TI DSP, MCU 및 Xilinx Zynq FPGA
프로그래밍 전문가 과정

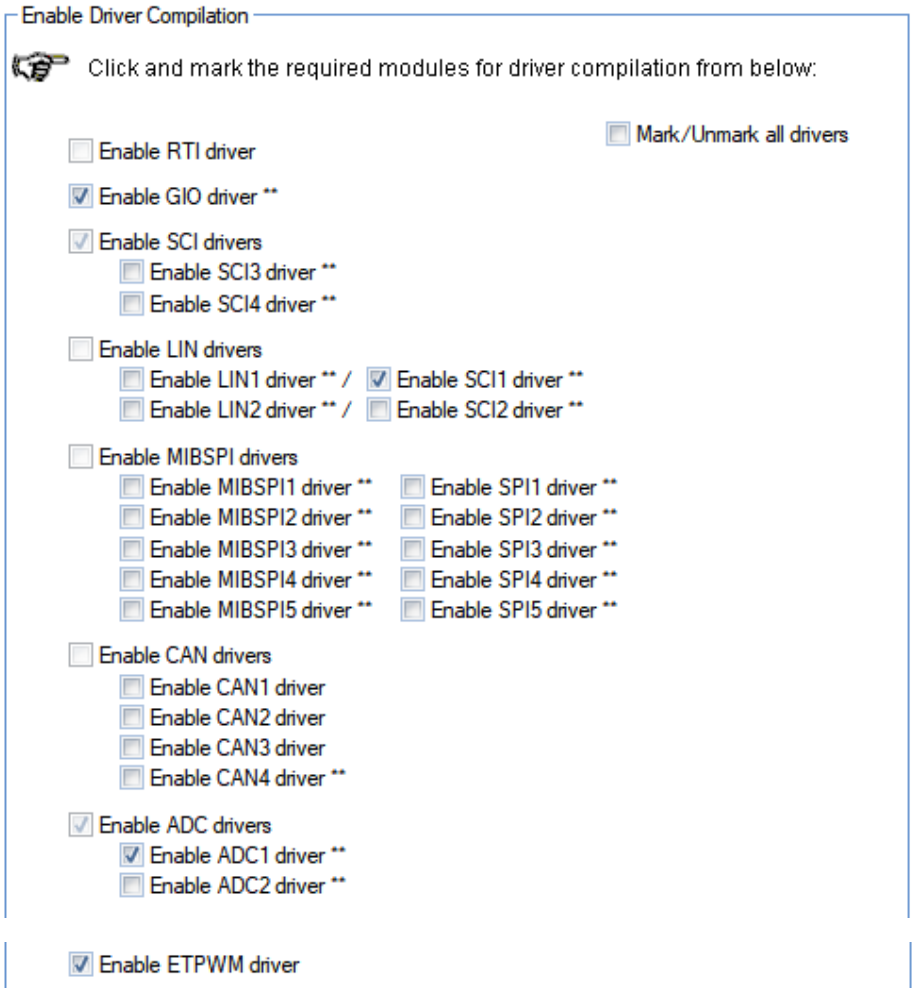
강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com

