마이크로프로세서 설계실험 Term Project 결과 보고서

경북대학교 전자공학부
마이크로프로세서 설계실험 15조
유한솔, 이세인

목 차

I. Term project 주제

이번 프로젝트 진행 주제는 'S32K144 Evaluation Board를 이용한 차량용 통합 제어기 및 클러스터 디스플레이 설계'이다.

II. 구성원 및 역할 분담

유한솔 - 제어기 요소 설계 위주
이세인 - 통신 모듈 및 디스플레이 설계 위주

III. 설계 목표 및 개발 일정

1. 주제 구상 및 개발 계획 수립
2. PyQt를 이용한 계기판 디스플레이 개발
3. 제어기 개발 1단계: 제어 구조 추상화 및 소프트웨어 계층 구체화
4. 제어기 개발 2단계: 개별 제어 모듈 설계 및 전체 기능 통합
5. 개발 결과 종합 및 보고서 작성

IV. 프로젝트 세부 기능

구현한 세부 기능은 다음과 같다.

1. 3가지 변속기 모드가 구현되어, 버튼을 눌러 모드 전환이 가능하다. Parking (이하 P) 모드에서는 모터가 회전하지 않으며, Reverse (이하 R) 모드에서는 모터가 역회전하고, Driving (이하 D) 모드에서는 모터가 정방향으로 회전한다. 모터 회전 속도는 가변 저항을 통해 제어할 수 있다.

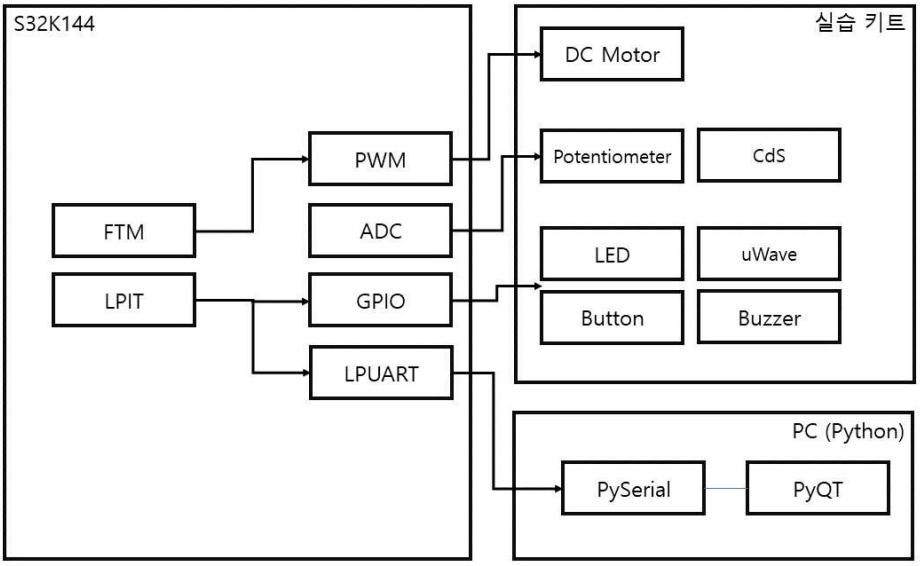
2. R 모드에서는 후방 감지기가 작동하여, 후방에 물체가 가까워질수록 버저가 더 짧은 간격으로 울린다.

3. D 모드에서는 전방 근접 경보가 작동하여, 전방에 물체가 가까워지면 짧은 버저 경고음이 울리고, 물체가 더 가까워지면 긴 버저 경고음이 울림과 동시에 모터가 작동을 정지한다.

4. 기어 모드에 상관없이 방향지시등, 전조등이 작동한다. 방향지시등은 버튼 입력으로 켤 수 있으며, 0.5초 간격으로 해당 방향에 맞는 LED를 점멸하며, Piezo 버저를 이용하여 딸깍거리는 구동음을 구현한다. 전조등은 버튼 입력을 통해 꺼짐, 차폭등, 전조등, 자동 모드를 전환할 수 있으며, LED 4개를 이용하여 꺼짐 모드에서는 0개, 차폭등 모드에서는 2개, 전조등 모드에서는 4개를 점등한다. 자동 모드에서는 조도 센서를 이용하여 전조등 모드를 상위 3개 중에서 알맞게 조절한다.

