REPORT

22조 텀프로젝트 결과 보고서

속도측정기

수강과목: 임베디드시스템

담당교수: 백윤주 교수님

학 과: 전기컴퓨터공학부 정보컴퓨터공학전공

조 원: 201624481 박윤형, 201624517 오세영

목차

1. 개요

임베디드 시스템이란 하드웨어와 소프트웨어가 내장되어 정해진 특정 동작을 수행하기 위한 기능을 제공해주는 시스템이다. 소프트웨어만 설치하면 원하는 작업을 수행할 수 있는 일반적인 PC와 다르게 임베디드 시스템은 특수한 목적을 가지고 설계된다. 따라서 임베디드 시스템은 전력 소비가 적고, 저렴해 장치를 효율적으로 제어하는 방법으로 사용된다. 임베디드 시스템을 구성하는 장치와 소프트웨어는 제한적이다. 구현 방식에 따라 하드웨어로만 구성을 할 수도 있고, 소프트웨어와 함께 구현할 수도 있다. 하드웨어 비중이 높은 시스템은 빠르지만 수정이나 추가 확장이 어렵기 때문에 대부분의 임베디드 시스템은 하드웨어와 소프트웨어를 조합하여 개발하는 것이 일반적이다.

2. 과제의 필요성

자동차는 우리가 일상생활을 하는데 반드시 필요하다. 자가용과 버스 및 택시 같은 교통수단이 없다면 우리의 생활반경은 굉장히 좁아져서 매우 불편하고, 문제가 많이 생길 것이다. 이렇게 우리의 생활에 유익한 자동차에게 이점만 있는 것은 아니다. 자동차가 가진 많은 단점들 중 가장 문제인 건 단연 교통사고이다. 자동차로 인해 벌어지는 교통사고는 순식간에 여러 생명들을 앗아갈 정도로 치명적일 수 있다.

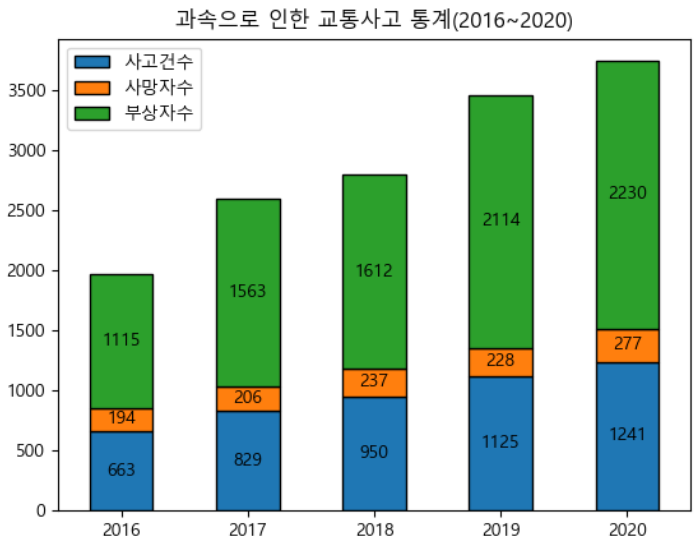

그림-1. 국내 과속으로 인한 교통사고 통계(도로교통공단 제공)

교통사고의 원인은 다양하고 모두 위험하지만, 특히 과속으로 인한 교통사고는 일반 교통사고와 비교해 사망 위험도가 매우 높다. 실제로 2019 년을 기준으로 비교했을 때, 전체 교통사고는 68.4 건당 1 명의 사망자가 나왔지만, 과속 교통사고는 4.9 건당 1 명으로 과속 교통사고로 인한 사망자 발생률은 전체 교통사고 발생률보다 14 배나 높다는 것을 알 수 있다. 그림-1 의 표에 따르면 우리나라에서 발생한 과속 교통사고는 지난 5 년간 꾸준히 높아져 2020 년에 발생한 사고건수와 부상자가 각각 2016 년에 비해 2 배 가까이 오른 것을 알 수 있다.

