대한상공회의소 서울기술교육센터
[Intel] 엣지 AI SW 아카데미 6기 – FW Programing

차량용 환풍기 프로젝트

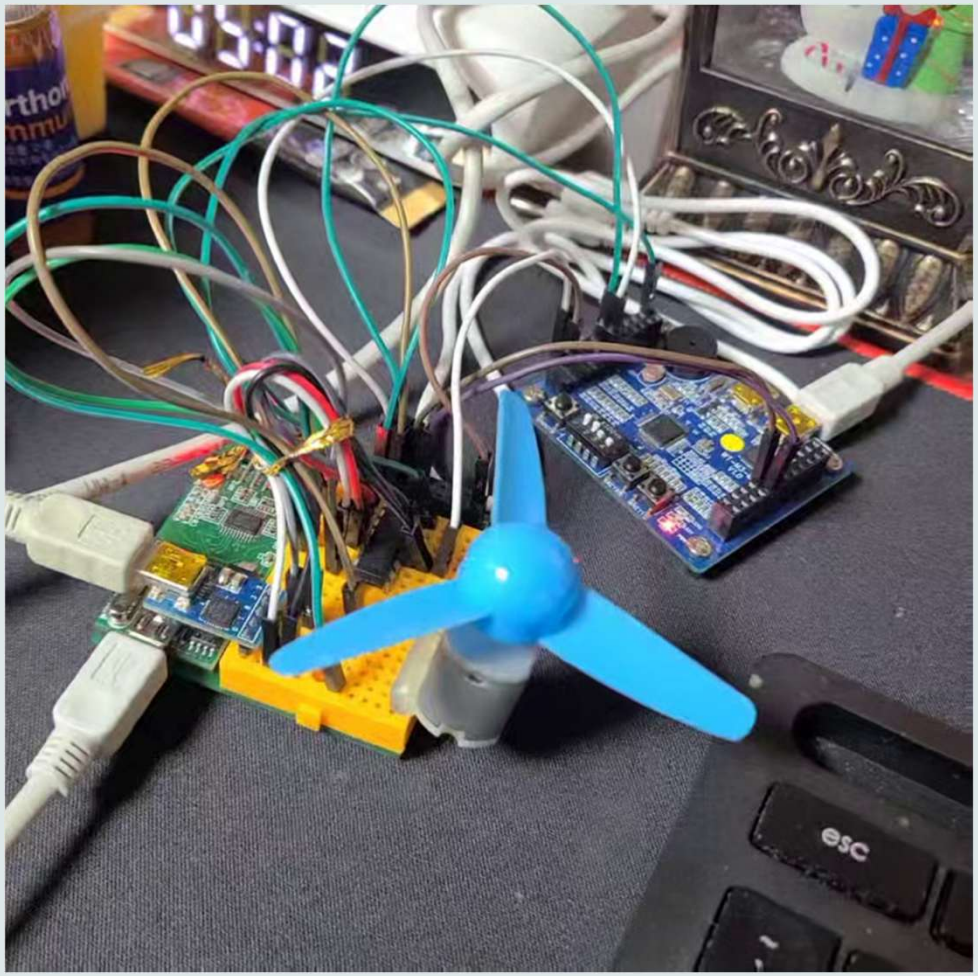
5팀
권태형 (rnjsxogud012@naver.com)
강준형 (junikang96@gmail.com)
설유승 (sus990415@naver.com)
신경임 (kkyungsh9771@naver.com)

