25-1 임베디드시스템
Term Project 보고서

부산대학교 정보컴퓨터공학부 컴퓨터공학전공

202345813 문성현

202155503 강태훈

목 차

1. 프로젝트 개요

1-1. 임베디드시스템/RTOS 개요

임베디드시스템은 특정한 기능을 수행하기 위해 설계된 소형 컴퓨터 시스템으로, 하드웨어와 소프트웨어가 밀접하게 결합되어 있는 것이 특징이다. 일반적인 컴퓨터와는 달리 범용적인 사용보다는 특정 목적에 최적화된 동작을 수행하며, 크기가 작고 저전력이며 실시간 제어 기능이 요구되는 경우가 많다.

이러한 임베디드시스템에서는 동시 다중 작업 수행 및 정해진 시간 안에 작업 완료가 필요한 경우가 많기 때문에, RTOS(Real-Time Operating System)가 널리 사용된다. RTOS는 Task의 스케줄링, 동기화, 시간 관리, 인터럽트 처리 등의 기능을 제공하여 시스템이 실시간성을 유지하면서도 안정적으로 동작할 수 있도록 한다.

1-2. 주제 선정 이유 및 소개

”여름철이면 누구나 한 번쯤 겪는 일이 있다. 선풍기를 켜놓고 잠이 들었는데, 깜빡하고 타이머를 설정하지 않아 새벽에 춥고 불쾌하게 깨어난 경험이다. 반대로, 잠들기 전 선풍기를 끄기 위해 일어나야 할 때도 있다. 이런 사소하지만 반복되는 불편을 줄일 수 있다면, 사용자에게 편안함을 줄 수 있는 스마트한 시스템이 되지 않을까?” 이러한 일상 속 경험에서 출발하여, 우리는 버튼과 타이머, 센서를 활용해 사용자의 최소한의 조작만으로도 동작하고, 스스로 꺼질 수 있는 선풍기 제어 시스템을 기획하게 되었다. 단순히 켜고 끄는 수준이 아닌, 풍속을 단계적으로 조절하고, 타이머 설정 및 거리 감지를 통한 자동 종료 기능까지 포함함으로써, 보다 실용적이고 확장 가능한 임베디드시스템을 만드는 것을 목표로 삼았다.

1-3. 사용한 디바이스 소개

1.3.1 버튼 모듈

디지털 버튼 모듈은 사용자가 누르는 물리적인 입력을 감지하는 장치로, 내부 풀업 저항을 통해 기본적으로 HIGH 상태를 유지하며, 버튼이 눌리면 LOW 신호로 전환된다. 단순한 구조로 인해 다양한 임베디드시스템에서 입력 장치로 널리 활용된다.

1.3.2 3색 LED 모듈

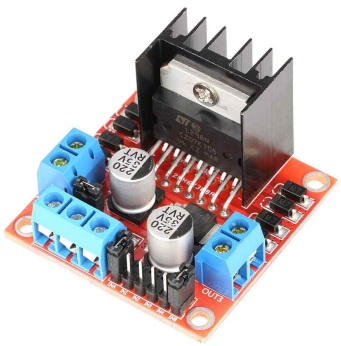
3색 LED 모듈은 빨강(R), 초록(G), 파랑(B)의 세 가지 색상을 단독 또는 조합하여 출력할 수 있는 시각 출력 장치이다. 각 색상의 핀에 LOW 또는 PWM 신호를 인가함으로써 다양한 색상을 구현할 수 있으며, 시스템 상태를 시각적으로 표현하는 데 주로 사용된다.

