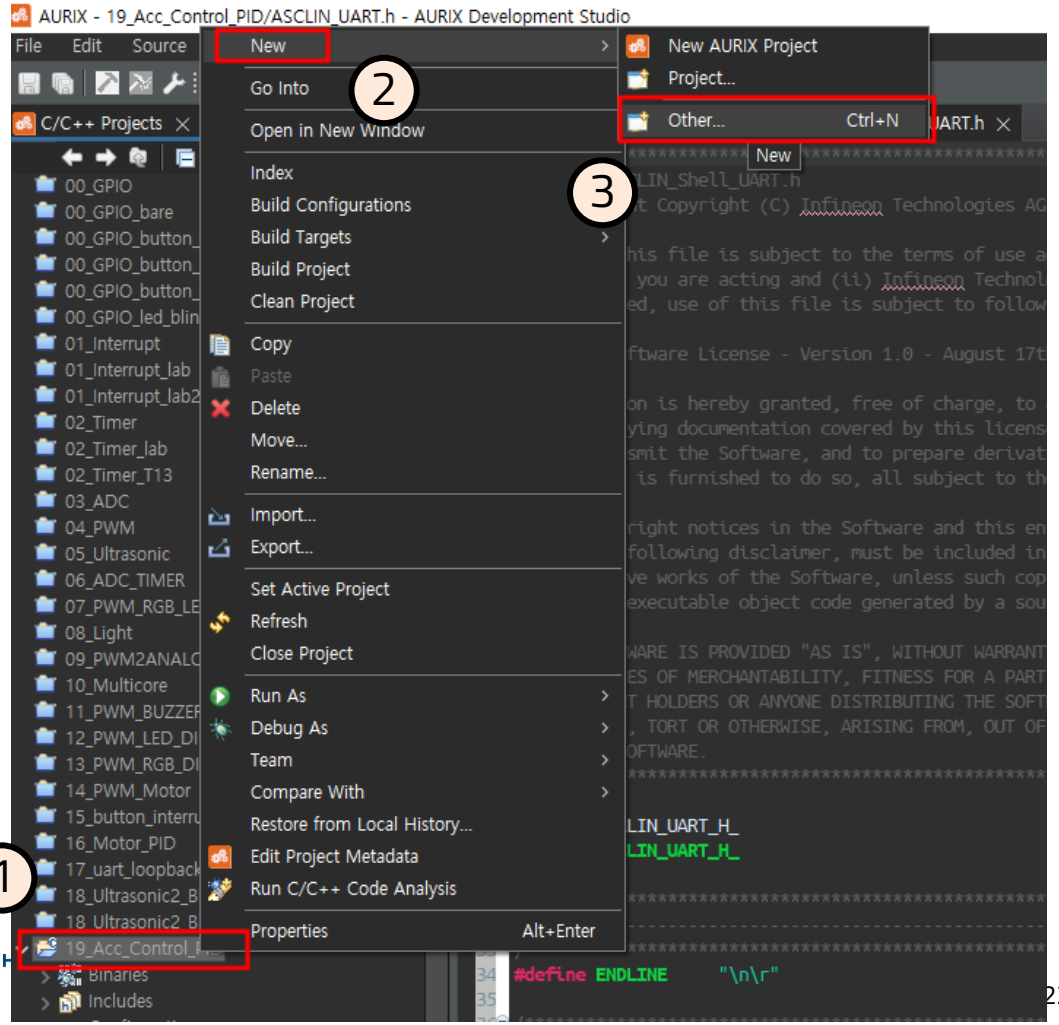
```
 * IN THE SOFTWARE.