지난 해 미국에서의 교통사고 사망률이 전년과 비교해 증가한 것으로 나타났다. 그러나 역설적이게도 같은 기간 교통사고 건수는 감소했다. 세계보건기구(WHO)는 코로나19 봉쇄령 기간에 전 세계 80개 이상 도시에서 전반적인 교통사고 건수는 줄어든 반면 교통사고 사망자 수는 감소하지 않았다고 발표했다. 이것은 치명적인 대형 교통사고가 전보다 늘었다는 의미이다. WHO 는 코로나 19 기간 교통사고 사망자 증가의 주요 원인으로 과속을 꼽았다. WHO 의 연구에 따르면 평균 속도가 시속 1km 증가할 때마다 충돌 위험은 4~5% 증가하는 것으로 나타났다. 또한 시속 30km 속도로 주행하는 차량과 부딪힌 경우 보행자의 생존 가능성은 99%지만, 시속 50km 로 높아졌을 때의 생존율은 80%로 떨어졌다.

이러한 상황에서 우리 조는 이번 과제를 통해서 임베디드 시스템을 이용해 과속 단속기를 만들기로 결심했다. 이렇게 임베디드 시스템을 통해 저렴한 가격으로 과속을 방지할 수 있는 수단을 만들어 사용한다면 여러 장소에 단속기를 설치하여 과속 방지를 전보다 더 효율적으로 수행할 수 있게 될 것이다.

3. 기술적 접근 방법 및 개발 내용

3.1. 초음파 센서의 사용

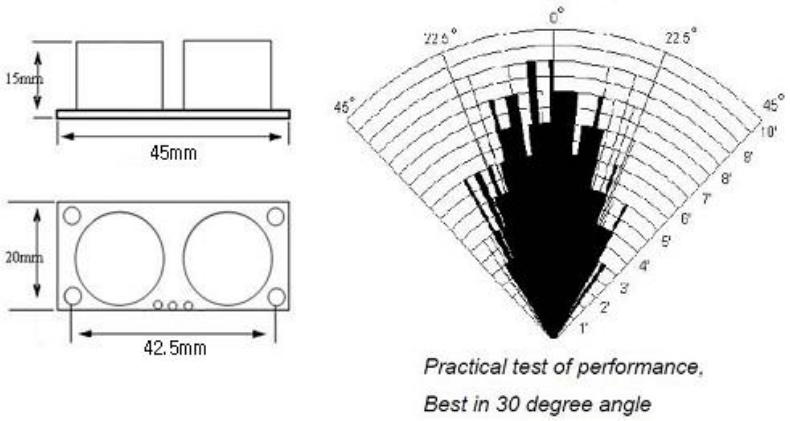

그림-2. HC_SR04 의 크기 및 측정 범위

그림-2 는 우리 조가 이번 프로젝트에서 사용한 주요 부품인 초음파 거리센서 HC_SR04 의 측정 범위를 보여준다. 해당 센서의 유효 측정 각도는 양쪽 각각 15 도 이내로 총 30 도 정도 된다. 측정 거리는 최소 2cm 에서 500cm 이다.

