강의명 : 임베디드 시스템
숙제 번호 : 5
숙제 제목 : Mbed application shield(Mbed 응용 쉴드)
학생 이름 : 오원묵(팀장), 황대은, 정우성(5팀)
학번 : 201810881, 201810897, 201810890

1. app-shield-lcd

1.1

```
#include "mbed.h"
#include "C12832.h"
C12832 lcd(D11, D13, D12, D7, D10);
int main()
{
    int count =0;
    lcd.cls();
    lcd.locate(0, 6);
    lcd.printf("Mbed Application Shield!");
    while(true) {
        lcd.locate(0, 16);
        lcd.printf("Counting: %d", count++);
        thread_sleep_for(1000);
    }
}
```

1.2

Mbed Application shield를 사용하여 원하는 문자 및 숫자를 LCD에 출력하는 프로그램을 작성하기 위해 우선 관련 라이브러리를 불러왔습니다. 이때 Mbed Application shield는 128 x 32 LCD를 가지므로 C12832라는 클래스를 활용했습니다. Mbed Application shield의 LCD는 커서의 개념을 사용하므로 화면에 문자 및 숫자를 출력하기 위해서는 커서의 위치를 지정해주는 것이 필요합니다. 따라서 Mbed Application Shield!라는 message와 초당 count가 나타나는 프로그램의 메인 함수를 작성할 때 cls 함수를 통해 화면을 초기화한 후 locate 함수를 사용하여 message가 출력되길 희망하는 위치로 커서를 옮겼습니다. 이후 printf 함수를 사용하여 message를 출력하도록 작성했습니다. 초당 count를 나타내기 위해서는 locate, printf 함수 및 반복문을 활용하여 1000ms 마다 count가 1씩 증가하여 출력되도록 프로그램을 작성했습니다.

1.3

하드웨어 구성사진 없음

1.4

링크

https://youtube.com/shorts/DVvr21uM2s8?feature=share

2. app-shield-rgb-led

2.1

```
#include "mbed.h"
#include "C12832.h"
C12832 lcd(D11, D13, D12, D7, D10);
PwmOut led_r(D5);
PwmOut led_g(D9);
int main()
{
    lcd.cls();
    lcd.locate(0, 6);
    lcd.printf("RGB LED!");
    while(true) {
        for(float f =0.0; f <1.0; f +=0.05) {
            led_r =1.0 - f;
            led_g =1.0;
            lcd.locate(0, 16);
            lcd.printf("Red=%.2f, Green=%.2f, Blue=%.2f", f, 0.0, 0.0);
            thread_sleep_for(10);
        }

        for(float f =0.0; f <1.0; f +=0.05) {
            led_r =1.0;
            led_g =1.0 - f;
            lcd.locate(0, 16);
            lcd.printf("Red=%.2f, Green=%.2f, Blue=%.2f", 0.0, f, 0.0);
            thread_sleep_for(10);
        }
    }
}
```

2.2

※ Mbed Application shield의 blue LED를 사용하면 프로그램이 정상적으로 작동되지 않아 blue LED는 제외하고 프로그램을 작성했습니다.

Mbed Application shield의 LED가 red, green 순서로 나타나며 색깔별로 밝기가 점점 밝아지고, 변화하는 밝기 값을 Mbed Application shield의 LCD에 나타내는 프로그램을 작성하기 위해서는 LED 밝기 변화를 위한 pulse-width modulation과 LCD 출력을 위한 C12832 클래스를 활용해야 합니다. 따라서 프로그램을 작성할 때 우선 LED_RED에 해당하는 D5, LED_GREEN에 해당하는 D9 핀에 연결된 2개의 PwmOut 클래스 형의 변수 led_r, led_g를 생성했습니다. 이후 프로그램의 메인 함수를 작성할 때 LCD에 RGB LED!라는 message가 출력되도록 cls 함수를 통해 화면을 초기화하고 locate, printf 함수를 사용했습니다. 이후 LED의 색깔 및 밝기의 변화가 반복되며 이에 따라 LCD에 변화하는 밝기 값이 출력되도록 while 반복문 내에 red, green 개수에 맞는 for 반복문이 나타나도록 프로그램을 작성했습니다. LED는 common anode 방식으로 연결되어 있고, pulse-width modulation 방식을 통해 밝기가 변화하므로 빨간색 LED가 나타나는 for 문을 작성할 때는 green은 off 상태가 되도록 1을 부여했고, red는 1-f를 부여하여 반복문 내에서 f의 값이 증가함에 따라 LED가 밝아지도록 프로그램을 작성했습니다. 이때 f는 0부터 시작하여 1보다 작을 때까지 값이 0.05씩 증가하는 실수이며 f 값은 밝기 값을 나타낸다고 할 수 있습니다. 따라서 for 문을 작성할 때 변화하는 밝기 값이 LCD에 나타나기 위해 locate 함수를 사용해 커서의 위

치를 옮기고, printf 함수를 사용해 f 값이 출력되도록 프로그램을 작성했습니다. 초록색 LED가 나타나는 for문도 이와 같은 방식으로 프로그램을 작성했습니다.

