TI DSP, MCU 및 Xilinx Zynq FPGA
프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)

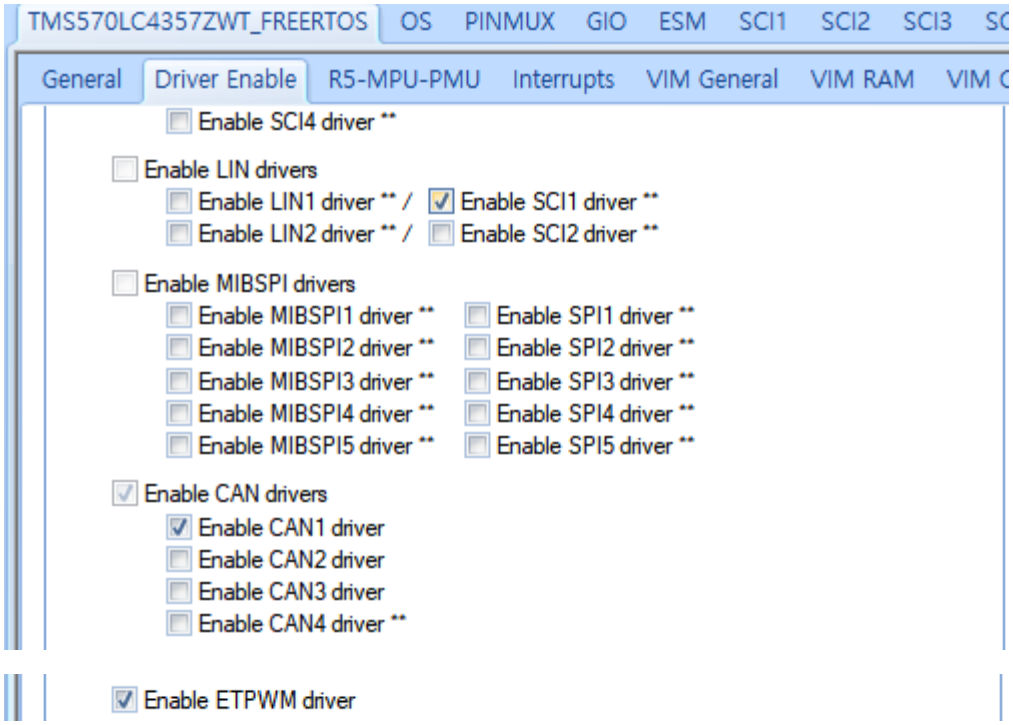
gcccompil3r@gmail.com

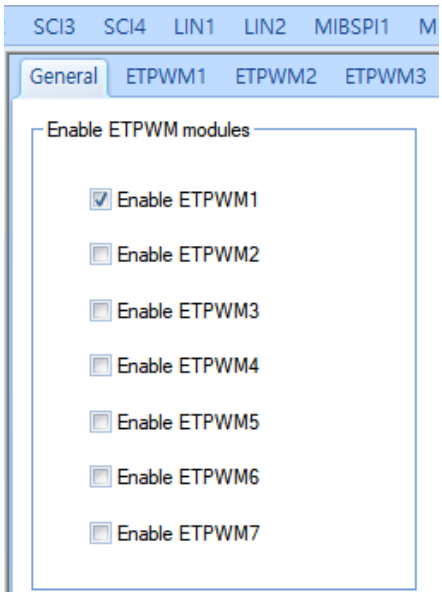
학생 – 문한나

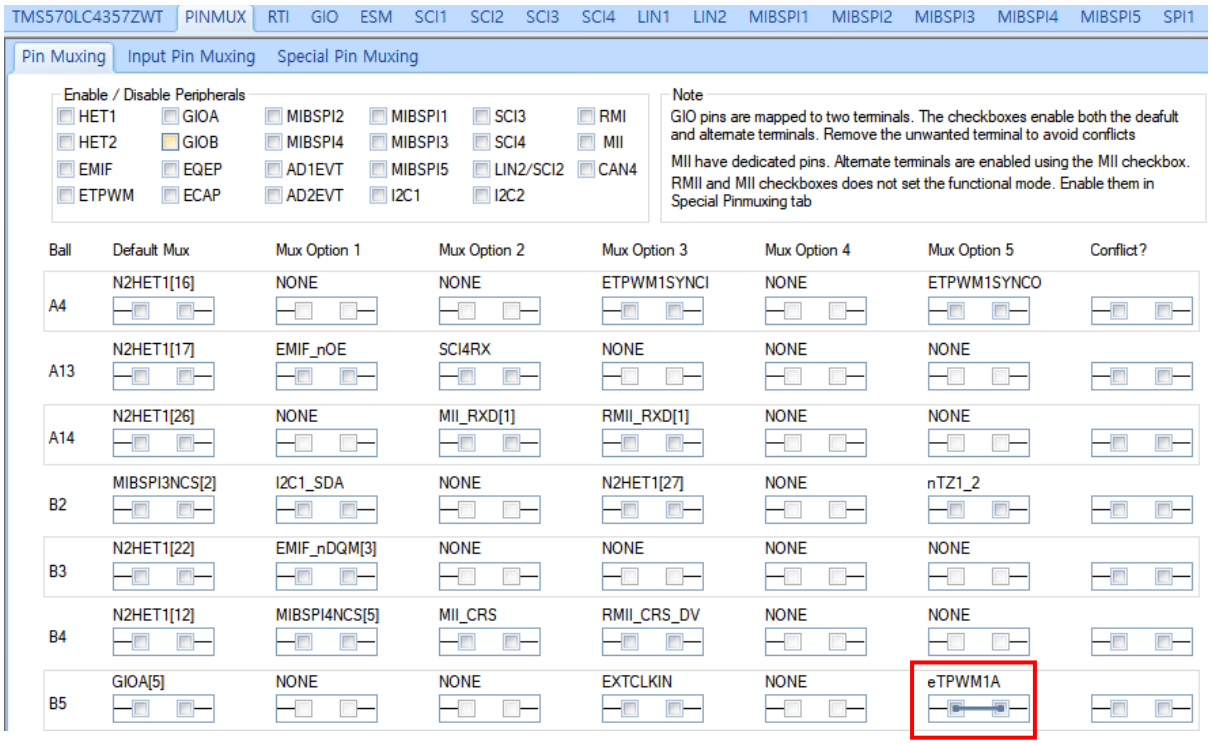
mhn97@naver.com