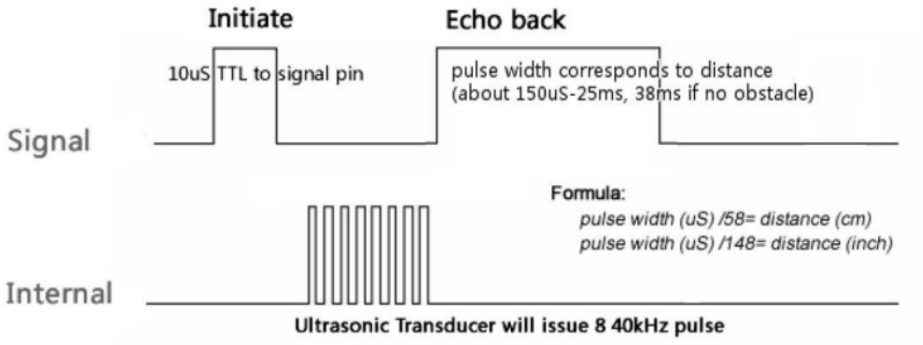

그림-3. HC_SR04 의 Timing Diagram

HC_SR04 를 사용하기 위해서는 위의 Timing Diagram 에 따라 측정 함수를 만들어줄 필요가 있다. Timing Diagram 에 따르면 HC_SR04 에 10us 의 신호를 입력하면 거리에 따라 최소 150us 부터 최대 25ms 의 출력이 발생한다. 다만 센서의 최대 측정거리가 너무 길기 때문에 본 프로젝트에서는 30cm 까지만 측정하는 것으로 설정했다. 이 출력신호는 측정되는 물체와의 거리에 따라 길어진다. 코드에서는 두개의 HC_SR04 를 제어하는 함수를 각각 read_ultrasonic1(), read_ultrasonic2()로 정의했다. 또한 10us 의 신호를 주기 위해 us 단위의 delay 를 만들어내는 함수인 TIM2_us_Delay()함수를 정의했고 이 함수에서는 TIM2 를 사용해서 delay 를 발생시킨다.