목차

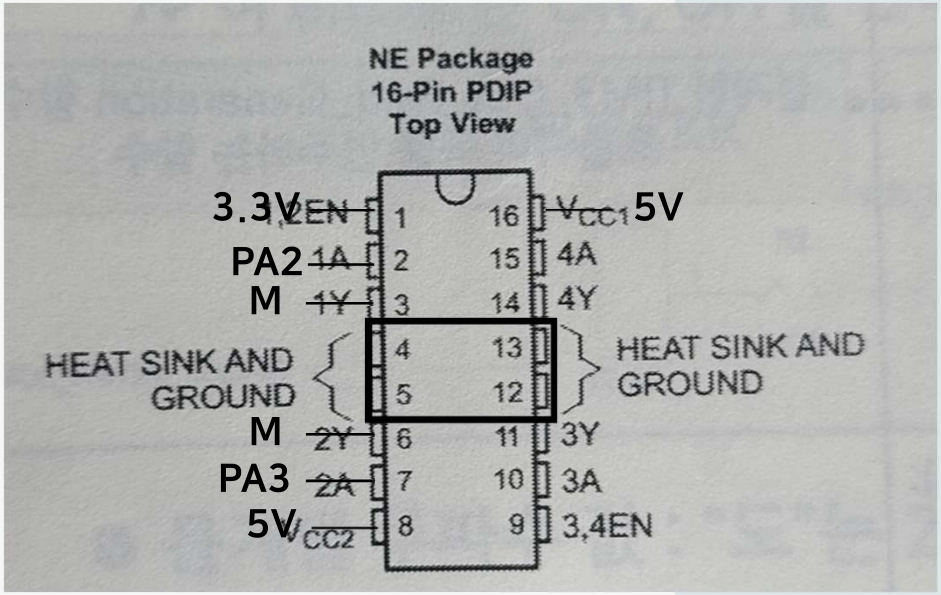
01. 프로젝트 요약

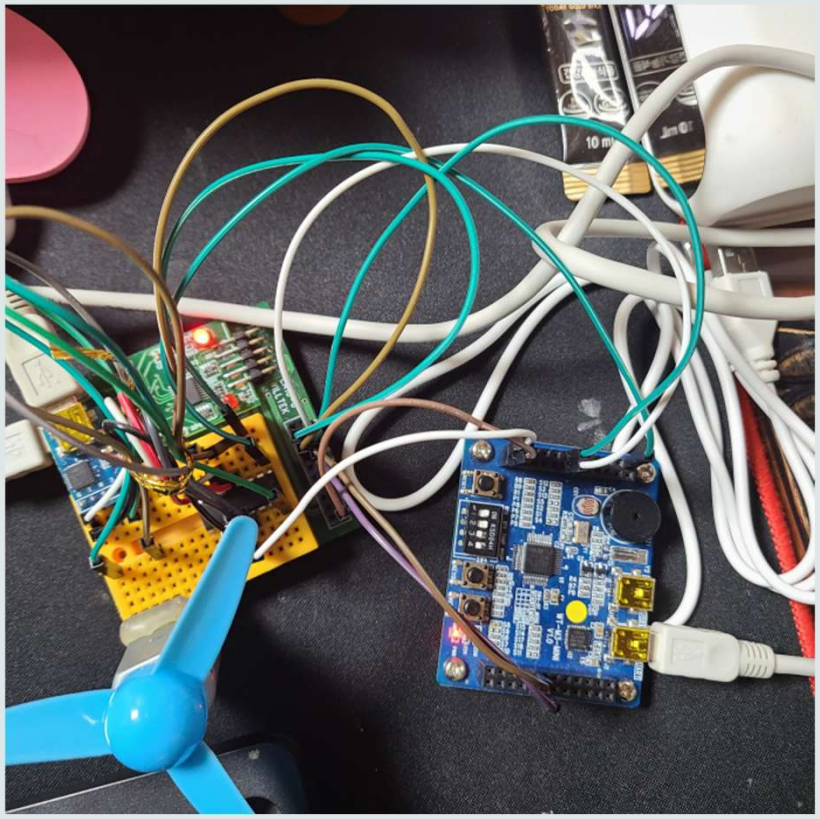
[개발 목표]
양방향 회전이 가능해 차량 내부의 공기를 배출하거나 외부의 공기를 흡입할 수 있는 환풍기 개발

[기능]
양방향 회전 가능
속도(1~9단계) 조절 가능
밝기에 따른 회전 속도 조절 기능



02. 프로젝트 설계과정 - 회로





02. 프로젝트 설계과정 - 코드

<main.c>
각 하드웨어별 초기화 함수를 Sys_Init 함수로 묶어 초기화

모터의 작동은 Motor_run 함수에 담아 작성함

초기에는 main 함수에서 모든 동작을 작성하여 수행한 후 각각을 나눠서 함수로 구현

```c
#include "device_driver.h"

static void Sys_Init(void) {
  Clock_Init();
  Uart_Init(115200);
  Key_Poll_Init();
  Motor_Init();
  Adc_Cds_Init();
}

void Main(void) {
  Sys_Init();

  for (;;)
  {
    Motor_run();
  }
}
```

