강의명: 임베디드 시스템
숙제 번호: 3
숙제 제목: Pulse-width modulation(펄스-폭 변조)
학생 이름: 정우성
학번: 201810890

1. 프로그램 pwm-led

1.1 프로그램 코드 쓰기

```
#include "mbed.h"

PwmOut led(PTA1); // led = PTA1
int main()
{
float duty;
led.period(10.0 / 1000);
while (true) {
for(duty = 0.0; duty < 1.1; duty = duty + 0.1) {
led.write(duty);
thread_sleep_for(1000);
}
}
}
```

1.2 프로그램 작성 아이디어 혹은 이유 설명 쓰기

pwm를 사용하기위해 PwmOut라는 class에 객체를 생성한다. PTA1핀을 pwm출력 모드로 설정한다. 그 다음 Period라는 함수를 통해 이 pwm에 주기를 10ms로 설정합니다.

while문 안에 for문에서 duty가 반복을 하며, pulse-width를 write로 조절해 출력한다. 이 때 pulse-width는 0이면 0, 1.0이면 위에서 설정한 주기10ms로 출력하며 for에 duty가 0에서 1.0까지 커지는 구조로 반복해 점점 led밝기가 커지는 상태를 만든다.

결과적으로 위 구조를 통해 led밝기가 없는 상태에서 커지는 작업을 반복하게 된다.

1.3 하드웨어 구성 사진 첨부하기

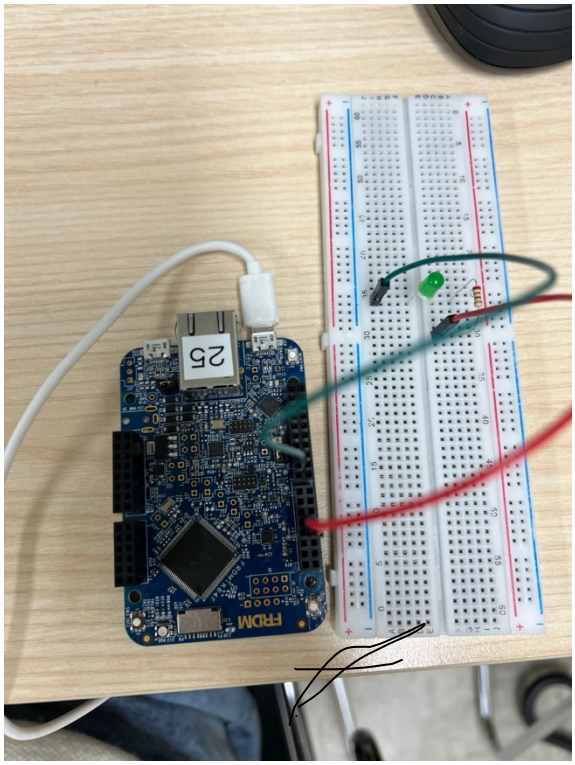

Figure 1 pwm-led 하드웨어 구성사진

1.4 프로그램 수행 사진/동영상(Youtube 링크) 첨부하기

https://youtu.be/65CWoLYsLJA

2 프로그램 pwm-music

2.1 프로그램 코드 쓰기

```
#include "mbed.h"
#include "pitches.h"

PwmOut buzzer(PTA1);        // buzzer = PTA1

int length = 75;

float frequency[] = {
    NOTE_E6, NOTE_E6, 0, NOTE_E6, 0, NOTE_C6, NOTE_E6, 0, NOTE_G6, 0, 0,
    0, NOTE_G5, 0, 0, 0, NOTE_C6, 0, 0, NOTE_G5, 0, 0, NOTE_E5, 0, 0,
    NOTE_A5, 0, NOTE_B5, 0, NOTE_AS5, NOTE_A5, 0, NOTE_G5, NOTE_E6, NOTE_G6,
    NOTE_A6, 0, NOTE_F6, NOTE_G6, 0, NOTE_E6, 0, NOTE_C6, NOTE_D6, NOTE_B5,
    0, 0, NOTE_C6, 0, 0, NOTE_G5, 0, 0, NOTE_E5, 0, 0, NOTE_A5, 0, NOTE_B5,
    0, NOTE_AS5, NOTE_A5, 0, NOTE_G5, NOTE_E6, NOTE_G6, NOTE_A6, 0, NOTE_F6,
    NOTE_G6, 0, NOTE_E6, 0, NOTE_C6, NOTE_D6, NOTE_B5, 0, 0
};

float beat[] = {
    12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,
    12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 9, 9, 9, 12,
```

```
    12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,
    12, 12, 12, 12, 12, 12, 12, 12, 12, 9, 9, 9, 12, 12, 12, 12, 12, 12, 12,
    12, 12, 12, 12, 12,
};

int main()
{
    while(true) {
        for(int i = 0; i <= length; i++) {
            if(frequency[i] == 0) {
                buzzer = 0.0;
            } else {
                buzzer.period(1.0 / frequency[i]);  // period = (1.0 / frequency)
                buzzer = 0.5;                       // duty cycle = 50%
            }
            thread_sleep_for(2500.0 / beat[i]);     // duration = (C / beat) ms
        }
    }
}
```

2.2 프로그램 작성 아이디어 혹은 이유 설명 쓰기

이 마리오 음악은 각각의 주파수와 길이(duration)에 해당하는 값을 배열에 담아 이 배열을 참고하여 buzzer로 출력한다.PwmOut라는 class에 buzzer라는 이름으로 생성한다. PTA1을 전달해 이 핀을 pwm출력 핀으로 설정한다. 그 후 main문 속 while문에 for문으로 들어가면 frequency[i]에 값을 참고해 0이면 buzzer를 write 0하고 예외에 경우는 frequency[i]에 역수를 period로 설정한 후 0.5를 write한다. 주파수의 역수는 파장이고, buzzer는 이 파장에 0.5크기에 pulse-width크기에 입력이 들어와야 그 주파수에 음을 출력한다. 따라서 이 구조로 음(frequency[i]에 값이 있는 경우)이 있으면 음을 출력하고 쉼표(frequency[i]이 0인 경우)이면 출력하지 않는다. 그 후 thread_sleep_for함수에 인자를 2500.0/beat[i]로 전달해 각 박자에 맞는 출력 duration을 가진다. 그리고 이 음에 출력 횟수는 전역변수 length에 담긴 값인75번 출력한다.