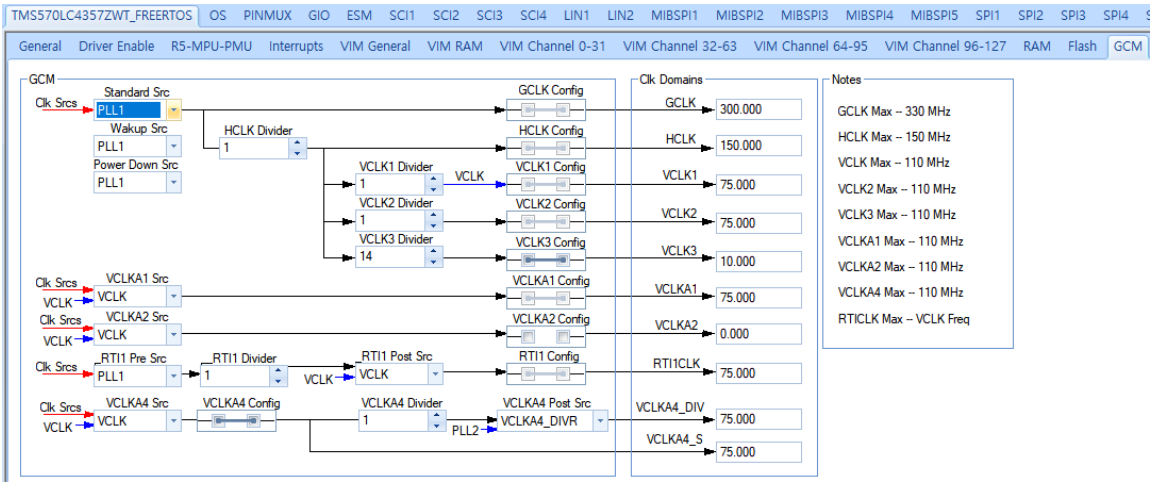
학생 – 문한나
mhn97@naver.com

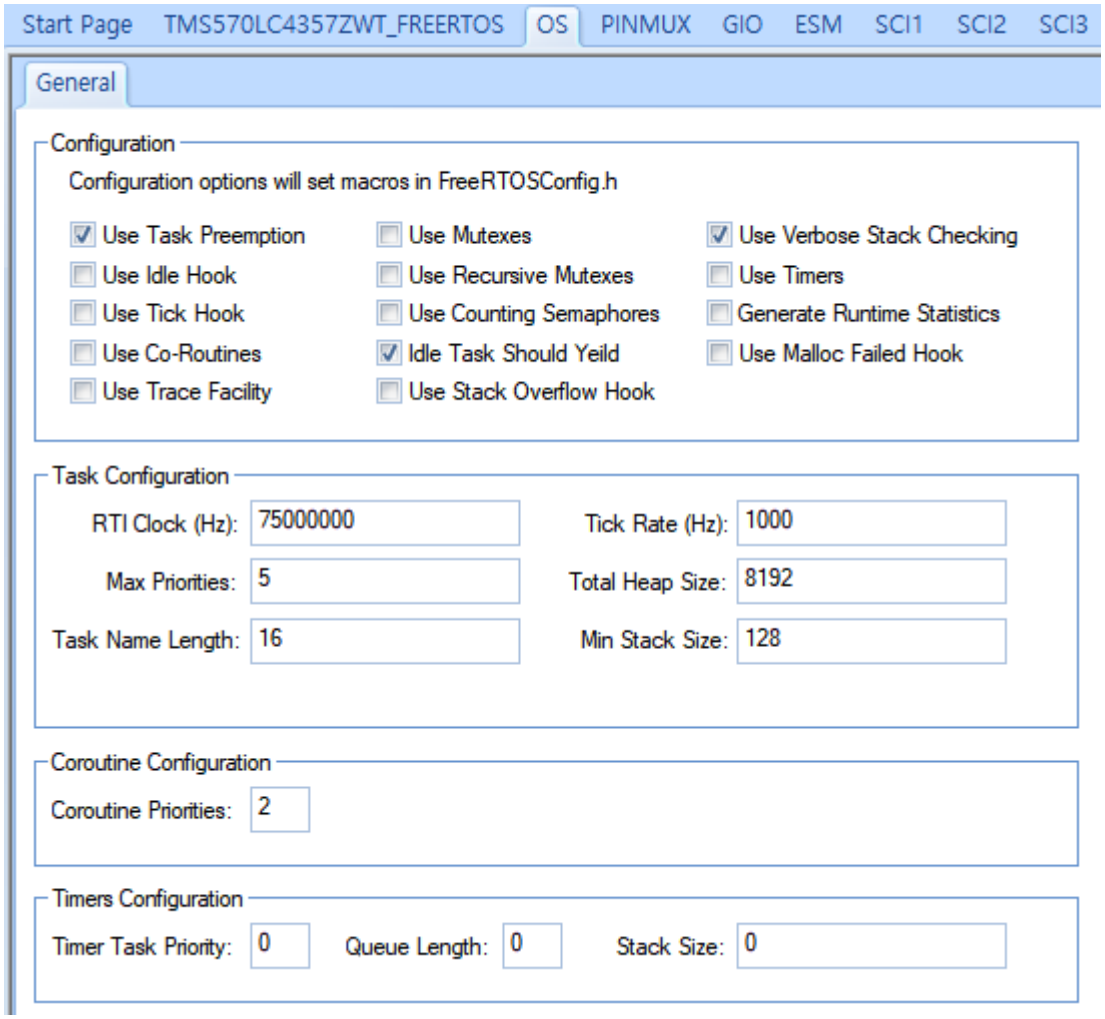
학생 – 장성환
redmk1025@gmail.com

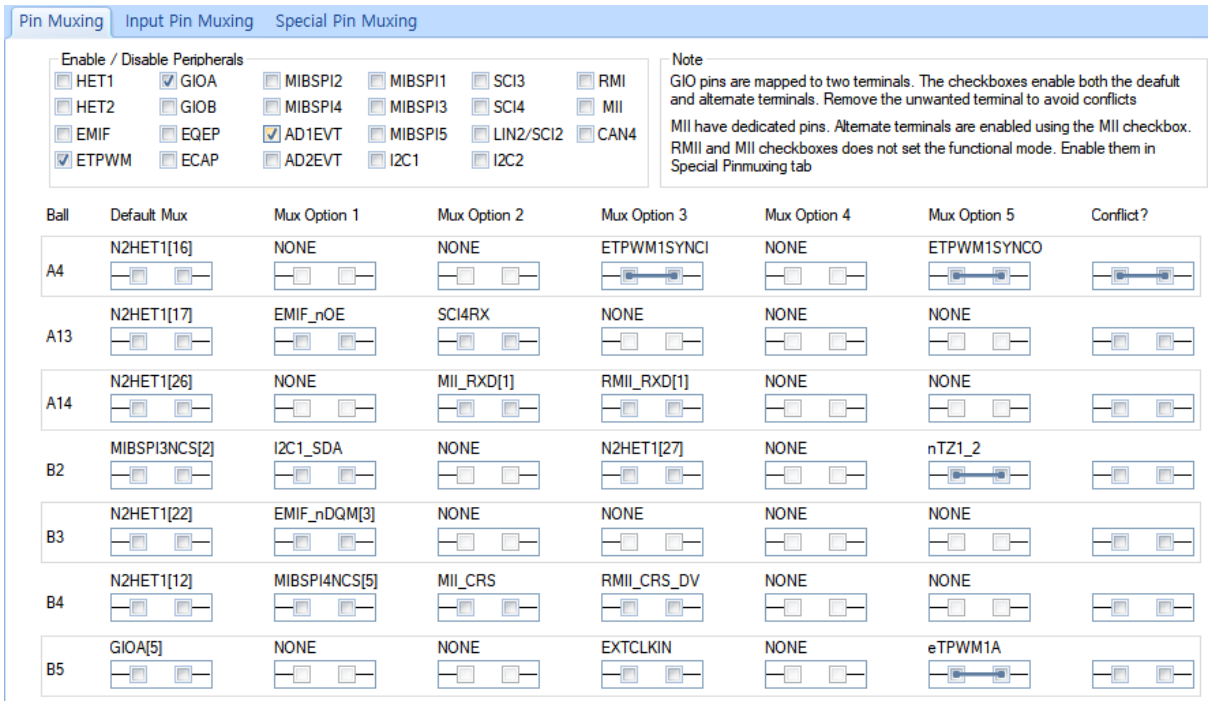
조도센서를 활용한 수동주행

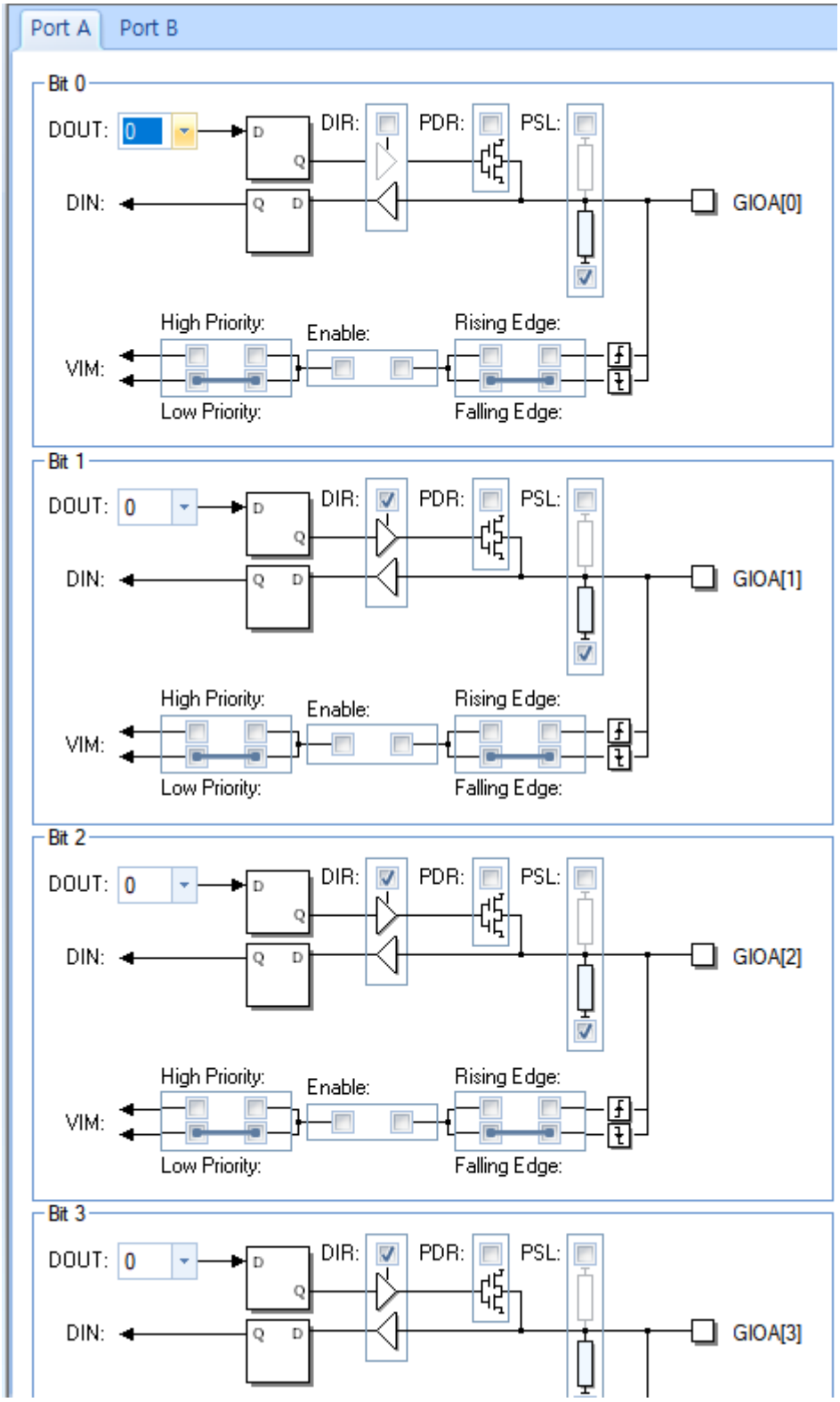
HCG 설정

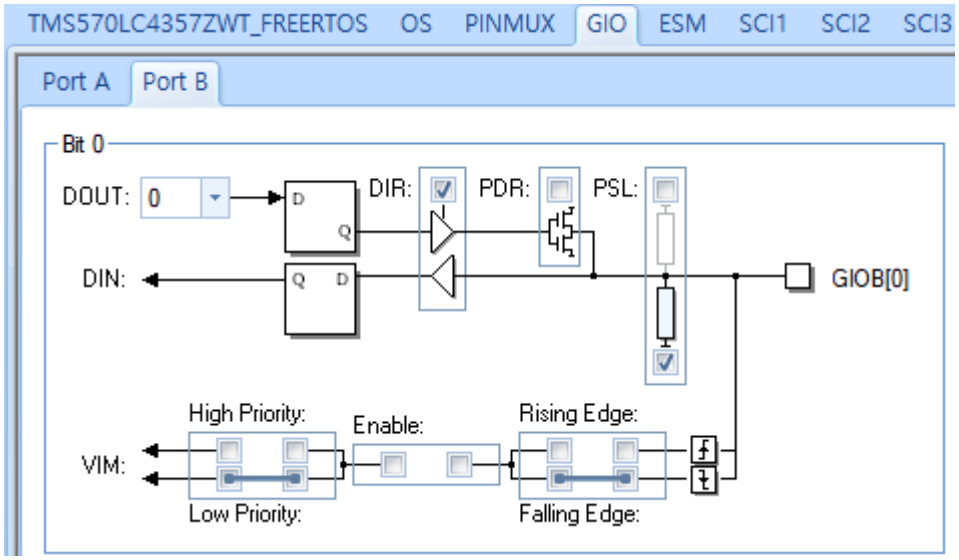


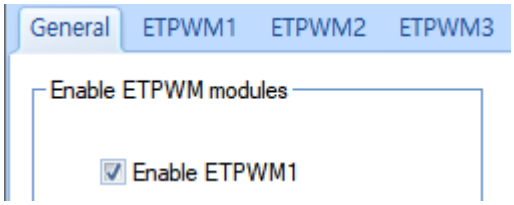


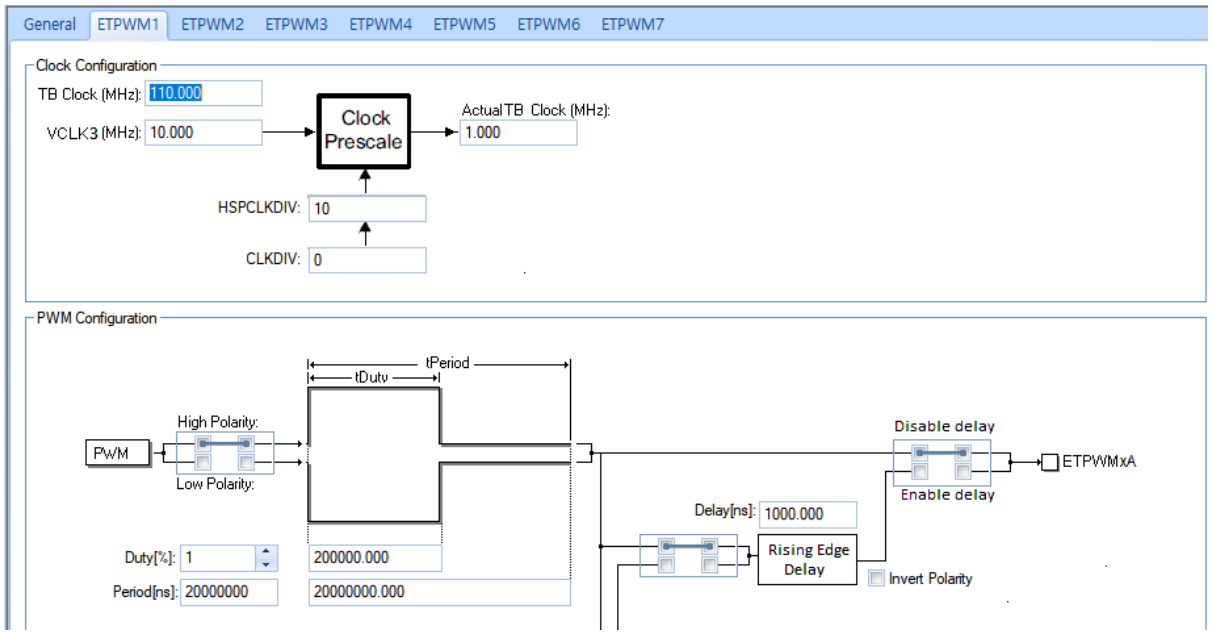

Start Page
TMS570LC4357ZWT_FREERTOS
OS
PINMUX
GIO
ESM
SCI1
SCI2
SCI3
General
Configuration
Configuration options will set macros in FreeRTOSConfig.h
Use Task Preemption
Use Idle Hook
Use Tick Hook
Use Co-Routines
Use Trace Facility
Use Mutexes
Use Recursive Mutexes
Use Counting Semaphores
Idle Task Should Yeild
Use Stack Overflow Hook
Use Verbose Stack Checking
Use Timers
Generate Runtime Statistics
Use Malloc Failed Hook
Task Configuration
RTI Clock (Hz): 75000000
Tick Rate (Hz): 1000
Max Priorities: 5
Total Heap Size: 8192
Task Name Length: 16
Min Stack Size: 128
Coroutine Configuration
Coroutine Priorities: 2
Timers Configuration
Timer Task Priority: 0
Queue Length: 0
Stack Size: 0