02. 프로젝트 설계과정 - 코드

<motor.c>
motor를 활용하기 위한 기본 세팅

Motor_Init에서는 사용하려 하는 PA2, PA3의 Alternate Function output Push-pull 모드로 설정

PWM을 활용하기 위해 PA2,PA3에 연결된 TIM2 초기화

추가적으로 필요한 enum변수와 전역변수 선언

```c
#include "device_driver.h"

enum { FRONT, REVERSE, STOP };
enum { MANUAL, CDS };
uint8_t fan_state;
uint8_t state;
unsigned int KeyState;

void Motor_Init(void) {
  Macro_Set_Bit(RCC->APB2ENR, 2);
  Macro_Write_Block(GPIOA->CRL, 0xff, 0xbb, 8);
  Macro_Set_Bit(RCC->APB1ENR, 0);
  Macro_Write_Block(TIM2->CCMR2, 0x7, 0x6, 4);
  Macro_Write_Block(TIM2->CCMR2, 0x7, 0x6, 12);
  TIM2->CCER |= (0 << 13) | (1 << 12) | (0 << 9) | (1 << 8);
  TIM2->PSC = 8;
  TIM2->ARR = (double)400;
  state = MANUAL;
  fan_state = STOP;
}
```

02. 프로젝트 설계과정 - 코드

<motor.c>
최초 프로그램을 실행하면 모터를 수동으로 제어할 수 있는 상태

F 입력 - 정회전 , R 입력 - 역회전
S 입력 - 정지

정회전과 역회전의 경우는 1~9단계에 따른 속도를 제어하기 위해 매개변수를 입력받아 Duty rate가 달라지게 설정

```c
void Motor_Stop(void) {
  TIM2->CCR4 = TIM2->ARR * 100 / 100;
  TIM2->CCR3 = TIM2->ARR * 100 / 100;  // duty rate
  Macro_Set_Bit(TIM2->EGR, 0);
  TIM2->CR1 = (1 << 4) | (0 << 3) | (1 << 0);
}

void Motor_Front(unsigned short level) {
  TIM2->CCR4 = TIM2->ARR * 0 / 100;
  TIM2->CCR3 = TIM2->ARR * (level * 5 + 55) / 100;  // duty rate
  Macro_Set_Bit(TIM2->EGR, 0);
  TIM2->CR1 = (1 << 4) | (0 << 3) | (1 << 0);
}

void Motor_Reverse(unsigned short level) {
  TIM2->CCR3 = TIM2->ARR * 0 / 100;
  TIM2->CCR4 = TIM2->ARR * (level * 5 + 55) / 100;  // duty rate
  Macro_Set_Bit(TIM2->EGR, 0);
  TIM2->CR1 = (1 << 4) | (0 << 3) | (1 << 0);
}

void Motor_level_Front(char data) { Motor_Front((int)data - 48); }

void Motor_level_Reverse(char data) { Motor_Reverse((int)data - 48); }
```

02. 프로젝트 설계과정 - 코드

<motor.c>
KEY0를 눌렀을 때 바뀌는 밝기에 따라 속도가 제어되는 모드

조도 센서를 통해 밝기를 측정하고 밝기가 밝아지면 속도가 빨라지는 코드 작성

총 4개의 단계로 구성

```
void Motor_Light(void) {
  static int prev=-1;
  Adc_Start();
  KeyState = Key0_Get_State();
  volatile unsigned int AdcState = Adc_Get_Data();
  TIM2_Delay(1);
  int current;
  if (0x000 <= AdcState && AdcState < 0x400) {
    current =0;
    Motor_Front(1);
    if (prev!=current) {
      Uart1_Printf("%d\n",60);
    }
  }
   else {
     current = -1;
   }

   prev = current;
}
```

