6 주차 실습 레포트 제출

2018125020 류호원(소프트웨어학과)

Task06-B,A

코드 제출

06-1

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#define SEALEVELPRESSURE_HPA (1013.25)
Adafruit_BME280 bme; // I2C
int delayTime;
void setup() {
bool status;
Serial.begin(115200);
Serial.println(F("BME280 test"));
// default settings
status = bme.begin(0x76); // bme280 I2C address = 0x76
if (!status) {
Serial.println("Could not find a valid BME280 sensor, check wiring, address, sensor ID!");
Serial.print("SensorID was: 0x"); Serial.println(bme.sensorID(),16);
Serial.print(" ID of 0xFF probably means a bad address, a BMP 180 or BMP 085₩n");
Serial.print(" ID of 0x56-0x58 represents a BMP 280,₩n");
Serial.print(" ID of 0x60 represents a BME 280.₩n");
Serial.print(" ID of 0x61 represents a BME 680.₩n");
```

```
while (1) delay(10);
}
Serial.println("-- Default Test --");
delayTime = 1000;
Serial.println();
}
void loop() {
printValues();
delay(delayTime);
}
void printValues() {
Serial.print("Temperature = ");
Serial.print(bme.readTemperature());
Serial.println(" *C");
Serial.print("Pressure = ");
Serial.print(bme.readPressure() / 100.0F);
Serial.println(" hPa");
Serial.print("Approx. Altitude = ");
Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
Serial.println(" m");
Serial.print("Humidity = ");
Serial.print(bme.readHumidity());
Serial.println(" %");
Serial.println();
}
```

06-2

코드 제출

```
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
/* sw spi
#define BME_SCK 13
#define BME_MISO 12
#define BME_MOSI 11 */
#define BME_CS 5 // cs for esp32 vspi
#define SEALEVELPRESSURE_HPA (1013.25)
//Adafruit_BME280 bme; // I2C
Adafruit_BME280 bme(BME_CS); // hardware SPI
//Adafruit_BME280 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK); // software SPI
int delayTime;

void setup() {
Serial.begin(115200);
Serial.println(F("BME280 test"));
bool status;
// default settings
status = bme.begin();
if (!status) {
Serial.println("Could not find a valid BME280 sensor, check wiring, address, sensor ID!");
Serial.print("SensorID was: 0x"); Serial.println(bme.sensorID(),16);
```

```
Serial.print(" ID of 0xFF probably means a bad address, a BMP 180 or BMP 085₩n");
Serial.print(" ID of 0x56-0x58 represents a BMP 280,₩n");
Serial.print(" ID of 0x60 represents a BME 280.₩n");
Serial.print(" ID of 0x61 represents a BME 680.₩n");
while (1) delay(10);
}
Serial.println("-- Default Test --");
delayTime = 1000;
Serial.println();
}
void loop() {
printValues();
delay(delayTime);
}
void printValues() {
Serial.print("Temperature = ");
Serial.print(bme.readTemperature());
Serial.println(" *C");
Serial.print("Pressure = ");
Serial.print(bme.readPressure() / 100.0F);
Serial.println(" hPa");
Serial.print("Approx. Altitude = ");
Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
Serial.println(" m");
Serial.print("Humidity = ");
Serial.print(bme.readHumidity());
```

```
Serial.println(" %");
Serial.println();
}
```

06-3

코드 제출

```
#include <EEPROM.h>
// define the number of bytes you want to access
#define EEPROM_SIZE 1
const int buttonPin = 15; // the number of the pushbutton pin
const int ledPin = 16; // the number of the LED pin
// Variables will change:
int ledState = HIGH; // the current state of the output pin
int buttonState; // the current reading from the input pin
int lastButtonState = LOW; // the previous reading from the input pin
// the following variables are unsigned longs because the time, measured in
// milliseconds, will quickly become a bigger number than can be stored in an int.
unsigned long lastDebounceTime = 0; // the last time the output pin was toggled
unsigned long debounceDelay = 50; // the debounce time; increase if the output flickers
void setup() {
Serial.begin(115200);
// initialize EEPROM with predefined size
EEPROM.begin(EEPROM_SIZE);
pinMode(buttonPin, INPUT);
pinMode(ledPin, OUTPUT);
// read the last LED state from flash memory
```