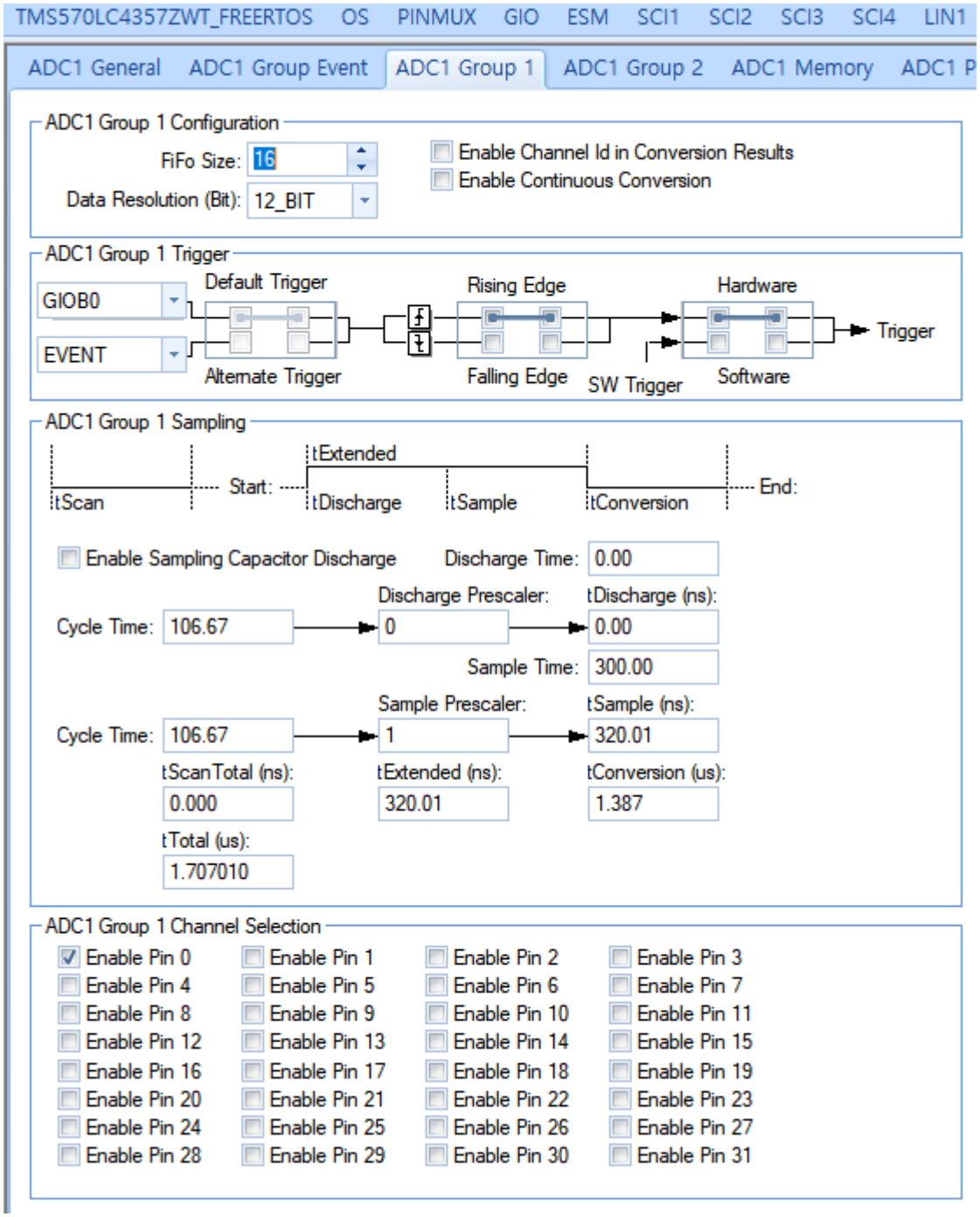

Pin Muxing
Input Pin Muxing
Special Pin Muxing
Enable / Disable Peripherals
HET1
HET2
EMIF
ETPWM
GIOA
GIOB
EQEP
ECAP
MIBSPI2
MIBSPI4
AD1EVT
AD2EVT
MIBSPI1
MIBSPI3
MIBSPI5
I2C1
SCI3
SCI4
LIN2/SCI2
I2C2
RMI
MII
CAN4
Note
GIO pins are mapped to two terminals. The checkboxes enable both the deafult and alternate terminals. Remove the unwanted terminal to avoid conflicts
MII have dedicated pins. Alternate terminals are enabled using the MII checkbox.
RMII and MII checkboxes does not set the functional mode. Enable them in Special Pinmuxing tab
Ball
Default Mux
Mux Option 1
Mux Option 2
Mux Option 3
Mux Option 4
Mux Option 5
Conflict?
A4 N2HET1[16] NONE NONE ETPWM1SYNCI NONE ETPWM1SYNCO
A13 N2HET1[17] EMIF_nOE SCI4RX NONE NONE NONE
A14 N2HET1[26] NONE MII_RXD[1] RMII_RXD[1] NONE NONE
B2 MIBSPI3NCS[2] I2C1_SDA NONE N2HET1[27] NONE nTZ1_2
B3 N2HET1[22] EMIF_nDQM[3] NONE NONE NONE NONE
B4 N2HET1[12] MIBSPI4NCS[5] MII_CRS RMII_CRS_DV NONE NONE
B5 GIOA[5] NONE NONE EXTCLKIN NONE eTPWM1A


Port A
Port B
Bit 0
DOUT:
0
DIR:
PDR:
PSL:
DIN:
GIOA[0]
High Priority:
Enable:
Rising Edge:
VIM:
Low Priority:
Falling Edge:
Bit 1
DOUT:
0
DIR:
PDR:
PSL:
DIN:
GIOA[1]
High Priority:
Enable:
Rising Edge:
VIM:
Low Priority:
Falling Edge:
Bit 2
DOUT:
0
DIR:
PDR:
PSL:
DIN:
GIOA[2]
High Priority:
Enable:
Rising Edge:
VIM:
Low Priority:
Falling Edge:
Bit 3
DOUT:
0
DIR:
PDR:
PSL:
DIN:
GIOA[3]


TMS570LC4357ZWT_FREERTOS
OS
PINMUX
GIO
ESM
SCI1
SCI2
SCI3
Port A
Port B
Bit 0
DOUT:
0
DIR:
PDR:
PSL:
DIN:
GIOB[0]
High Priority:
Enable:
Rising Edge:
VIM:
Low Priority:
Falling Edge:


General
ETPWM1
ETPWM2
ETPWM3
Enable ETPWM modules
Enable ETPWM1


General
ETPWM1
ETPWM2
ETPWM3
ETPWM4
ETPWM5
ETPWM6
ETPWM7
Clock Configuration
TB Clock (MHz):
110.000
VCLK3 (MHz):
10.000
Clock Prescale
Actual TB Clock (MHz):
1.000
HSPCLKDIV:
10
CLKDIV:
0
PWM Configuration
tPeriod
tDuty
High Polarity:
PWM
Low Polarity:
Disable delay
ETPWMxA
Enable delay
Delay[ns]:
1000.000
Rising Edge Delay
Invert Polarity
Duty[%]:
1
200000.000
Period[ns]:
20000000
20000000.000