2.3
하드웨어 구성사진 없음

2.4
링크
https://youtube.com/shorts/1jIYdLJFwpc?feature=share

3. app-shield-joystick

3.1

```
#include "mbed.h"
#include "C12832.h"
C12832 lcd(D11, D13, D12, D7, D10);
DigitalIn up(A2);
DigitalIn down(A3);
DigitalIn left(A4);
DigitalIn rite(A5);
DigitalIn center(D4);
int main()
{
    lcd.cls();
    lcd.locate(0, 6);
    lcd.printf("Move Joystick!");
    while(true) {
        if(up) {
            lcd.locate(0, 16);
            lcd.printf("Up     ");
        }
        if(down) {
            lcd.locate(0, 16);
            lcd.printf("Down     ");
        }
        if(left) {
            lcd.locate(0, 16);
            lcd.printf("Left     ");
        }
        if(rite) {
            lcd.locate(0, 16);
            lcd.printf("Right     ");
        }
        if(center) {
            lcd.locate(0, 16);
            lcd.printf("Center     ");
        }
        thread_sleep_for(200);
    }
}
```

3.2

Mbed Application shield의 5-Way joystick은 위, 아래, 왼쪽, 오른쪽 4방향으로 움직일 수 있고, 가운데를 누를 수 있습니다. 이를 통해 연결된 보드에 디지털 입력 신호를 보낼 수 있습니다. Mbed Application shield의 5-Way joystick을 위, 아래, 왼쪽, 오른쪽으로 움직이거나 가운데를 눌러 이에 대응하는 결과를 Mbed Application shield의 LCD에 출력하는 프로그램을 작성하기 위해서는 DigitalIn 클래스와 C12832 클래스를 활용해야 합니다. 따라서 프로그램을 작성할 때 SW_UP에 해당하는 A2, SW_DOWN에 해당하는 A3, SW_LEFT에 해당하는 A4, SW_RIGHT에 해당하는 A5, SW_CENTER에 해당하는 D4 핀에 연결된 5개의 DigitalIn 클래스 형의 변수 DigitalIn up, DigitalIn down, DigitalIn left, DigitalIn rite, DigitalIn center를 생성했습니다. 메인 함수를 작성할 때 LCD에 Move Joystick!이라

는 message를 나타내기 위해 cls 함수를 통해 화면을 초기화하고, locate 함수를 통해 커서의 위치를 옮긴 후 printf 함수를 사용하여 message를 출력했습니다. 이후 반복문과 조건문을 활용하여 5-Way joystick을 위, 아래, 왼쪽, 오른쪽 방향으로 움직이거나 가운데를 누를 때마다 이에 대응하는 결과가 LCD에 출력되도록 프로그램을 작성했습니다. 이를 위해 up, down, left, rite, center에 해당하는 5개의 조건문을 작성하여 현재 입력으로 들어오는 방향에 해당하는 조건문만 실행되도록 프로그램을 작성했습니다. 이때 각 조건문을 작성할 때 locate 함수와 printf 함수를 사용하여 LCD에 출력 결과가 나타나도록 프로그램을 작성했습니다.

3.3
하드웨어 구성사진 없음

3.4
링크
https://youtube.com/shorts/l10b8sVL90I?feature=share

4. app-shield-pot

4.1

```
#include "mbed.h"
#include "C12832.h"

C12832 lcd (D11 , D13 , D12 , D7 , D10 ); // lcd = (MOSI, SCK, RESET, A0, nCS)
AnalogIn pot1 (A0 ); // pot1 = A0
AnalogIn pot2 (A1 ); // pot2 = A1
PwmOut led_r (D5 ); // led_r = (Red LED)
PwmOut led_g (D9 ); // led_g = (Green LED)

int main ()
{
  lcd .cls ();
  lcd .locate (0 , 6 );
  lcd .printf ("Rotate Potentiometers!");
  while (true ) {
    lcd .locate (0 , 16 );
    lcd .printf ("Pot1=%4.2f, Pot2=%4.2f", pot1 .read (), pot2 .read ());
    led_r =1.0 -pot1 .read (); // pot1 -> Red LED
    led_g =1.0 -pot2 .read (); // pot2 -> Green LED
    thread_sleep_for (200 );
  }
}
```