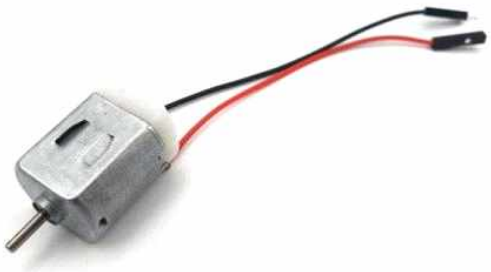
1.3.3 초음파 거리센서

초음파 거리 센서는 센서에서 보내는 초음파를 이용하여 물체의 거리를 측정하는 장치이다. Trig 핀을 통해서 10μs 길이의 HIGH 신호를 보내면 센서가 40kHz의 초음파를 8회 발신, Echo 핀에서 수신되면 LOW 상태로 바뀌며 센서가 HIGH 신호를 보낸 시점부터 LOW로 바뀌기까지 시간을 측정하여 거리를 계산한다.

1.3.4 5V DC 모터

DC 모터를 직접 보드에 연결하여 동작하는 것은 많은 전류 소비를 유발하므로 다른 기능 수행 또는 보드 자체에 문제가 생길 수 있으므로 모터 드라이버모듈인 L298N을 사용한다. 전원을 공급하고 보드에서 드라이버모듈로, 다시 드라이버모듈로부터 DC 모터로 연결하면 모터가 회전하게 된다.





2. 시스템 구성

시스템 전체에서 팬의 상태를 공유하고 제어하기 위해, 전역 구조체 Fan_Control_TypeDef를 정의하였다. 이 구조체는 두 개의 필드로 구성되어 있으며, 선풍기의 전원 상태와 풍속 레벨을 포함한다.

```
typedef struct {
    CPU_BOOLEAN fan_on;
    uint8_t     speed_level;
} Fan_Control_TypeDef;

volatile Fan_Control_TypeDef g_fan_control;

volatile uint32_t fan_timer_seconds = 0;
volatile CPU_BOOLEAN timer_active = DEF_FALSE;

static OS_SEM AppFanCtrlSem;
```

fan_on은 팬의 전원이 켜져 있는지 여부를 나타내는 논리형 변수로, TRUE일 경우 팬이 동작 중임을 의미한다. speed_level은 팬의 풍속 레벨을 나타내는 정수형 변수로, 0은 정지 상태이며, 1단부터 3단까지의 풍속을 단계별로 구분한다.

fan_timer_seconds는 타이머 기능을 구현하기 위해 사용되는 전역 변수로, 남은 시간을 초 단위로 저장한다. timer_active는 타이머가 현재 활성화되어 있는지를 나타내는 논리형 변수이다. 타이머가 작동 중일 때는 DEF_TRUE로 설정되며, LED 점멸 효과와 타이머 종료 조건 등에 영향을 준다.

본 프로젝트에서는 g_fan_control, fan_timer_seconds, timer_active와 같은 전역 변수에 모두 volatile 키워드를 명시하였다. 이 변수들이 하나의 Task뿐 아니라 여러 Task 또는 인터럽트 컨텍스트에 의해 동시에 읽히거나 변경될 수 있기 때문이다.

OS_SEM AppFanCtrlSem은 여러 태스크에서 공유되는 전역 변수들에 대한 동시 접근 충돌을 방지하기 위해 사용하는 세마포어이다. uC/OS-III의 OSSemPend() 및 OSSemPost()를 통해 임계 영역 보호를 수행하며, 시스템 상태의 안정성을 보장한다.

2-1. 버튼 입력

AppTaskButton 태스크를 통해 사용자 버튼 입력을 처리한다. 총 세 개의 버튼(BTN1, BTN2, BTN3)에 대한 입력을 감지하여 시스템의 전반적인 동작을 제어한다. 이 태스크는 약 50ms 주기로 버튼 상태를 스캔하며, 버튼이 눌리는 순간을 감지하기 위해 Falling Edge를 감지하여 입력 이벤트를 처리하는 구조이다.

공유 자원인 g_fan_control, fan_timer_seconds, timer_active는 다른 태스크들과도 공유되므로, uC/OS-III의 세마포어(OS_SEM)를 사용하여 접근 시 충돌을 방지하고 동기화를 보장하도록 하였다.