2.3 하드웨어 구성 사진 첨부하기

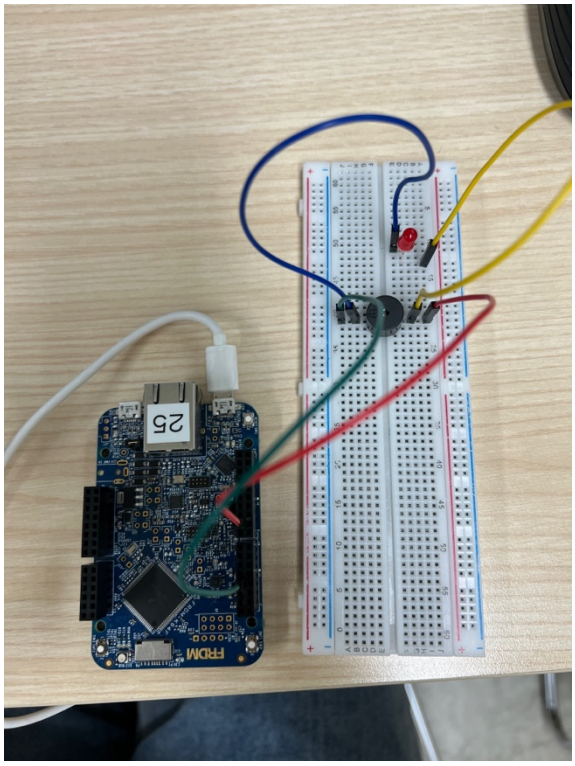

Figure 2 pwm-music 하드웨어 사진

2.4 프로그램 수행 사진/동영상(Youtube 링크) 첨부하기

https://youtu.be/lToSmlFrl1k

3. 프로그램 pwm-sg90-servo

3.1 프로그램 코드 쓰기

```
#include "mbed.h"

PwmOut servo(PTA1); // servo = PTA1
int main()
{
float w;
servo.period(20.0/1000.0); // period = 20 ms for SG90 Servo
while(true) {
// 500 to 2,500 us for degree 0 to 180
for(w = 500; w <= 2500; w = w + 100) {
servo.write(w / (20.0 * 1000.0));
thread_sleep_for(1000);
}
}
}
```

3.2 프로그램 작성 아이디어 혹은 이유 설명 쓰기

이 서브 모터는 주기가 20ms에서 500μs~2500μs Duty-cycle을 가지는 범위에 입력을 바탕으로 동작된다.

먼저 PwmOut라는 class에 객체를 servo라는 이름으로 생성한다. 그리고 PTA1로 이 핀에 모드를 pwm 출력으로 설정한다. 그 후 period라는 함수를 통해 주기를 20ms(=20/1000 s)로 설정한다. 그러면 for문에서 write함수에 w/(20*1000)이라는 인자를 전달해 w가 500에서 2500으로 증가하면 servo motor에 각도를 증가시키는 작업을 반복한다.(w/(20*1000)이라는 구조로 인자를 전달해 출력이 500μs~2500μs되게 했다.) 그리고 이 for문은 while문안에 있어 0에서 180도까지 변하는 과정을 계속 반복한다.

3.3 하드웨어 구성 사진 첨부하기

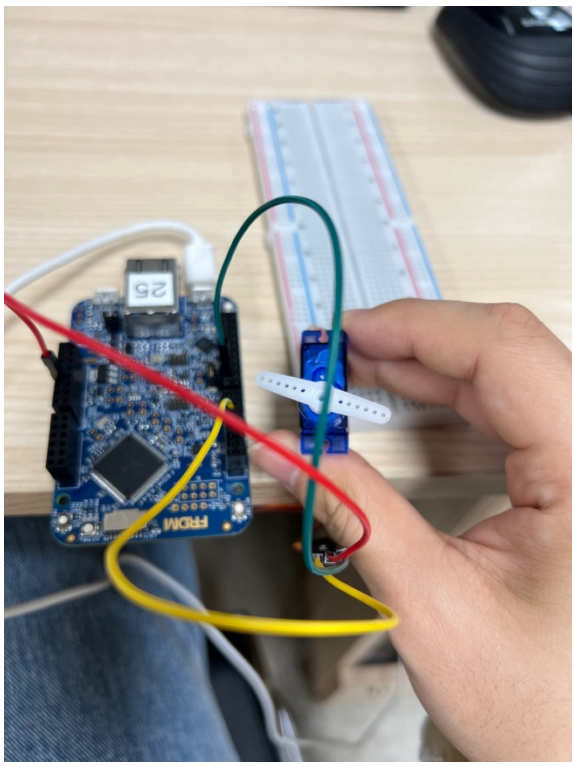

Figure 3 pwm-sg90-servo 하드웨어 사진

3.4 프로그램 수행 사진/동영상(Youtube 링크) 첨부하기

https://youtu.be/tPtX0kjdABs

끝.