4.2

Mbed application shield의 2개의 가변 저항을 이용해 저항을 변화시킴으로써 Mbed application shield의 LED red, LED green의 밝기를 변화시키고, 밝기 값을 Mbed application shield의 LCD에 나타내기 위해서는 AnalogIn 클래스, PwmOut 클래스, C12832 클래스를 활용해야 합니다. AnalogIn 클래스는 가변 저항을 회전시킴에 따라 달라지는 전압을 읽어오기 위해 사용됩니다. PwmOut 클래스는 LED 밝기의 변화를 나타내기 위해 사용됩니다. C12832 클래스는 LCD에 밝기 값을 나타내기 위해 사용됩니다. 따라서 프로그램을 작성할 때 2개의 가변 저항을 통해 2개의 LED 밝기의 변화를 나타내기 위해 AnalogIn 클래스 형의 변수 pot1, pot2와 PwmOut 클래스 형의 변수 led_r, led_g를 생성했습니다. 이후 메인 함수를 작성할 때 cls 함수를 통해 화면을 초기화하고, locate 함수를 통해 커서의 위치를 옮긴 후 printf 함수를 통해 Rotate Potentiometers!라는 message를 LCD에 출력하도록 프로그램을 작성했습니다. 이후 가변 저항을 회전시킴에 따라 지속적으로 LED red, LED green의 밝기가 변하고, 밝기 값을 LCD에 나타내기 위해 반복문을 사용했습니다. 이때 옴의 법칙과 LED가 common anode 방식으로 연결되어있는 특징을 활용해 프로그램을 작성했습니다. 이와 같은 특징을 활용하여 LED red의 밝기인 led_r = 1.0-pot1.read()로 나타냈을 때 pot1.read()의 값이 클수록 더 밝은 빛을 내며, pot1.read()의 값은 밝기 값을 나타낸다고 할 수 있습니다. 이때 pot1.read()는 가변 저항을 회전시킴에 따라 변화하는 전압의 percentage로

0과 1 사이의 실수를 의미합니다. 따라서 반복문을 작성할 때 locate, printf 함수를 사용하여 pot1.read(), pot2.read()를 출력하여 LCD에 LED red, LED green의 밝기 값을 나타냈습니다. 또한, led_r = 1.0-pot1.read(), led_g = 1.0-pot2.read()를 통해 LED에 밝기 변화가 나타나도록 프로그램을 작성했습니다.

4.3
하드웨어 구성사진 없음

4.4
링크
https://youtube.com/shorts/AhKuwXtpVgU?feature=share

5. app-shield-speaker

5.1

```
#include "mbed.h"
#include "C12832.h"
#include "pitches.h"

C12832 lcd (D11 , D13 , D12 , D7 , D10 ); // lcd = (MOSI, SCK, RESET, A0, nCS)
PwmOut speaker (D6 ); // speaker = D6

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 ()
{
  lcd .cls (); lcd .locate (0 , 6 ); lcd .printf ("Listen to speaker!");
  lcd .locate (0 , 16 ); lcd .printf ("Mario~~~");

  while (true ) {
   for (int i =0 ; i <= length; i ++) {
    if (frequency [i] ==0 ) {
     speaker =0.0 ;
    } else {
     speaker .period (1.0 /frequency [i]);  // period = (1.0 / frequency)
     speaker =0.5 ;          // duty cycle = 50%
```

```
      }
      thread_sleep_for (2500.0 /beat [i]);
    }
  }
}
```

5.2

Mbed application shield의 LCD에 message를 출력하고, Mbed application shield의 speaker를 통해 음악을 연주하는 프로그램을 작성하기 위해서는 C12832 클래스와 PwmOut 클래스를 활용해야 합니다. C12832 클래스는 LCD 출력을 위해 사용되며, PwmOut 클래스는 pulse-width modulation을 통해 음악을 연주할 때 필요한 period, duty-cycle을 활용하기 위해 사용됩니다. 음악을 연주하기 위해서는 그 음악을 구성하는 각 음의 주파수와 길이가 필요합니다. 따라서 프로그램을 작성할 때 이를 2개의 배열로 나타냈습니다. 이후 메인 함수를 작성할 때 cls 함수를 통해 화면을 초기화하고, locate 함수, printf 함수를 사용해 Listen to speaker!라는 message와 Mario~~~라는 message가 LCD에 출력되도록 했습니다. 이후 반복문을 사용하여 각 음이 순서대로 Mbed application shield의 speaker를 통해 출력되도록 했습니다. pulse-width modulation을 위해서는 period와 duty-cycle이 필요합니다. 주기 = 1/주파수이므로 음악의 주파수를 통해 period를 구하고, duty-cycle은 50%가 되도록 반복문을 작성했습니다. 이때 조건문을 활용하여 주파수 배열에서 주파수가 0인 부분은 duty-cycle을 0%로 만들어 speaker를 통해 출력되지 않도록 프로그램을 작성했습니다.