 *********************************************

#ifndef ASCLIN_UART_H_
#define ASCLIN_UART_H_

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

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

#endif /* ASCLIN_UART_H_ */

```

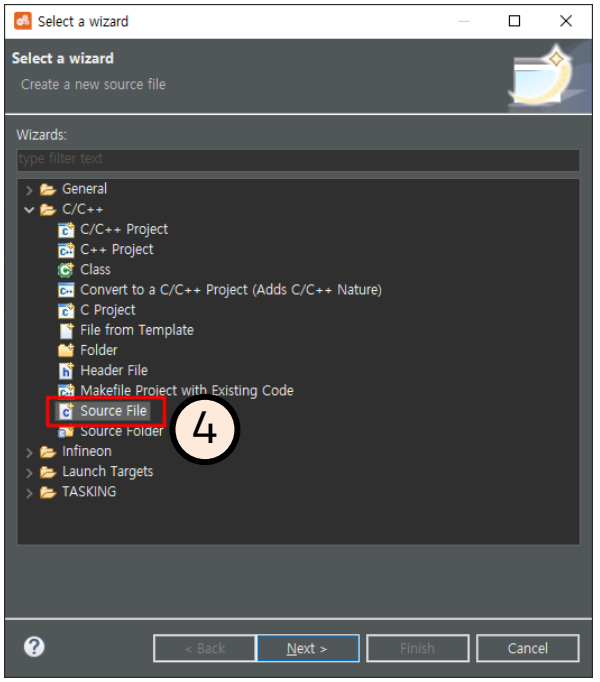
Lab10

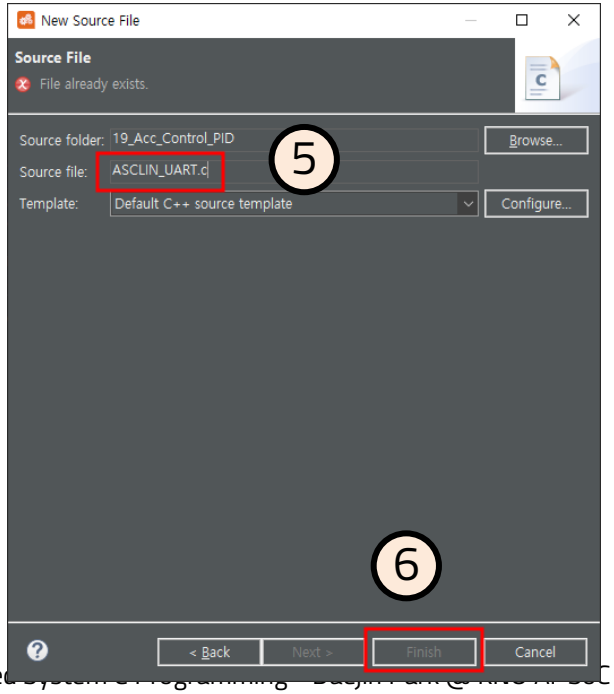
: UART configuration 소스 파일