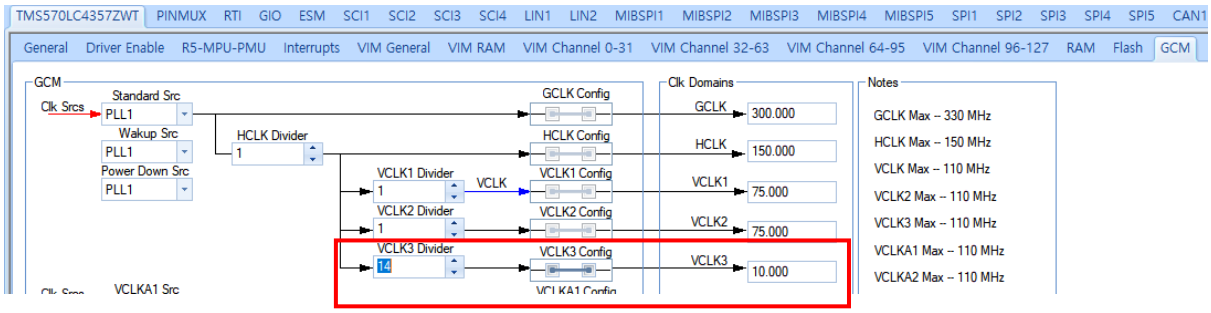
자율 주행 자동차 조향 제어

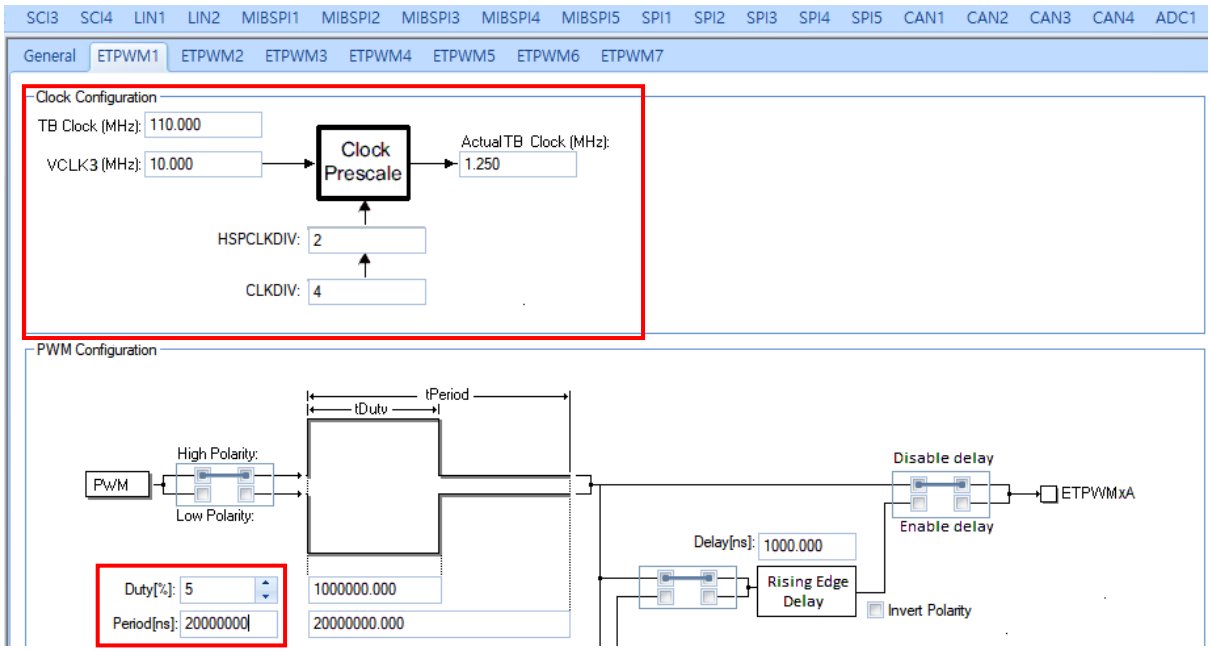
HCG 설정

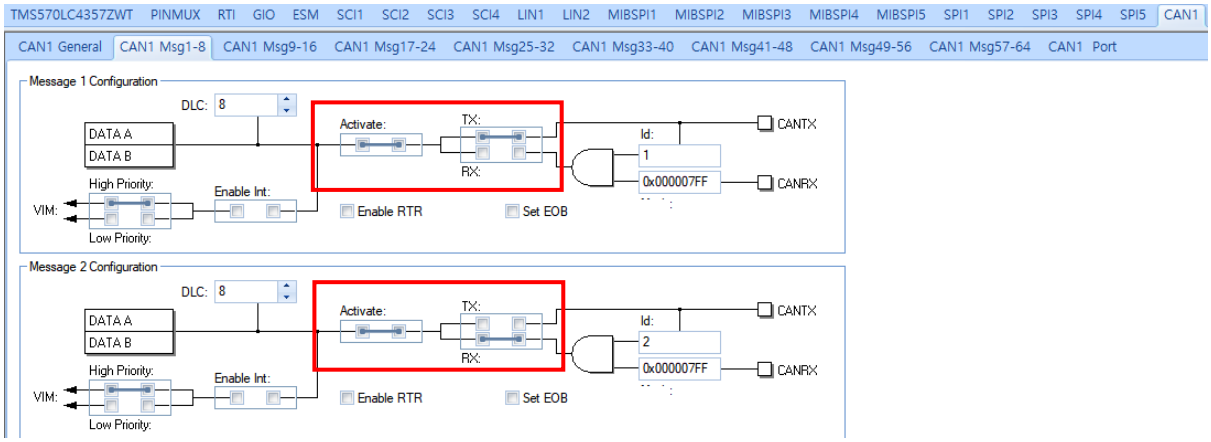





TMS570LC4357ZWT
PINMUX
RTI GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1 LIN2 MIBSPI1 MIBSPI2 MIBSPI3 MIBSPI4 MIBSPI5 SPI1
Pin Muxing
Input Pin Muxing
Special Pin Muxing
Enable / Disable Peripherals
HET1 GIOA MIBSPI2 MIBSPI1 SCI3 RMI
HET2 GIOB MIBSPI4 MIBSPI3 SCI4 MII
EMIF EQEP AD1EVT MIBSPI5 LIN2/SCI2 CAN4
ETPWM ECAP AD2EVT I2C1 I2C2
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