버튼	기능	설명
BTN1	전원	선풍기 On/Off
BTN2	풍속 조절	버튼 입력 시 1단 → 2단 → 3단 순환
BTN3	타이머 설정	3초씩 증가, 최대 4번의 입력(12초)까지 설정 가능 타이머 시간 경과시 선풍기 자동으로 Off

(실제 선풍기의 타이머 기능은 일반적으로 30분 단위로 설정되지만, 본 프로젝트에서는 시연 시간과 촬영 환경의 제약을 고려하여 타이머 단위를 3초로 축소하여 구현하였다.)

BTN1을 누르면 선풍기의 전원이 켜지거나 꺼지며, 전원이 꺼질 경우 풍속과 타이머도 함께 초기화된다. BTN2는 선풍기가 켜져 있는 상태에서만 작동하며, 풍속 레벨을 1단 → 2단 → 3단 → 1단 ... 순으로 반복하여 조절할 수 있다. BTN3은 타이머 시간을 3초씩 증가시키는 버튼으로, 총 12초까지 설정이 가능하다. 설정된 시간이 모두 경과하면 선풍기가 자동으로 꺼진다.

2-2. LED 제어

본 프로젝트에서는 LED 모듈을 활용하여 선풍기의 동작 상태와 타이머 상태를 시각적으로 사용자에게 전달하도록 설계하였다. LED 제어는 AppTaskFanControl과 AppTaskTimerControl 두 개의 태스크에서 상황에 따라 동적으로 제어된다.

1) 팬 풍속에 따른 LED 색상 표시 (AppTaskFanControl)

팬의 풍속 레벨은 전역 구조체 g_fan_control을 통해 관리되며, 풍속에 따라 다음과 같이 LED 색상이 변경된다.

풍속	LED
Off	꺼짐
1단	초록색
2단	파란색
3단	빨간색

AppTaskFanControl은 약 100ms 주기로 실행되며, 현재 풍속 값을 읽어 이에 해당하는 LED 색상을 설정한다. 이 과정은 모터의 PWM 출력과 함께 수행되며, 팬 동작 상태를 직관적으로 표현하는 시각적 피드백의 역할을 한다.

2) 타이머 동작 중 점멸 효과 (AppTaskTimerControl)

타이머가 활성화되었을 경우, LED는 일정한 주기로 꺼졌다 켜지기를 반복하여 사용자에게 타이머 작동 여부를 시각적으로 알린다. 이 점멸 제어는 AppTaskTimerControl 태스크 내에서 세마포어를 통해 공유 자원(fan_timer_seconds, g_fan_control.speed_level)에 접근하여 안전하게 수행되며, 이 점멸 효과는 아래와 같이 동작한다.

타이머	LED
Off	깜빡이지 않고 계속 점등
On	1.5초 주기로 깜빡임

타이머 시간이 모두 경과하면 선풍기는 Off되고, LED도 모두 꺼지게 된다.

2-3. 초음파 거리 센서

AppTaskUltrasonic 태스크를 통해 초음파 거리를 측정하며 이 기능은 버튼 조작이나 선풍기 모터 제어보다 우선순위가 낮게 설정한다. 사용자가 현재 선풍기를 사용하고 있지 않다고 판단될 경우 선풍기의 전원을 차단하므로 선풍기가 켜져있는지 상태를 확인하고 꺼져있을 경우 거리 측정은 실행되지 않고 다음 주기에 다시 실행된다.

현재 선풍기가 켜져있는지 확인하기 위해 세마포어를 획득하여 상태를 확인하고, 켜져있다면 Measure_Distance_mm을 실행하여 물체까지의 실제 거리를 측정하기 시작한다. 측정된 왕복 시간을 바탕으로 거리를 계산하며, 측정된 시간을 소리의 진행 속도(약 343m/s)로 곱한 뒤 초음파의 왕복 거리를 편도 거리로 변환하고 단위를 밀리미터로 맞추기 위해 2000을 나누어 값을 계산한다.