TMS570LC4357ZWT_FREERTOS
OS
PINMUX
GIO
ESM
SCI1
SCI2
SCI3
SCI4
LIN1
ADC1 General
ADC1 Group Event
ADC1 Group 1
ADC1 Group 2
ADC1 Memory
ADC1 P
ADC1 Group 1 Configuration
FiFo Size:
16
Data Resolution (Bit):
12_BIT
Enable Channel Id in Conversion Results
Enable Continuous Conversion
ADC1 Group 1 Trigger
GIOB0
EVENT
Default Trigger
Alternate Trigger
Rising Edge
Falling Edge
Hardware
SW Trigger
Software
Trigger
ADC1 Group 1 Sampling
tScan
Start:
tExtended
tDischarge
tSample
tConversion
End:
Enable Sampling Capacitor Discharge
Discharge Time:
0.00
Discharge Prescaler:
tDischarge (ns):
Cycle Time:
106.67
0
0.00
Sample Time:
300.00
Sample Prescaler:
tSample (ns):
Cycle Time:
106.67
1
320.01
tScanTotal (ns):
tExtended (ns):
tConversion (us):
0.000
320.01
1.387
tTotal (us):
1.707010
ADC1 Group 1 Channel Selection
Enable Pin 0
Enable Pin 1
Enable Pin 2
Enable Pin 3
Enable Pin 4
Enable Pin 5
Enable Pin 6
Enable Pin 7
Enable Pin 8
Enable Pin 9
Enable Pin 10
Enable Pin 11
Enable Pin 12
Enable Pin 13
Enable Pin 14
Enable Pin 15
Enable Pin 16
Enable Pin 17
Enable Pin 18
Enable Pin 19
Enable Pin 20
Enable Pin 21
Enable Pin 22
Enable Pin 23
Enable Pin 24
Enable Pin 25
Enable Pin 26
Enable Pin 27
Enable Pin 28
Enable Pin 29
Enable Pin 30
Enable Pin 31

CCS 코드

```
/* Include Files */

#include <include/FreeRTOS.h>
#include <include/FreeRTOSConfig.h>
#include <include/HL_adc.h>
#include <include/HL_etpwm.h>
#include <include/HL_gio.h>
#include <include/HL_hal_stdtypes.h>
#include <include/HL_reg_etpwm.h>
#include <include/HL_reg_gio.h>
#include <include/HL_reg_sci.h>
#include <include/HL_sci.h>
#include <include/os_mpu_wrappers.h>
#include <include/os_portmacro.h>
#include <include/os_projdefs.h>
#include <include/os_queue.h>
#include <include/os_semphr.h>
#include <include/os_task.h>
#include <stdlib.h>
#include <string.h>

xTaskHandle xTask1Handle;
xTaskHandle xTask2Handle;
xTaskHandle xTask3Handle;
xTaskHandle xTask4Handle;
xTaskHandle xTask5Handle;

QueueHandle_t mutex = NULL;

void vTask1(void* pvParameters);
void vTask2(void* pvParameters);
void vTask3(void* pvParameters);
void vTask4(void* pvParameters);
void vTask5(void* pvParameters);

#define MAX 10

uint32 fir[10] = { 0, };
uint32 ave = 0;
uint8 x[32] = { 0, };
char flag = 0;
int i;
adcData_t data;
uint8 msg[32] = { 0, };

void send_data(sciBASE_t *sci, uint8* byte, uint32 length)
{
    int i;
    for (i = 0; i < length; i++)
        sciSendByte(sciREG1, byte[i]);
    sciSendByte(sciREG1, '\r');
    sciSendByte(sciREG1, '\n');
}

void delay(uint32 delay)
{
    int i;
    for (i = 0; i < delay; i++)
        ;
}
```

```c
int main(void)
{
    gioInit();
    sciInit();
    adcInit();
    etpwmInit();

    etpwmStartTBCLK();

    gioSetDirection(gioPORTA, 0xE);
    gioSetPort(gioPORTA, 0xffffffff);
    gioSetBit(gioPORTA, 0, 0);
    gioSetBit(gioPORTA, 1, 1);
    gioSetBit(gioPORTA, 2, 1);
    gioSetBit(gioPORTA, 3, 1);

    gioSetBit(gioPORTB, 0, 0);
    adcStartConversion(adcREG1, adcGROUP1);

    etpwmREG1->CMPA = 1500; // 정지

    vSemaphoreCreateBinary(mutex)

    if (xTaskCreate(vTask1, "Task1", configMINIMAL_STACK_SIZE, NULL, 1,
                    &xTask1Handle) != pdTRUE)
    {
        while (1)
            ;
    }
    if (xTaskCreate(vTask2, "Task2", configMINIMAL_STACK_SIZE, NULL, 1,
                    &xTask2Handle) != pdTRUE)
    {
        while (1)
            ;
    }
    if (xTaskCreate(vTask3, "Task3", configMINIMAL_STACK_SIZE, NULL, 1,
                    &xTask3Handle) != pdTRUE)
    {
        while (1)
            ;
    }
    if (xTaskCreate(vTask4, "Task4", configMINIMAL_STACK_SIZE, NULL, 1,
                    &xTask3Handle) != pdTRUE)
    {
        while (1)
            ;
    }
    if (xTaskCreate(vTask5, "Task5", configMINIMAL_STACK_SIZE, NULL, 1,
                    &xTask3Handle) != pdTRUE)
    {
        while (1)
            ;
    }

    vTaskStartScheduler();

    while (1)
        ;

    return 0;
}
```

```
void vTask1(void *pbParameters)
{
    while (1)
    {
        //delay(10000000);
        if (flag == 1)
        {
            if (xSemaphoreTake(mutex, ( TickType_t ) 10) == pdTRUE)
            {
                for (i = 0; i < 10; i++)
                {
                    gioSetBit(gioPORTB, 0, 1);
                    while (adcIsConversionComplete(adcREG1, adcGROUP1) == 0)
                        ;
                    adcGetData(adcREG1, adcGROUP1, &data);
                    fir[i] = data.value;
                    gioSetBit(gioPORTB, 0, 0);

                }
                //필터 생성
                ave = (fir[0] + fir[1] + fir[2] + fir[3] + fir[4] + fir[5]
                       + fir[6] + fir[7] + fir[8] + fir[9]) / MAX;
                //필터값 출력
                ltoa(ave, x);
                send_data(sciREG1, x, strlen(x));

                gioSetBit(gioPORTB, 0, 1);
                while (adcIsConversionComplete(adcREG1, adcGROUP1) == 0)
                    ;
                adcGetData(adcREG1, adcGROUP1, &data);
                ltoa(data.value, x);
                //현재값 출력
                send_data(sciREG1, x, strlen(x));

                gioSetBit(gioPORTB, 0, 0);

                flag = 5;
                //vTaskDelay(portMAX_DELAY);
                xSemaphoreGive(mutex);
                vTaskDelay(10);
            }
            else
            { //키를 받아오지 못하는 경우에 실행 코드를 작성
              //flag = 1;
                xSemaphoreGive(mutex);
                vTaskDelay(10);
            }
        }
    }
}

void vTask2(void *pbParameters)
{
    while (1)
    {
        if (flag == 2)
        {
            if (xSemaphoreTake(mutex, ( TickType_t ) 10) == pdTRUE)
            {

                //현재값
                gioSetBit(gioPORTB, 0, 1);
```