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);
}
```

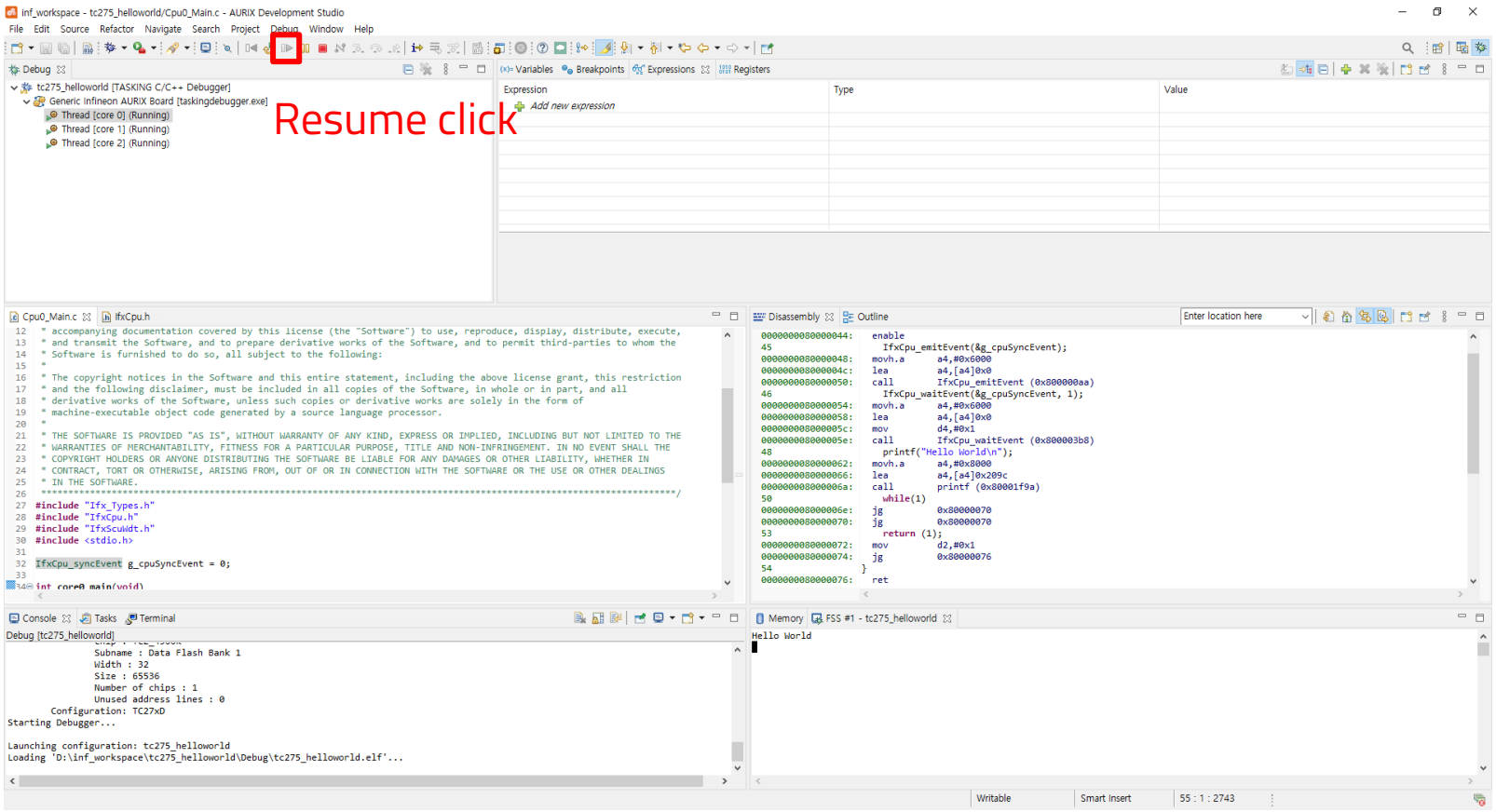
Lab10 (계속)

: 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

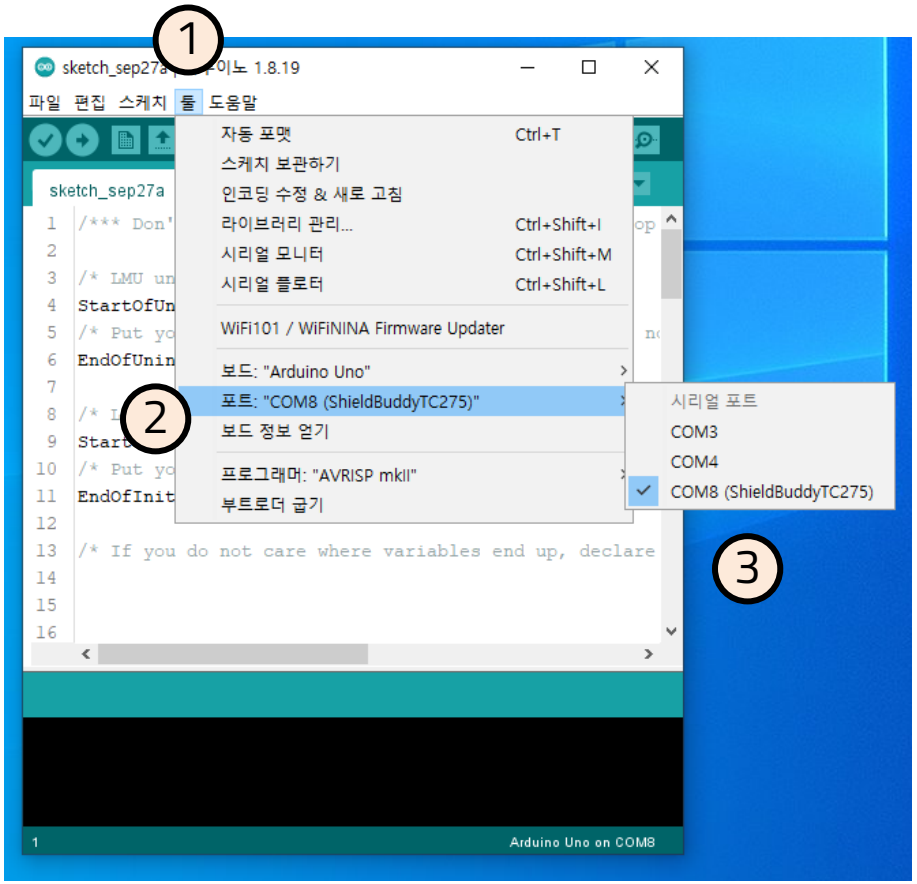
- 프로젝트 빌드 (ctrl + b)
- 디버그 수행하여 보드에 실행 파일 flash



TC275에서 출력되는 로그 확인

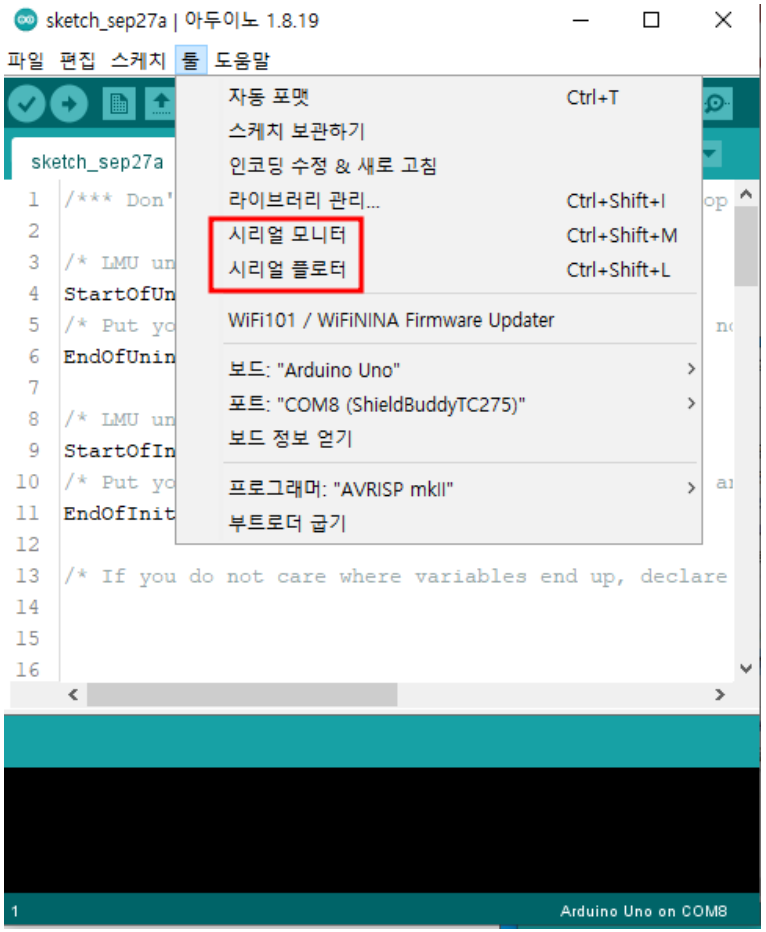
- 아두이노 IDE 설치

- TC275에 해당하는 포트 연결
  - COM* 중에서 TC275 선택