측정된 거리가 300mm(30cm) 이상 멀어지게 되면 선풍기를 꺼야 하는 조건이 충족하게 되며 세마포어에 접근하여 선풍기 상태에 접근, 선풍기를 끄게 되며 타이머 역시 0으로 초기화하여 다시 켜졌을 때 이전 상태가 유지되지 않도록 한다.

이후 지연상태에 진입후 다시 깨어나서 다음 거리 측정 사이클을 반복한다.

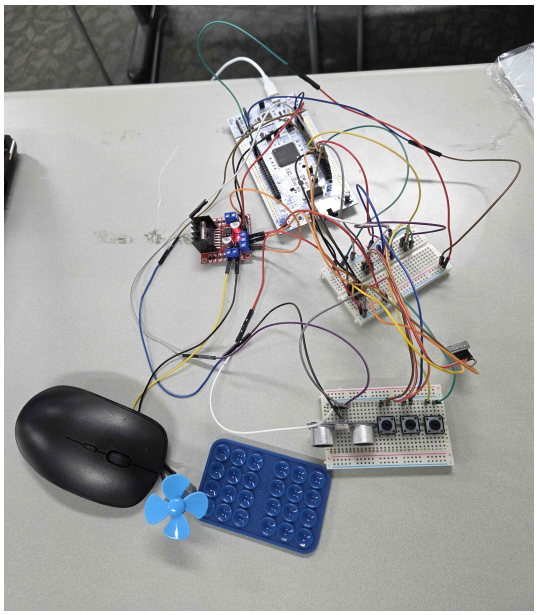
2-4. DC 모터 제어

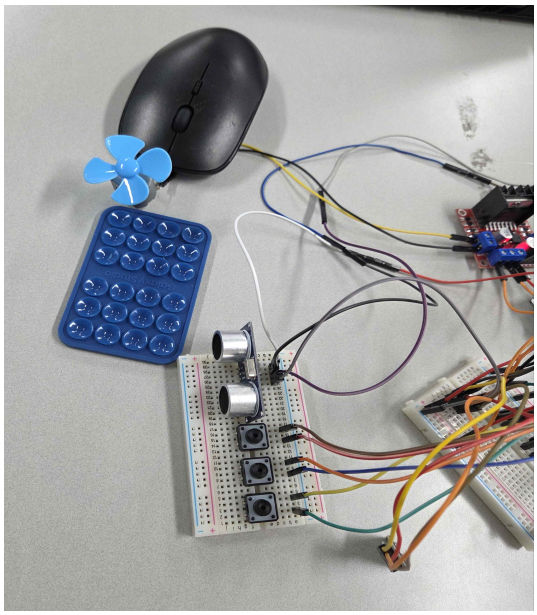
AppTaskFanControl 태스크를 통해 모터의 제어를 수행한다. 먼저 현재 선풍기의 상태를 확인하기 위해 세마포어에 접근하여 상태를 불러오기를 시도한다. 여기서 에러 없이 세마포어 접근에 성공하면 현재 선풍기의 켜져있는지 여부, 그리고 세기를 태스크 내 지역변수로 복사한 뒤 세마포어를 해제한다.

이후 설정된 값을 바탕으로 선풍기의 회전 상태를 업데이트하게 되며, 설정된 풍속 레벨에 따라 Duty Cycle을 설정, 모터가 단계 별로 다른 속도로 회전하게 된다. 설정된 단계에 따른 회전속도는 다음과 같다.

풍속	회전 속도(duty Cycle)
Off	0%
1단	60%
2단	80%
3단	100%

3. 프로젝트 결과 (사진)





4. 역할 분담

문성현 : 버튼, LED 모듈 관련
강태훈 : DC모터, 적외선 센서 관련
→ 전체적으로 공동 개발

5. 참고 자료

[1] STMicroelectronics, “NUCLEO-F429ZI,” Mbed OS Platform, https://os.mbed.com/platforms/ST-Nucleo-F429ZI/