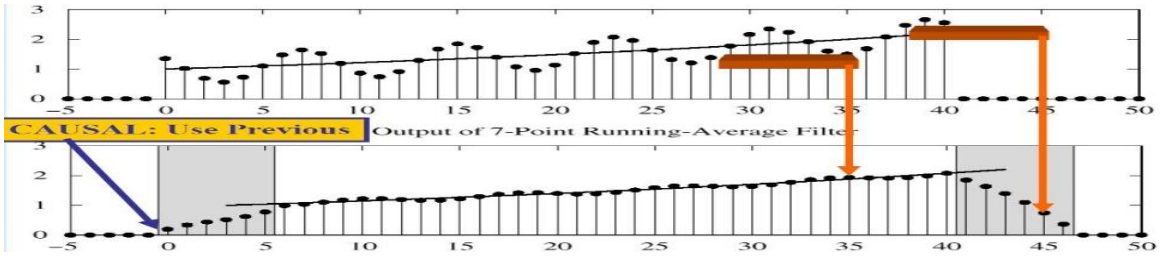
```
                while (adcIsConversionComplete(adcREG1, adcGROUP1) == 0)
                    ;
                adcGetData(adcREG1, adcGROUP1, &data);
                ltoa(data.value, x);
                //현재값 출력
                send_data(sciREG1, x, strlen(x));

                gioSetBit(gioPORTB, 0, 0);

                if ((data.value - ave) > 0 && (data.value - ave) <= 200)
                {

                    flag = 3;

                }
                else if ((ave - data.value) > 200)
                {

                    flag = 4;

                }

                //vTaskDelay(portMAX_DELAY);
                xSemaphoreGive(mutex);
                vTaskDelay(10);

            }
            else
            { //키를 받아오지 못하는 경우에 실행 코드를 작성
              //flag = 1;
                xSemaphoreGive(mutex);
                vTaskDelay(10);
            }
        }
    }
}

void vTask3(void *pbParameters)
{
    while (1)
    {
        if (flag == 3)
        {
            if (xSemaphoreTake(mutex, ( TickType_t ) 10) == pdTRUE)
            {

                etpwmREG1->CMPA -= 1;

                flag = 0;
                xSemaphoreGive(mutex);
                vTaskDelay(10);
            }

            else
            { //키를 받아오지 못하는 경우에 실행 코드를 작성
              //flag = 1;
                xSemaphoreGive(mutex);
                vTaskDelay(10);
            }
        }
        else if (flag == 4)
        {
            if (xSemaphoreTake(mutex, ( TickType_t ) 10) == pdTRUE)
```

```
            {
                etpwmREG1->CMPA -= 2;

                flag = 0;
                xSemaphoreGive(mutex);
                vTaskDelay(10);
            }
            else
            { //키를 받아오지 못하는 경우에 실행 코드를 작성
              //flag = 1;
                xSemaphoreGive(mutex);
                vTaskDelay(10);
            }
        }
    }
}

void vTask4(void *pbParameters)
{
    while (1)
    {
        if (flag == 5)
        {
            if (xSemaphoreTake(mutex, ( TickType_t ) 10) == pdTRUE)
            {

                etpwmREG1->CMPA += 5;
                if ((etpwmREG1->CMPA) >= 1500)
                {

                    etpwmREG1->CMPA = 1500;

                }

                flag = 0;
                xSemaphoreGive(mutex);
                vTaskDelay(10);
            }

            else
            {
                xSemaphoreGive(mutex);
                vTaskDelay(10);
            }
        }
    }

}

void vTask5(void *pbParameters)
{
    while (1)
    {
        if (flag == 0)
        {
            if (xSemaphoreTake(mutex, ( TickType_t ) 10) == pdTRUE)
            {

                if (gioGetBit(gioPORTA, 0) == 0)
                {

                    flag = 1;
                    vTaskDelay(10);
```

```
                }
                else if (gioGetBit(gioPORTA, 0) == 1)
                {

                    flag = 2;
                    vTaskDelay(10);

                }

                xSemaphoreGive(mutex);
                vTaskDelay(10);
            }
            else
            {
                xSemaphoreGive(mutex);
                vTaskDelay(10);
            }
        }
    }
}
```

ADC 의 조도 센서는 주변환경의 영향을 많이 받는다. 밝은 곳으로 가면 기본 ADC 값이 높은 상태가 되어 버리고 어두운 곳으로 가면 기본 ADC 값이 낮은 상태가 된다.