5. PC의 GUI 라이브러리를 이용해, MCU의 UART 기능을 이용한 직렬 통신으로 전송받은 데이터를 기반으로 계기판 화면을 구동한다.

V. 하드웨어 구성



<그림 1> 하드웨어 구성도

이번 프로젝트에 사용될 하드웨어의 전체적인 구성도는 그림 1과 같다.

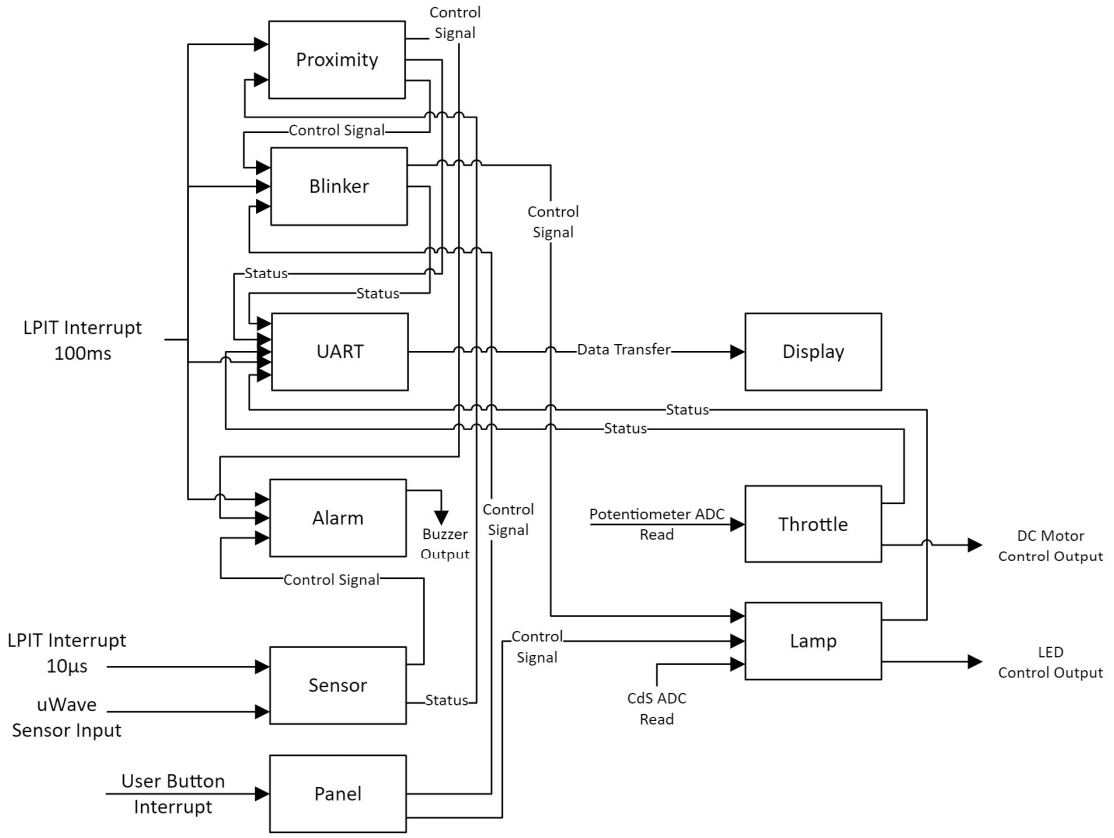
S32K144 보드는 ADC 모듈로 가변저항과 조도 센서의 값을 읽어, PWM을 통해 DC 모터를 제어한다. 또한 GPIO 모듈로 버튼 인터럽트와 LED, 초음파 센서, 버저를 제어한다. LPUART 모듈은 PC로 통신 신호를 전송한다.

PWM은 FTM 타이머에 동기화되어 동작하고, GPIO 중 일부와 LPUART 모듈은 LPIT가 주기적으로 생성하는 인터럽트 타이머에 동기화된다.