02. 프로젝트 설계과정 - 코드

<motor.c>
초기 모드인 수동 제어 모드에서의 모터 작동 함수

while문을 대기하는 중 key가 입력되면 탈출하고 모드가 변경되도록 설계

f와 r 입력시 최초 9단계로 시작하도록 설정

```
void Motor_Manual(void) {
  while (Macro_Extract_Area(USART1->SR,0x1,5)!=1)
  {
    if ((KeyState = Key0_Get_State())==1)
    {
      return;
    }
  }
  char data = USART1->DR;
  USART1->DR = data;

  switch (fan_state) {
    case FRONT:
      if (49<=data && data<=57)
      {
        Motor_level_Front(data);
      }
      if (data == 'S' || data == 's') {
        fan_state = STOP;
        Motor_Stop();
      }
      if (data == 'R' || data == 'r') {
        fan_state = REVERSE;
        Motor_level_Reverse('9');
      }
      if (data == 'F' || data == 'f')
        ;
      break;
    case REVERSE:
      if (49<=data && data<=57)
      {
        Motor_level_Reverse(data);
      }
      if (data == 'S' || data == 's') {
        fan_state = STOP;
        Motor_Stop();
      }
      if (data == 'F' || data == 'f') {
        fan_state = FRONT;
        Motor_level_Front('9');
      }
      break;
    case STOP:
      if (data == 'F' || data == 'f') {
        fan_state = FRONT;
        Motor_level_Front('9');
      }
      if (data == 'R' || data == 'r') {
        fan_state = REVERSE;
        Motor_level_Reverse('9');
      }
      break;
  }
}
```

02. 프로젝트 설계과정 - 코드

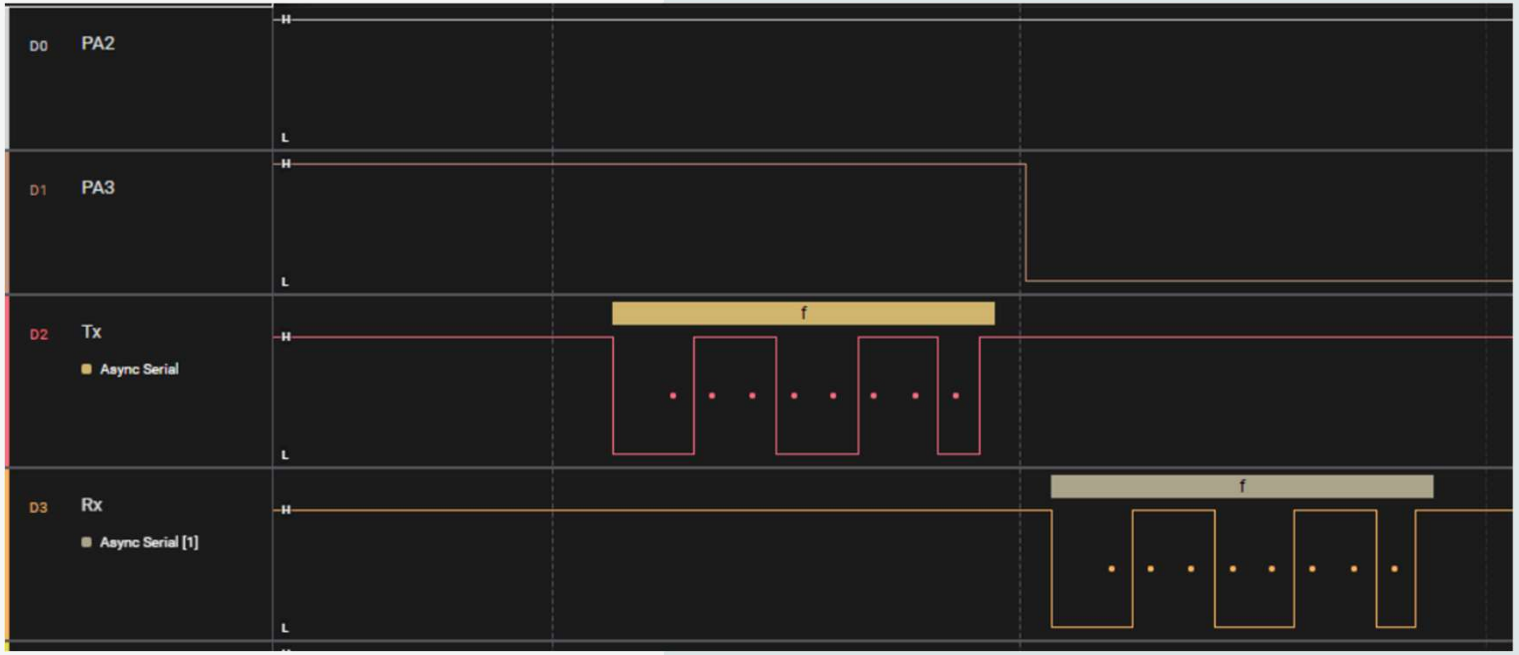
<motor.c>
수동 제어 모드와 밝기에 따른 속도 제어 모드가 key 입력에 따라 변경되도록 설정

```
void Motor_run(void) {

  switch (state) {
    case MANUAL:
      Motor_Manual();
      if (KeyState == 1) {
        state = CDS;
        KeyState = 0;
      }
      break;
    case CDS:
      Motor_Light();
      if (KeyState == 1) {
        state = MANUAL;
        fan_state = STOP;
        Motor_Stop();
        KeyState = 0;
      }
      break;
  }
}
```