TMS570LC4357ZWT
PINMUX RTI GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1 LIN2 MIBSPI1 MIBSPI2 MIBSPI3 MIBSPI4 MIBSPI5 SPI1 SPI2 SPI3 SPI4 SPI5 CAN1
General Driver Enable R5-MPU-PMU Interrupts VIM General VIM RAM VIM Channel 0-31 VIM Channel 32-63 VIM Channel 64-95 VIM Channel 96-127 RAM Flash GCM
GCM
Clk Srcs
Standard Src
PLL1
Wakup Src
PLL1
Power Down Src
PLL1
HCLK Divider
1
VCLK1 Divider
1
VCLK
VCLK2 Divider
1
VCLK3 Divider
14
GCLK Config
HCLK Config
VCLK1 Config
VCLK2 Config
VCLK3 Config
VCLKA1 Src
Clk Domains
GCLK 300.000
HCLK 150.000
VCLK1 75.000
VCLK2 75.000
VCLK3 10.000
Notes
GCLK Max -- 330 MHz
HCLK Max -- 150 MHz
VCLK Max -- 110 MHz
VCLK2 Max -- 110 MHz
VCLK3 Max -- 110 MHz
VCLKA1 Max -- 110 MHz
VCLKA2 Max -- 110 MHz


SCI3 SCI4 LIN1 LIN2 MIBSPI1 MIBSPI2 MIBSPI3 MIBSPI4 MIBSPI5 SPI1 SPI2 SPI3 SPI4 SPI5 CAN1 CAN2 CAN3 CAN4 ADC1
General ETPWM1 ETPWM2 ETPWM3 ETPWM4 ETPWM5 ETPWM6 ETPWM7
Clock Configuration
TB Clock (MHz): 110.000
VCLK3 (MHz): 10.000
Clock Prescale
ActualTB Clock (MHz): 1.250
HSPCLKDIV: 2
CLKDIV: 4
PWM Configuration
tPeriod
tDuty
High Polarity:
PWM
Low Polarity:
Disable delay
ETPWMxA
Enable delay
Delay[ns]: 1000.000
Rising Edge Delay
Invert Polarity
Duty[%]: 5
Period[ns]: 20000000
1000000.000
20000000.000


TMS570LC4357ZWT PINMUX RTI GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1 LIN2 MIBSPI1 MIBSPI2 MIBSPI3 MIBSPI4 MIBSPI5 SPI1 SPI2 SPI3 SPI4 SPI5 CAN1
CAN1 General CAN1 Msg1-8 CAN1 Msg9-16 CAN1 Msg17-24 CAN1 Msg25-32 CAN1 Msg33-40 CAN1 Msg41-48 CAN1 Msg49-56 CAN1 Msg57-64 CAN1 Port
Message 1 Configuration
DLC: 8
DATA A
DATA B
Activate:
TX:
RX:
Id: 1
0x000007FF
CANTX
CANRX
High Priority:
VIM:
Low Priority:
Enable Int:
Enable RTR
Set EOB
Message 2 Configuration
DLC: 8
DATA A
DATA B
Activate:
TX:
RX:
Id: 2
0x000007FF
CANTX
CANRX
High Priority:
VIM:
Low Priority:
Enable Int:
Enable RTR
Set EOB