최대한 ACC 페달 주변에 빛의 영향을 받지 않도록 밀폐 시켰지만, 장소에 따라 값의 변화가 커서 어느 ADC 값의 기준점을 가지고 알고리즘을 쓸 수 없었다.

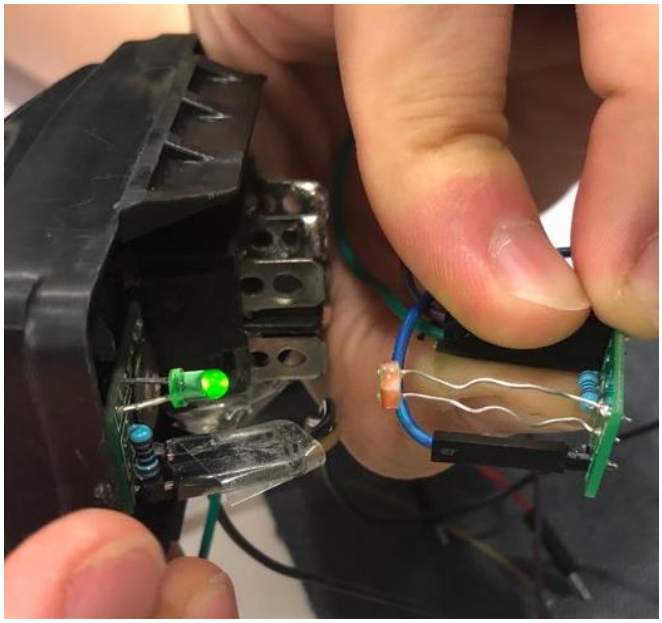


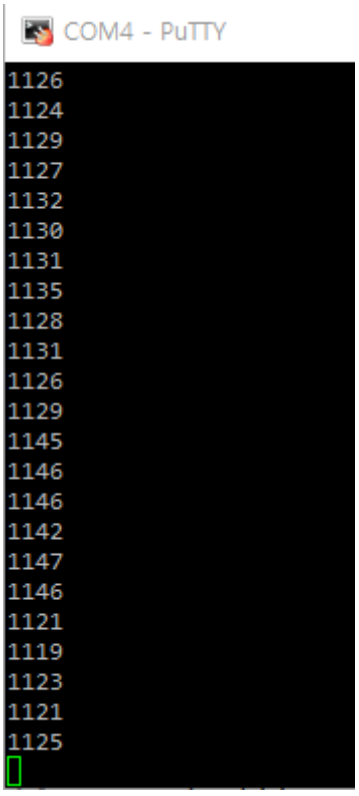
<FIR 필터>

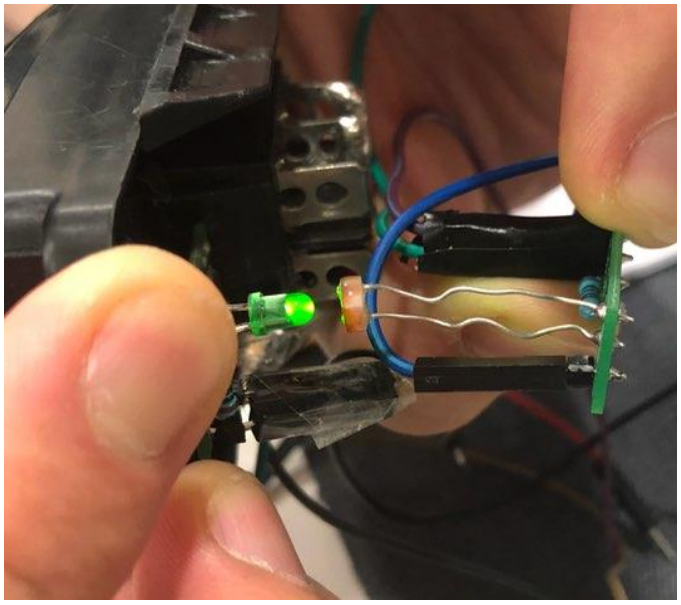
위 그림처럼 노이즈가 있는 상황에서 피드백이 없는 시스템의 경우 FIR 필터를 사용하면 좋은 결과를 얻을 수 있다.

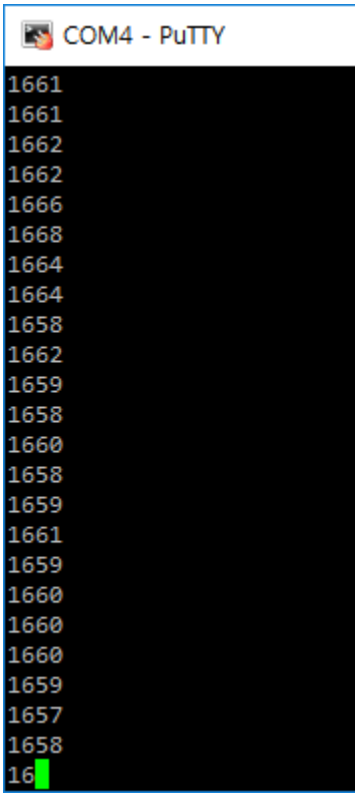
FIR 필터를 사용하여 현재 상태의 ADC 값을 저장해 두고, 현재 상태의 값에서 변동이 생기는 크기에 따라서 주행 가속도가 결정되는 방식을 사용하니 안정적으로 동작하였다.

결과




COM4 - PuTTY
1126
1124
1129
1127
1132
1130
1131
1135
1128
1131
1126
1129
1145
1146
1146
1142
1147
1146
1121
1119
1123
1121
1125




COM4 - PuTTY
1661
1661
1662
1662
1666
1668
1664
1664
1658
1662
1659
1658
1660
1658
1659
1661
1659
1660
1660
1660
1659
1657
1658
16