3.2. 초음파 센서를 사용한 속도 계산

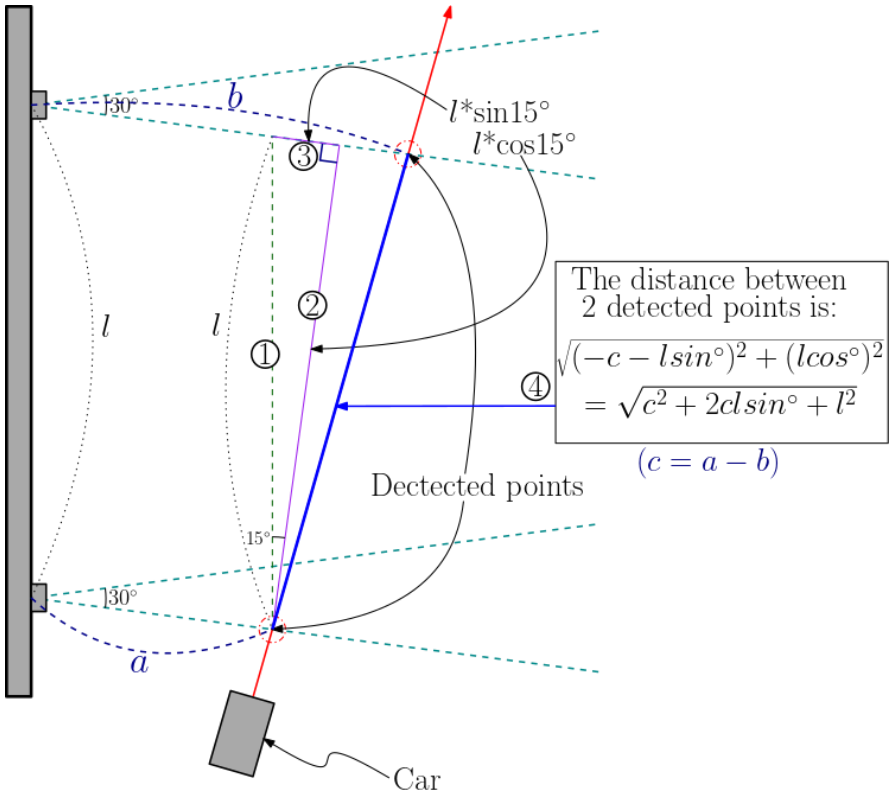

그림-4. 초음파 거리센서를 사용한 이동 거리 측정 방법

그림-4 는 초음파 거리센서 2 개를 사용하여 움직이는 물체의 이동거리를 측정한 방법을 설명한다. 먼저 두개의 초음파 거리센서의 유효 측정 범위에 처음 물체가 닿는 각각의 점을 표시하고, 이 두 점을 잇게 되면 물체의 이동거리가 된다. 거리를 구하기 위해 일단 **1)** 첫 번째 점에서 벽에 평행이 되게 선을 그어주면 해당 선의 길이는 센서 간의 거리 l이 된다. **2)** 같은 점에서 두 번째 센서의 유효 측정범위에 수직이 되게 선을 그어준다. 이 때 두 선분 사이의 각도는 15 도이므로 선의 길이는 $l * cos15°$ 가 된다. **3)** 마찬가지로 두 선분으로 이루어진 삼각형의 나머지 변의 길이는 $l * sin15°$가 된다. **4)** 이제 구하려는 길이가 포함된 직각 삼각형의 두 변의 길이를 알고 있으므로 피타고라스의 정리를 이용하여 이동거리를 구할 수 있다. 구한 이동거리를 걸린 시간으로 나누면 물체의 속도를 구할 수 있다.

3.3. 능동 부저 모듈

그림-5. 능동 부저 모듈

능동 부저 모듈은 전압을 인가해주고 신호를 주면 일정한 음의 소리가 출력된다.

4. 시스템 및 task 구성

4.1. Task 및 함수 설명

```
/*
*********************************************************************************************************
*                                          FUNCTION PROTOTYPES
*********************************************************************************************************
*/
static  void  AppTaskStart          (void     *p_arg);
static  void  AppTaskCreate         (void);
static  void  AppObjCreate          (void);

static void AppTask_Measure(void *p_arg);       //task measure the speed of object
static void AppTask_Alert(void *p_arg);         //control LED and buzzer
static void AppTask_USART(void *p_arg);         //take speed limit from USART and print the speed of object

static void  sensor_Init();
void TIM2_us_Delay(uint32_t delay);
uint32_t read_ultrasonic1();
uint32_t read_ultrasonic2();
uint32_t calc_speed(uint32_t distance1, uint32_t distance2,  OS_TICK time);
uint32_t isqrt(uint32_t number);
```

main 함수에서 AppTaskStart()를 생성하고 AppTaskStart()에서 AppTask_Measure(), AppTask_Alert(), AppTask_USART()를 생성한다. sensor_Init()은 초음파 거리센서와 능동 부저를 사용하기위해 pin 설정을 하는 함수로서 AppTaskStart에서 실행된다.

AppTask_Measure()은 물체가 두개의 초음파 거리센서를 순서대로 지나갈 때 물체의 속력을 측정한다. 물체와의 거리를 측정하기 위해 read_ultrasonic1()와 read_ultrasonic2()가 사용되고 측정된 지점의 시간차를 구하기 위해 OSTimeSet()과 OSTimeGet()을 사용했다. 이렇게 구해진 두개의 거리와 시간을 바탕으로 calc_speed() 함수에서 속력을 계산한다. 계산된 속력이 제한속도보다 높은지 판단해서 전역변수에 과속 여부를 판단한다.

AppTask_USART()는 serial통신을 위한 task이다. USART로 제한속도를 입력받아 저장하고 AppTask_Measure()에서 측정된 속력과 과속여부를 USART로 출력해 모니터링 할 수 있다. Task는 무한루프로 실행되며 프로그램을 실행하고 첫 loop에서는 제한속도를 입력받고 그 다음 loop부터는 속력을 모니터링한다.

AppTask_Alert()는 AppTask_Measure()에서 물체가 초음파 거리센서를 지나갔다고 판단되었을 때 해당 물체의 과속 여부에 따라 보드의 LED와 능동부저를 조작한다. 물체가 감지되지 않을 때 LED1을 켜고, 물체가 감지되었지만 속도제한을 준수할 때 LED2를 1초동안 켜고, 감지된 물체가 과속 상태일 때 1초동안 LED3을 켜고 능동부저를 울린다.