CCS 코드

```
#include <FreeRTOS.h>
#include <FreeRTOSConfig.h>
#include <HL_hal_stdtypes.h>
#include <HL_reg_sci.h>
#include <HL_sci.h>
#include <os_mpu_wrappers.h>
#include <os_projdefs.h>
#include <os_semphr.h>
#include <os_task.h>
#include "string.h"
#include "stdio.h"
#include "stdlib.h"
#include "HL_can.h"
#include "HL_sys_common.h"
#include "HL_system.h"
#include "HL_esm.h"
#include "HL_sys_core.h"
#include "HL_etpwm.h"

xTaskHandle xTask1Handle;
xTaskHandle xTask2Handle;
xTaskHandle xTask3Handle;

QueueHandle_t mutex = NULL;

long check = 0;
char flag = 0;
uint8 rx_data[8] = { 0 };

void vTask1(void* pvParameters);
void vTask2(void* pvParameters);
void vTask3(void* pvParameters);
void send_data(sciBASE_t* sci, uint8* msg, int length);

void delay(int time)

{
    int i;
    for (i = 0; i < time; i++)
        ;
}

int main(void)
{
    sciInit();
    canInit();
    etpwmInit();

    etpwmStartTBCLK();

    delay(1000000);

    canEnableErrorNotification(canREG1);

    etpwmREG1->CMPA = 1775;

    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)
                ;
    }

    vTaskStartScheduler();

    while(1)
        ;
}
void vTask1(void *pbParameters)
{
    uint8 str[32] = {'s','t','r','a','i','g','h','t','\r','\n'};
    uint8 rig[32] = {'r','i','g','h','t','\r','\n'};
    uint8 lef[32] = {'l','e','f','t','\r','\n'};
    uint8 msg[32] = {'e','r','r','o','r','\r','\n'};

    while(1){

        if(flag == 2){
            if(xSemaphoreTake(mutex, ( TickType_t ) 10) == pdTRUE){

                if(*rx_data == 0){
                    send_data(sciREG1, str, strlen(str));
                }else if(*rx_data == 1){
                    send_data(sciREG1, rig, strlen(rig));
                }else if(*rx_data == 2){
                    send_data(sciREG1, lef, strlen(lef));
                 }else
                     send_data(sciREG1,msg,strlen(msg));

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

            else{
                }
        }
        else{
        }
    }
}

void vTask2(void *pbParameters)
{

    uint8 msg[32] = {'e','r','r','o','r','\r','\n'};

    while(1){

          if(flag == 1){
              if(xSemaphoreTake(mutex, ( TickType_t ) 10) == pdTRUE){

                  if(rx_data[0] == 1){ //우회전
```

```
                    etpwmREG1->CMPA = 1875 + (13.3 * rx_data[1]);
                    sciSendByte(sciREG1, rx_data[1]);

                }else if(rx_data[0] == 2){ //좌회전
                    etpwmREG1->CMPA = 1875 - (13.3 * rx_data[1]);
                    sciSendByte(sciREG1, rx_data[1]);
                   // send_data(sciREG1, rx_data[1], strlen(rx_data[1]));
                }else if(rx_data[0] == 0){ //직진
                    etpwmREG1->CMPA = 1875;
                }else if(rx_data[0] == 3){
                    etpwmREG1->CMPA = 675; //5a(90) 1875 - 1200 90도 움직임
                }else if(rx_data[0] == 4){
                    etpwmREG1->CMPA = 3075; //5a(90) 1875 + 1200 90도 움직임
                }else
                    send_data(sciREG1,msg,strlen(msg));

                xSemaphoreGive(mutex);
                flag = 2;
                vTaskDelay(10);
                }

            else{
                }
        }
        else{
        }
    }
}

void vTask3(void *pbParameters)
{
    while(1){

        if(flag == 0){ //실행 순서를 정해주는 flag

            if(xSemaphoreTake(mutex, ( TickType_t ) 10) == pdTRUE){
                if (canIsRxMessageArrived(canREG1, canMESSAGE_BOX2)){
                            canGetData(canREG1, canMESSAGE_BOX2, rx_data);
                            //send_data(sciREG1, rx_data, strlen(rx_data));
                            //sciSendByte(sciREG1,'\r');
                            //sciSendByte(sciREG1,'\n');
                            xSemaphoreGive(mutex);
                            flag = 1;
                            vTaskDelay(10);

                }
            }
            else{
                //flag = 1;
                xSemaphoreGive(mutex);
                vTaskDelay(10);
                }
        }
        else{
        }
    }
}
```

```
void send_data(sciBASE_t* sci, uint8* msg, int length)
{
    int i;
    for(i=0;i<length;i++)
        sciSendByte(sci,msg[i]);
}
```

코드는 RTOS상에서 돌아가도록 작성하였다.