VI. 소프트웨어 구성

1. 소프트웨어 추상화 구조

전체적인 소프트웨어 추상화 다이어그램은 그림 2과 같다.

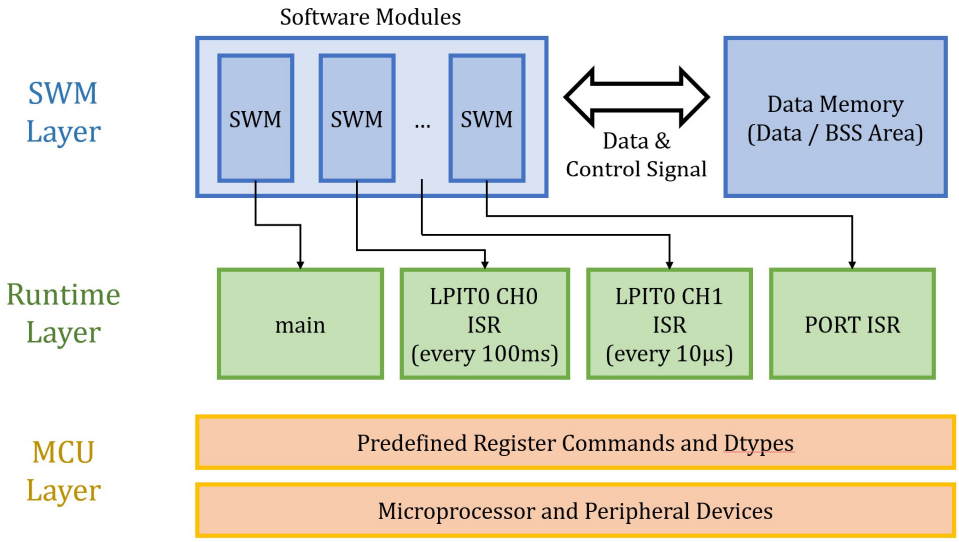


<그림 2> 소프트웨어 추상화 계층 구성도

각 블록은 하나의 소프트웨어 모듈을 의미한다. 그리고 블록으로 들어가는 화살표는 소프트웨어 모듈의 제어 신호 또는 데이터 입력을, 블록에서 나오는 화살표는 제어 신호 또는 데이터 출력을 의미한다.

소프트웨어 모듈에 대한 자세한 설명은 아래 소프트웨어 표준화 구조 항목에서 계속된다.

2. 소프트웨어 표준화 구조



<그림 3> 소프트웨어 표준화 계층 구성도

프로젝트의 가독성과 유지보수의 용이성을 위해, 소프트웨어 구조를 그림 3과 같이 체계화하였다.

가장 상위 계층인 SWM (Software Module) Layer는 소프트웨어 모듈, 그리고 소프트웨어 모듈 간의 데이터 상호 작용을 정의한 계층이다. 하나의 소프트웨어 모듈은 비슷한 역할을 하는 코드들을 모아 놓은 하나의 함수로 구성되며, 소프트웨어 모듈 간에는 계층 구조를 갖지 않는 수평 구조로 정의한다. 즉, 소프트웨어 모듈 내에서 다른 소프트웨어 모듈을 호출하지 않는다. 소프트웨어 모듈 간의 데이터 교환은 공통의 전역 변수 메모리 영역을 이용한다.

그 아래 Runtime Layer는 어떤 시점에 소프트웨어 모듈이 호출되어 실행될 것인지를 결정하는 계층이다. 총 4가지의 트리거 조건이 존재한다. 첫 번째로, main에 트리거 조건이 따로 없이 항상 실행될 모듈들이 포함된다. LPIT CH0 ISR은 매 100ms마다 호출되며, 비교적 긴 주기를 갖고 동작하는 모듈들이 포함된다. LPIT CH1 ISR은 매 10μs마다 호출되며, 짧은 주기를 갖고 동작하는 모듈들이 포함된다. 마지막으로, PORT ISR에는 사용자 입력에 반응하여 동작해야 하는 모듈들이 포함된다.

가장 아래 계층인 MCU Layer는 컴파일러 단계와 실제 실행 단계를 의미한다. 이 계층에서는 제어 신호 및 통신 데이터의 구조를 정의하고, 추상화된 명령을 실제 레지스터 제어 코드와 mapping 한다.