4.2. Task 간의 통신

```
118 typedef enum{
119     NOT_PASSING,
120     REGULATION_SPEED,
121     OVER_SPEED
122 }measurement_status;
123
124 measurement_status measure = NOT_PASSING;
125
126 int waiting_input = 1;      // 1: waiting USART input, 0: don't wait USART input
127 int speed_limit, speed;
128
129 OS_SEM MySem;                           // Semaphore
130
```

각 Task간의 통신은 semaphore와 전역변수를 통해 이루어진다. 코드에서 MySem이라는 이름의 Semaphore를 선언하고 통신에 사용되는 전역변수는 제한속도를 저장할 speed_limit, 측정된 속도를 저장할 speed, 물체감지여부와 과속여부를 저장하는 measure가 있다. 각각의 Task들의 priority는 AppTask_Measure가 0, AppTask_USART가 1, AppTask_Alert가 2이다.

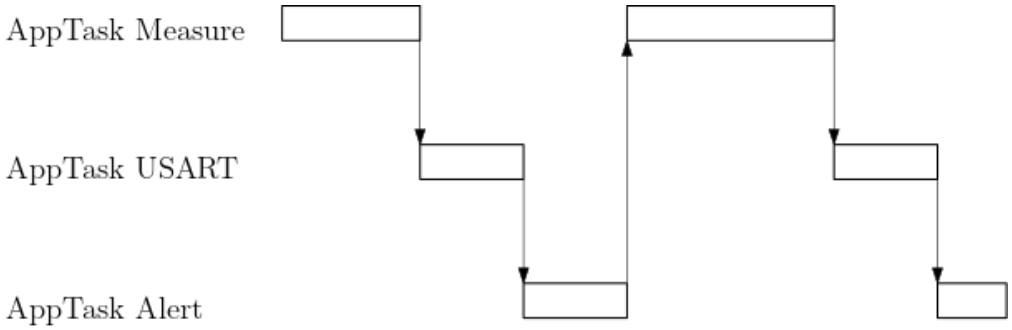

그림-6. Semaphore diagram

세 task들의 semaphore를 점유하는 순서는 그림-6과 같다. AppTask_Measure()에서 속도를 측정하기 전에 OSSemPend()로 semaphore를 점유하고 감지된 물체의 과속여부를 판단하면 측정된 속력과 과속여부를 전역변수에 저장하고 OSSemPost()로 semaphore를 푼다. AppTask_Measure()가 semaphore를 풀 때 기다리고 있는 AppTask_USART()와 AppTask_Alert()중 우선순위가 높은 AppTask_USART()가 semaphore를 점유한다. AppTask_USAT()에서 전역변수에 저장된 물체의 속력과 과속여부를 serial통신으로 출력한후 OSSemPost()로 semaphore를 풀어준다. 마지막 우선순위가 가장 낮은 AppTask_Alert()가 semaphore를 점유하고 전역변수에서 감지된 물체의 과속여부에 따라 LED와 능동부저를 제어하고 OSSemPost()로 semaphore를 풀어준다.

5. 역할분담

조원	역할
박윤형	Task구현, Task간의 통신, 입출력 pin 설정, 초음파 거리센서 사용함수 작성
오세영	물체 속도 계산 모델 설계, 속도계산 함수 작성, 하드웨어 제작, 보고서 작성