02. 프로젝트 설계과정 - 타이밍 차트

<타이밍차트 - 수동 모드>



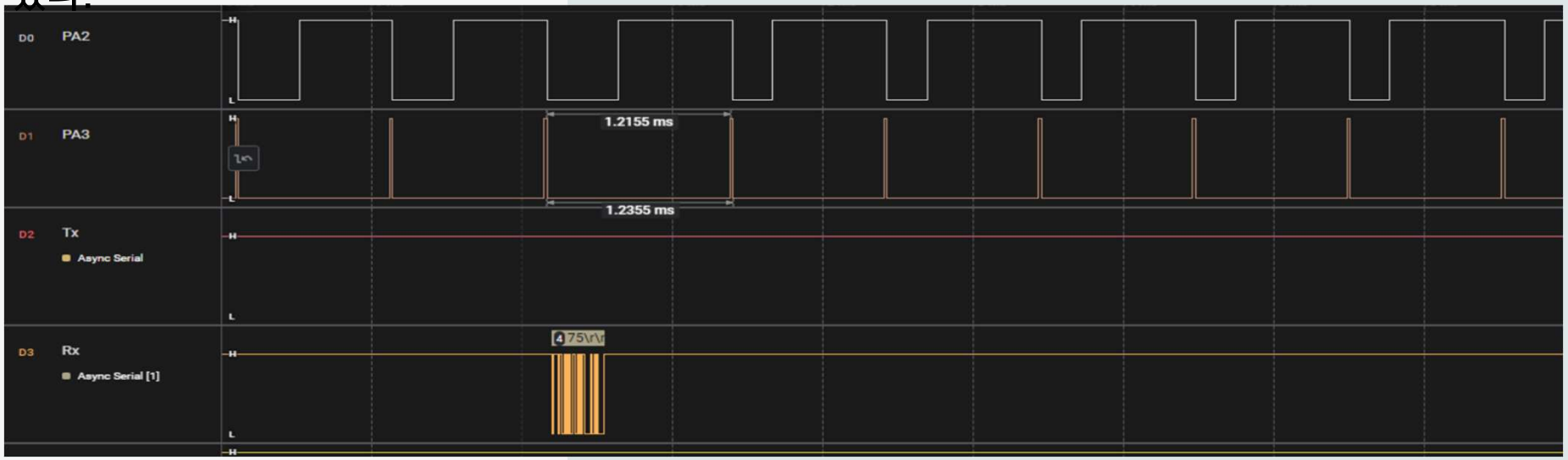
02. 프로젝트 설계과정 - 타이밍 차트

<타이밍차트 - 수동 모드>

02. 프로젝트 설계과정 - 타이밍 차트

<타이밍차트 - 밝기 모드>

밝기 모드에서는 밝기에 따라 Duty rate가 달라져서 PA2의 H/L 비율이 다른 것을 확인할 수 있다.



03. 프로젝트 후기 - 보완할 점 및 추후 과제

1. key를 입력받기 위해 while문에 조건을 달아 탈출하는 방법을 사용하였는데 Interrupt를 사용한다면 불필요한 조건문을 달 필요가 없을 것으로 보임.

2. UP&DOWN 모드를 구현하려 하였으나 시간의 부족으로 구현하지 못함. 추후 기회가 된다면 UP&DOWN 모드 추가

3. 가능하다면 공기 중 먼지 농도를 측정하는 센서와 습도 센서를 활용해 차 내부 먼지가 많으면 내부 공기 배출, 습도 차이에 의한 창문 김서림 현상 예방 등에 활용 가능할 것으로 보임

03. 프로젝트 후기 - 느낀 점

이번 프로젝트를 진행하며, 앞선 수업에서 배운 함수와 레지스터 정보를 활용해 코드를 작성했음에도 불구하고 여러 어려움에 직면했습니다.

그러나 프로젝트를 수행하면서 코드를 다시 차분히 분석하고, 레지스터 설정과 디버깅의 요령을 터득하게 되었습니다.

부족한 점이 많았지만 나름의 성과를 낸 것에 성취감을 느낄 수 있었습니다.

Q&A