5.3

하드웨어 구성사진 없음

5.4

링크

https://youtube.com/shorts/XW-TnDe4WGo?feature=share

6. app-shield-3-axis

6.1

```
#include "mbed.h"
#include "C12832.h"
#include "MMA7660.h"
C12832 lcd(D11, D13, D12, D7, D10); // lcd = (MOSI, SCK, RESET, A0, nCS)
MMA7660 MMA(D14, D15); // MMA = (I2C_SDA, I2C_SCL)
int main()
{
  lcd.cls();
  lcd.locate(0, 6);
  lcd.printf("Mbed 3-Axis accelerometer!");
  while(true) {
    lcd.locate(0, 16);
    lcd.printf("x=%.2f y=%.2f z=%.2f", MMA.x(), MMA.y(), MMA.z());
    thread_sleep_for(100);
  }
}
```

6.2

Mbed application shield의 LCD에 message와 Mbed application shield의 3축 가속도계의 변화하는 입력값을 출력하는 프로그램을 작성하기 위해서는 C12832 클래스와 MMA7660 클래스를 활용합니다. C12832 클래스는 LCD에 출력을 위해 사용되며, MMA7660 클래스는 x축, y축, z축의 입력값을 읽어오기 위해 사용됩니다. 3축 가속도계는 i2c 방식을 통해 보드와 통신합니다. 따라서 프로그램을 작성할 때 SDA는 D14, SCL은 D15 핀에 연결하여 MMA7660 클래스 형의 변수 MMA를 생성했습니다. 메인 함수에서 우선 cls 함수를 통해 화면을 초기화하고, locate, printf 함수를 통해 Mbed 3-Axis accelerometer!라는 message를 출력할 수 있도록 프로그램을 작성했습니다. 이후 반복문을 통해 지속적으로 3축 가속도계의 변화하는 입력값을 읽어와 LCD에 출력하도록 프로그램을 작성했습니다. 이때 MMA7660 클래스의 함수인 x(), y(), z()를 사용하여 x축, y축, z축 각각의 값을 읽어와 출력하도록 했습니다.

6.3

하드웨어 구성사진 없음

6.4

링크

https://youtube.com/shorts/IZyoGzB07VE?feature=share

7. app-shield-temp

7.1

```
#include "mbed.h"
#include "C12832.h"
#include "LM75B.h"
C12832 lcd(D11, D13, D12, D7, D10); // lcd = (MOSI, SCK, RESET, A0, nCS)
LM75B temp(D14, D15); // temp = (SDA, SCL)
int main()
{
  lcd.cls();
  lcd.locate(0, 6);
  lcd.printf("Current temperature!");
  while(true) {
    lcd.locate(0, 16);
    lcd.printf("%5.2f degree Celsius", temp.read());
    thread_sleep_for(1000);
  }
}
```

7.2

Mbed application shield의 LCD에 message와 Mbed application shield의 온도 센서로 측정한 현재 섭씨 온도를 출력하는 프로그램을 작성하기 위해서는 C12832 클래스와 LM75B 클래스를 활용해야 합니다. C12832 클래스는 LCD 출력을 위해 사용되며, LM75B 클래스는 현재 온도를 읽어오기 위해 사용됩니다. Mbed application shield의 온도 센서는 보드와 i2c 방식으로 통신합니다. 따라서 프로그램을 작성할 때 SDA는 D14, SCL은 D15 핀에 연결하여 LM75B 클래스 형의 변수 temp를 생성했습니다. Current temperature!이라는 message를 LCD에 나타내기 위해 메인 함수를 작성할 때 cls 함수를 통해 화면을 초기화하고, locate 함수를 통해 커서의 위치를 옮긴 후 printf 함수를 통해 출력하도록 작성했습니다. 이후 반복문을 통해 1초마다 현재 온도를 읽어와 LCD에 출력하도록 했습니다. 이때 LM75B 클래스의 함수 read()를 통해 섭씨 온도를 읽어올 수 있도록 반복문을 작성했습니다.

7.3

하드웨어 구성사진 없음

7.4

링크

https://youtube.com/shorts/MYBIZAblYsU?feature=share