6. 결과 및 토의

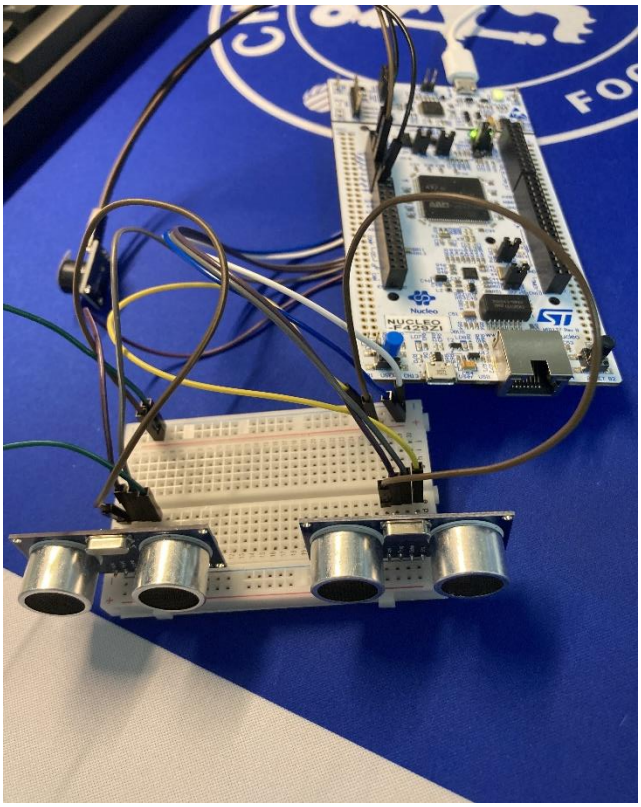

그림-7. 보드 및 센서 구성

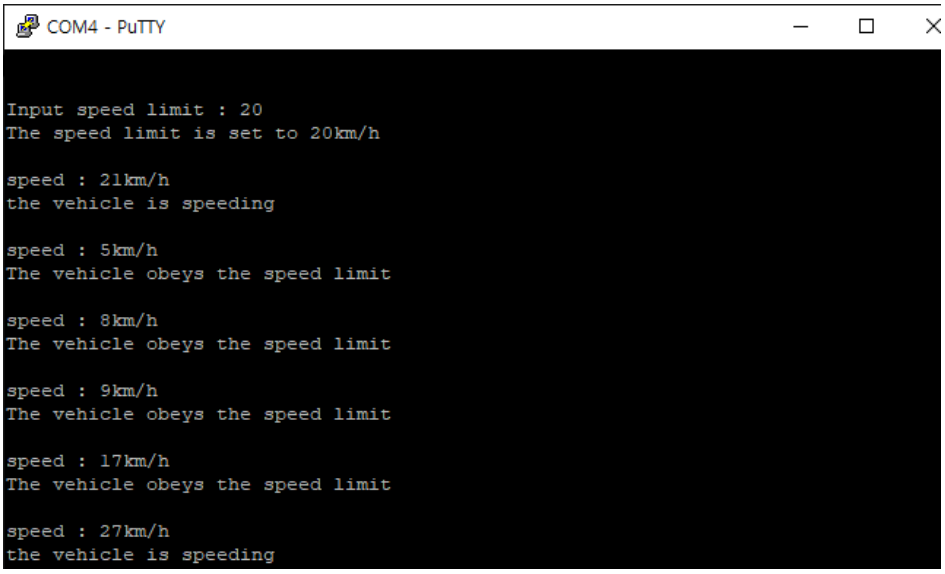

그림-8. Serial 통신을 통한 모니터링

그림-7에 보이는 것처럼 STM32F409보드와 초음파 거리센서, 능동 부저 모듈을 사용해 속도측정기를 구현해 보았다. 보드를 시작하고 물체가 두 센서에 순서대로 잡히게 되면 그림-8의 맨 위와 같이 serial 통신으로 제한속도를 설정할 수 있다. 이후에 두 센서에 물체가 지나가면 속도가 측정되고 측정된 속도에 따라 과속 여부를 알려준다. 물체가 제한속도를 넘은 것으로 판단되면 빨간색 LED가 켜지고 부저음이 울린다. 물체가 제한속도를 준수한 것으로 판단되면 파란색 LED가 켜진다. serial통신에서는 물체의 속도가 측정될 때마다 화면에서 속도 확인이 가능하다. 다만 속도 측정 시 두 초음파 센서 사이의 거리가 좁기 때문에 어느정도 이상의 속도는 측정이 불가하다는 한계점이 존재했다. 이 한계점은 초음파 센서 사이의 거리를 벌림으로서 해결할 수 있지만 너무 많이 벌리면 하나의 물체가 온전히 지나가기 전에 다른 물체가 감지될 수 있기 때문에 시스템의 많은 부분을 수정해야 할 수 있다.