위와 같은 계층 구조의 핵심은, 어떤 한 계층을 구현할 때는 다른 계층에 대한 정보가 적거나 없더라도 구현에 문제가 없어야 한다는 것이다. 즉, 소프트웨어 모듈을 구현할 때는 모듈 자체의 알고리즘에만 집중하여 구현할 수 있어야 한다. 예를 들어, 소프트웨어 모듈을 구현할 때, MCU Layer에 속하는 레지스터를 설정하는 코드를 직접 입력하는 대신, 미리 정의된 추상화된 명령을 입력하여 레지스터를 제어하는 것이다. 이를 통해, 코드의 흐름을 단순화하고 기능의 수정과 추가를 더욱 간편하게 할 수 있다.

3. 단위 기능 구성

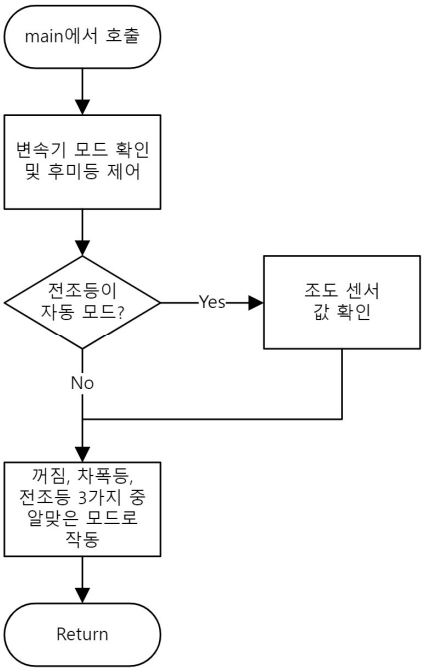
단위 기능의 동작은 소프트웨어 모듈 내에 정의되어 있다.

- Blinker 모듈

이 모듈은 Panel 모듈이 생성한 제어 신호에 따라, 방향지시등을 제어하는 제어 신호를 생성하여 Lamp 모듈로 전달하는 역할을 한다. Runtime Layer의 LPIT0 CH0 ISR 함수에서 호출된다. 버튼 입력에 알맞게 0.5초마다 제어 신호를 생성하는 간단한 flow를 가지고 있어 flow chart는 따로 작성하지 않았다.

- Lamp 모듈

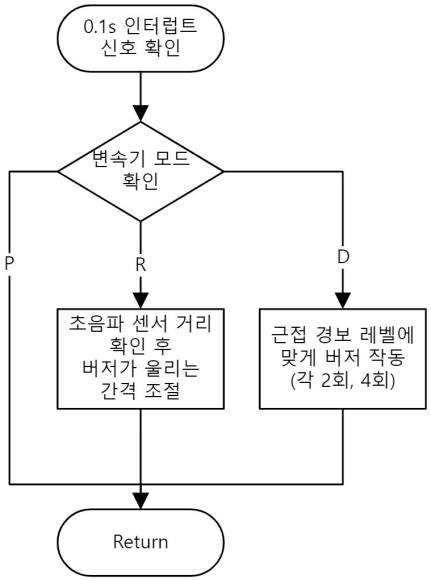
이 모듈은 변속기 모드에 알맞게 후미등을 제어하고, 꺼짐, 차폭등, 전조등, 자동 모드에 따라 알맞게 전조등을 제어하고, Blinker 모듈이 생성한 방향지시등 제어 신호에 맞추어 방향지시등을 제어한다. Runtime Layer의 main 함수에서 호출된다.



<그림 4> Lamp 모듈의 flow chart

- Alarm 모듈

이 모듈은 Sensor 모듈이 생성한 거리 값에 따라 버저를 제어하는 모듈이다. Runtime Layer의 LPIT0 CH0 ISR 함수에서 호출된다.