TC275에서 출력되는 로그 확인 (계속)

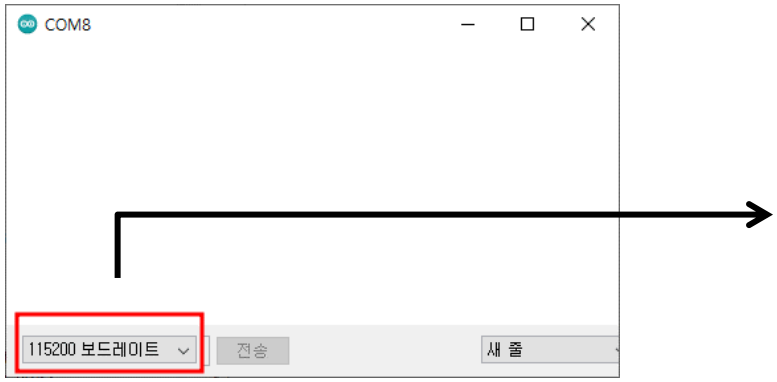
- 문자로 로그를 확인하고 싶으면
  - 시리얼 모니터
- 그래프 형태로 로그를 확인하고 싶으면
  - 시리얼 플로터



TC275에서 출력되는 로그 확인 (계속)

: RPM, Duty 변화 추이 그래프로 확인

- 시리얼 플로터 **open**



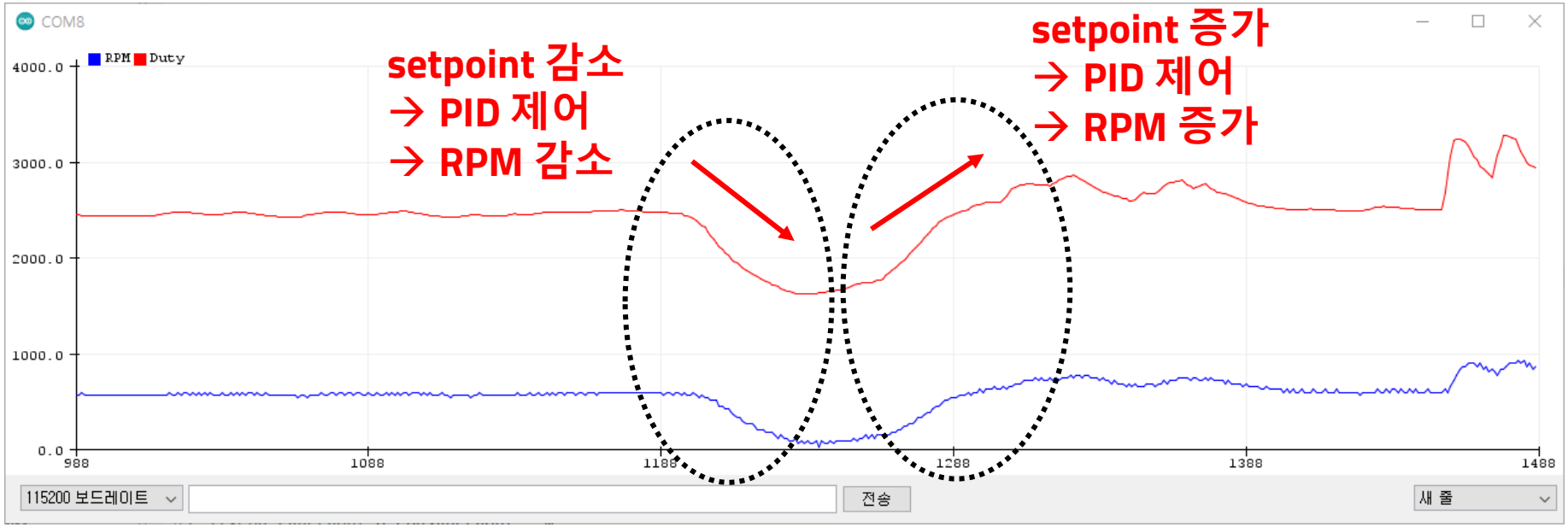
ASCLIN_UART.c 파일에서 정의됨

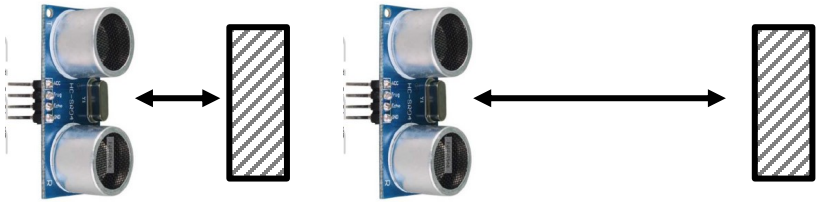
```
37  /*---------------------------------------------
38  /*********************************************
39  /* Communication parameters */
40  #define ISR_PRIORITY_ASCLIN_TX      8
41  #define ISR_PRIORITY_ASCLIN_RX      4
42  #define ISR_PRIORITY_ASCLIN_ER      12
43  #define ASC_TX_BUFFER_SIZE          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





물체와의 거리 감소　　물체와의 거리 증가

감사합니다. 휴식~~