프로토콜은 임의로 첫 번째 바이트를 직진, 우회전, 좌회전으로 정했다.

(이후에 정해진 프로토콜을 기준으로 수정 할 예정이다.)

PWM 신호의 TBPRD의 값은 25000으로 정하였다. 일반적으로 서보를 제어 할 PWM신호는 20ms 주기에 5% ~ 10%의 듀티비를 가져야 한다. 5%의 듀티비를 가지는 CMPA의 값은 1250이고,

10%의 듀티비는 2500 이다.

하지만, 실험을 통하여 실제 조향 각을 제어할 수 있도록 시도하는 것이 좋다고 판단하였다.

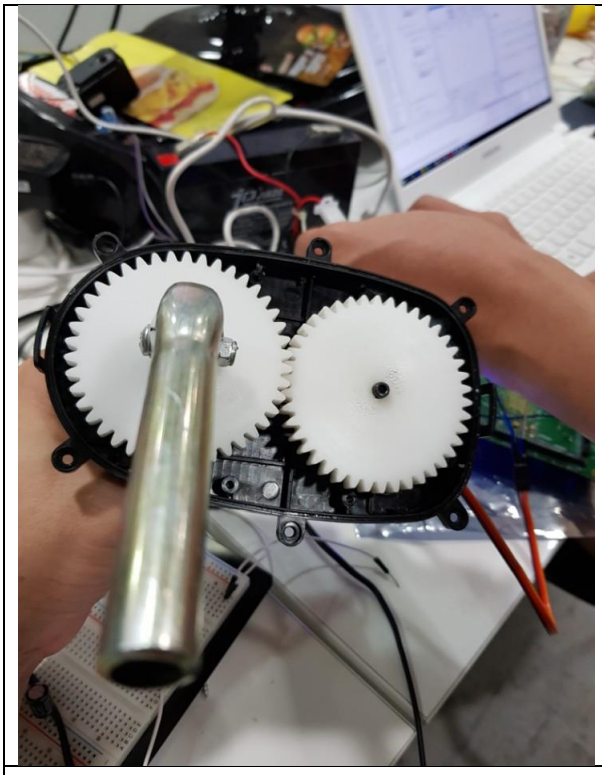
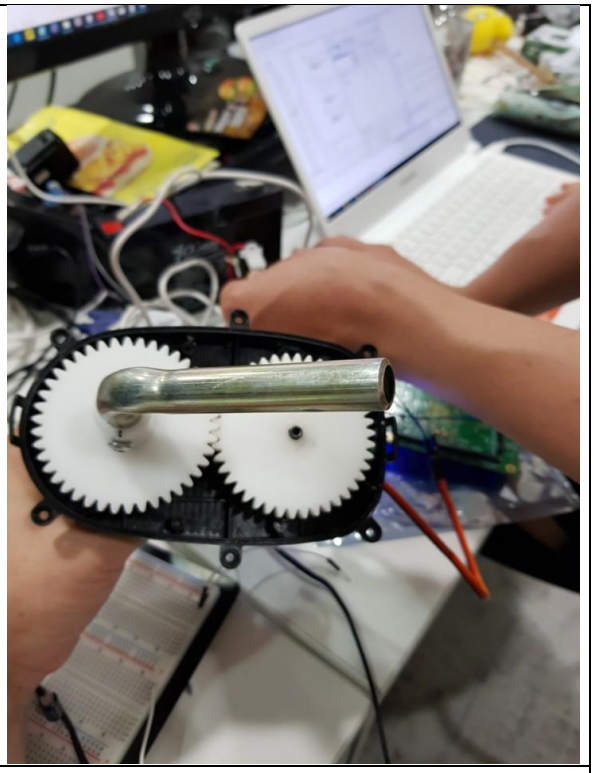
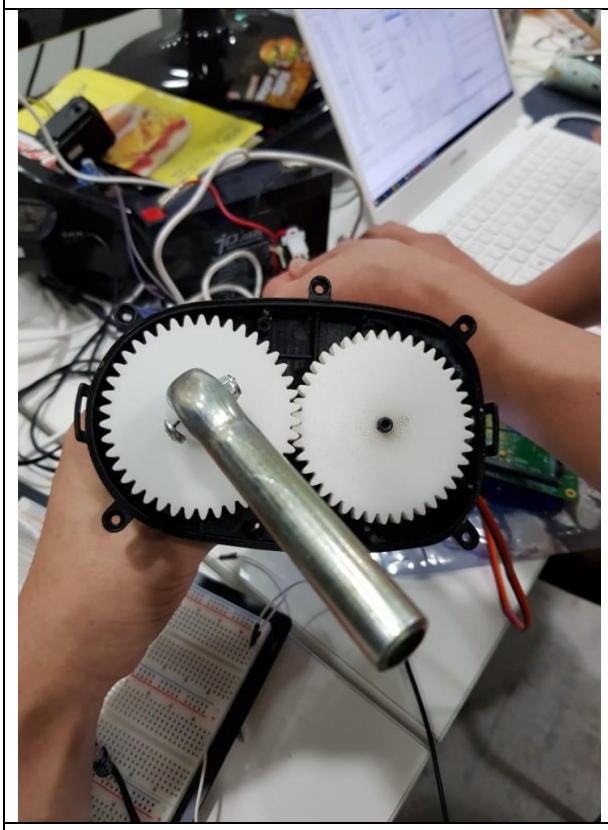
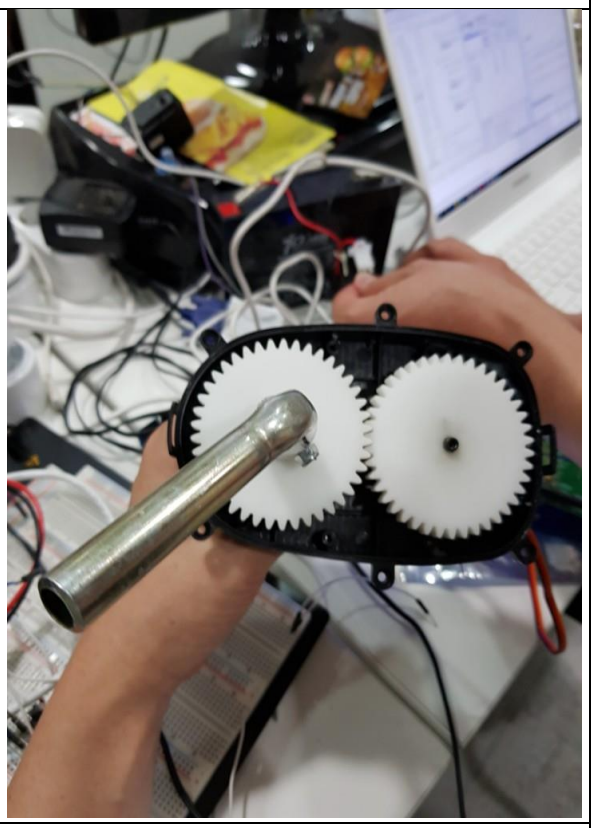
실험 한 결과, 최대 조향이 가능한 CMPA의 값은 675 ~ 3075 의 구간을 가지게 되었다. 이를 이용하여 1도마다 움직일 수 있는 알고리즘을 구하게 되었고 그 코드는 다음과 같다.

```
etpwmREG1->CMPA = 1875 + (13.3 * rx_data[1]);
```

rx_data[1]인 부분은 CAN 데이터로 들어오는 원하는 조향 각을 나타낸다.

그리고 rx_data[0]을 이용하여 직진, 좌회전, 우회전을 할지 판단하게 된다.

실험 결과

<table>
<tr><td>  </td><td>  </td></tr>
<tr><td>중립 제어</td><td>최대 우향각 제어</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>우향 45도 제어</td><td>좌향 45도 제어</td></tr>
</table>