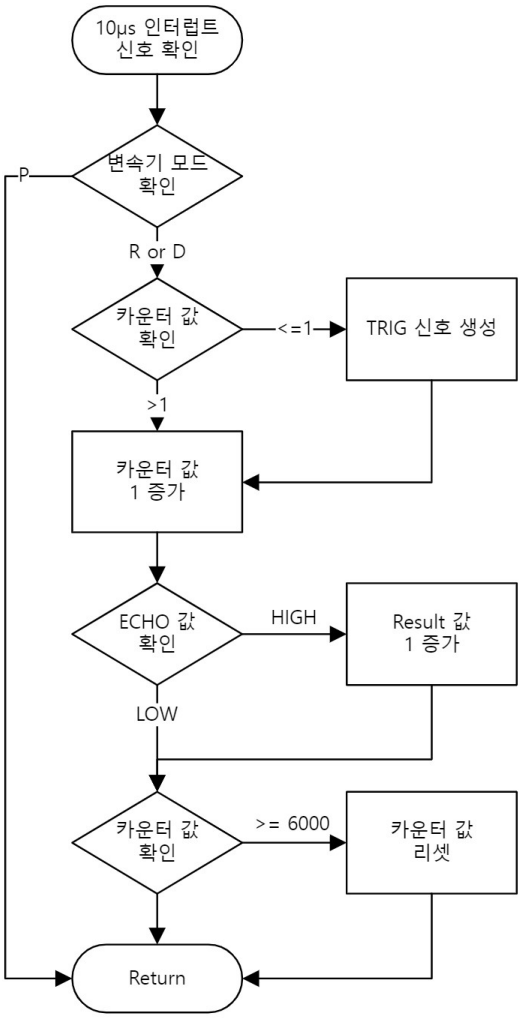
<그림 5> Alarm 모듈의 flow chart

- UART 모듈

이 모듈은 PC로 전송할 UART 신호를 생성하고 전송하는 모듈이다. Runtime Layer의 LPIT0 CH0 ISR 함수에서 호출된다. 이 모듈은 분기 제어가 없는 단일 흐름으로 구현되기 때문에 flow chart는 따로 작성하지 않았다.

- Sensor 모듈

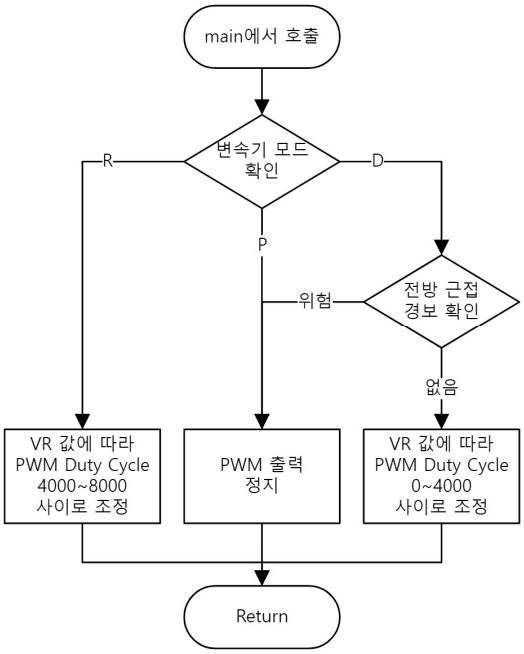
이 모듈은 초음파 센서를 동작시키는 모듈이다. Runtime Layer의 LPIT0 CH1 ISR 함수에서 호출된다.



<그림 6> Sensor 모듈의 flow chart

- Throttle 모듈

이 모듈은 가변저항 값을 읽고, DC 모터를 제어하는 모듈이다. Runtime Layer의 main 함수에서 호출된다.



<그림 7> Throttle 모듈의 flow chart

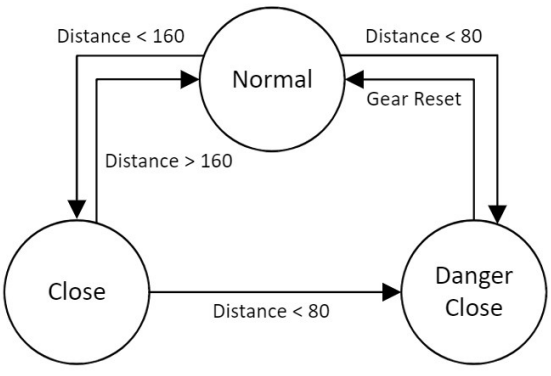
- Panel 모듈

이 모듈은 사용자 버튼 입력을 읽어 알맞은 제어 신호를 생성하는 모듈이다. Runtime Layer의 PORTC ISR에서 호출된다. ISR 내에서 interrupt flag를 확인하여 변수의 값을 알맞게 설정하는 간단한 flow를 가지고 있어 flow chart는 따로 작성하지 않았다.

- ProxWarning 모듈

이 모듈은 전방 근접 경보를 관리하는 모듈이다. Sensor 모듈이 생성한 거리 값을 읽어, 전방 경고 신호를 알맞게 생성한다. Runtime Layer의 LPIT0 CH0 ISR에서 호출된다.

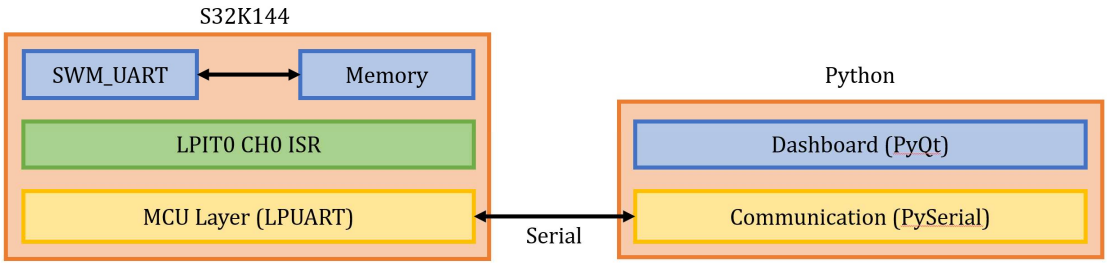
이 모듈의 동작은 아래 Finite State Machine 다이어그램으로 간단하게 표현할 수 있다.



<그림 8> ProxWarning 모듈의 FSM Diagram

- Display 모듈

디스플레이 모듈은 엄밀히 말하면 현재 정의된 표준화 구조의 소프트웨어 모듈 계층에 포함되지 않는 별개의 개념이다.



<그림 9> 소프트웨어 표준화 구조를 바탕으로 한 Display 모듈의 구현

그림 9와 같이, 디스플레이 모듈은 S32K144의 SWM Layer의 UART 모듈과, Runtime Layer의 LPIO0 CH0 ISR, 그리고 MCU Layer의 LPUART 기능을 통해 PC(Python)의 PySerial 패키지와 직렬 통신을 수행하고, 이 통신 데이터를 바탕으로 Qt GUI 라이브러리를 이용해 계기판을 구동하는 방식이다.

VII. 논의 사항

구현된 프로젝트 코드는 대부분 정상적으로 작동하지만, 더 높은 신뢰성을 위해 개선이 필요한 사항이 있다.

현재 SWM Layer에서는 모든 데이터의 저장을 공유 메모리 영역에 의존하고 있다. 이는 Runtime Layer에서 소프트웨어 모듈 함수의 call과 return이 계속 반복되는 구현 방식 때문으로, 함수의 return 이후에도 데이터를 계속 유지하기 위해서는 불가피한 방법이다. 하지만 이 방법은 interrupt handling이나 context switching의 진행 상황에 따라 data hazard를 발생시킬 위험이 있으며, 이 메모리 영역은 어디에서나 접근이 가능한 개방된 영역이기 때문에 의도치 않게 데이터 무결성이 깨지게 될 가능성이 존재한다. 이를 해결하기 위해, 다른 소프트웨어 모듈에서는 보이지 않는, 소프트웨어 모듈 간의 데이터 전달만을 위한 계층을 새롭게 정의하거나, 함수의 스택 영역을 사용할 수 있도록 Runtime Layer를 새롭게 정의하는 방법 등을 시도해 볼 수 있을 것이다.

VIII. 소스 코드

프로젝트 코드 전문은 아래 GitHub Repository에서 확인할 수 있다.
https://github.com/lsin07/mcu_termproject