```
ledState = EEPROM.read(0);
// set the LED to the last stored state
digitalWrite(ledPin, ledState);
}
void loop() {
// read the state of the switch into a local variable:
int reading = digitalRead(buttonPin);
// check to see if you just pressed the button
// (i.e. the input went from LOW to HIGH), and you've waited long enough
// since the last press to ignore any noise:
// If the switch changed, due to noise or pressing:
if (reading != lastButtonState) {
// reset the debouncing timer
lastDebounceTime = millis();
}
if ((millis() - lastDebounceTime) > debounceDelay) {
// whatever the reading is at, it's been there for longer than the debounce
// delay, so take it as the actual current state:
// if the button state has changed:
if (reading != buttonState) {
buttonState = reading;
// only toggle the LED if the new button state is HIGH
if (buttonState == HIGH) {
ledState = !ledState;
}
}
```

```
}
// save the reading. Next time through the loop, it'll be the lastButtonState:
lastButtonState = reading;
// if the ledState variable is different from the current LED state
if (digitalRead(ledPin)!= ledState) {
Serial.println("State changed");
// change the LED state
digitalWrite(ledPin, ledState);
// save the LED state in flash memory
EEPROM.write(0, ledState);
EEPROM.commit();
Serial.println("State saved in flash memory");
}
}
```

Task06-C

코드 + 주석 설명 제출

```
#include <EEPROM.h>
// 핀세팅
const int ledChannel = 0;
const int resolution = 8;
const int buzPin = 23;
const int duty = 128;

// 변수 선언
```

```
int vNote=0, vDur=0, i ,eep;
const int dDur = 250; // default duration

// notes
//enum Notes {C3=0, CS3, D3, DS3, E3, F3};
int nFrq[] = {262, 277, 294, 311, 330, 349, 370, 392, 415, 440, 466, 494, 523};
int nDur[] = { 2000, 1500, 1000, 750, 500, 375, 250 };

void playNote(int note, int dur) {
if (note == -1) {
// 소리 묵음
      ledcSetup(ledChannel, 0, resolution);
      ledcWrite(ledChannel, 0);
   }
   else {
      ledcSetup(ledChannel, nFrq[note], resolution);
      ledcWrite(ledChannel, duty);
   }
   Serial.println(String(note)+","+String(dur));
   delay(nDur[dur]);
}

void setup() {
   Serial.begin(115200);
   EEPROM.begin(512);
   // attach the channel to the GPIOs
```

```
  ledcAttachPin(buzPin, ledChannel);
  eep=0;
  i=1;
  if(EEPROM.read(0)=='l'){
// eeprom(0)에 1이 있으면 eeprom(1~512)까지 번호를 읽어 들여서 note와 dur을 나누고
//playnote로 eeprom에 저장되어 있는 숫자들을 연주한다
    for(int i = 1; i < 512; i+=2){
      int eepnote = EEPROM.read(i);
      int eepdur = EEPROM.read(i+1);
      playNote(eepnote, eepdur);
    }
  }
}

void loop(){
  Serial.println("0");
  if (Serial.available() > 0) {
    vNote = Serial.read();
    if(vNote=='<'){
// 읽어들인 값이 '<' 라면 eepro(0)에 1을 넣고 값을 저장하기 위한 플레그로 eep에 1을 넣는다
      vNote = Serial.read();
      eep=1;
      EEPROM.write(0, 'l');
    }
    if (Serial.available() > 0) {
      vDur = Serial.read();
```

```
        if (vDur <= '6' && vDur >= '0')
          vDur -= '0';
        else vDur = dDur;
        if (vNote <= '9' && vNote >= '0')
          vNote -= '0';
        else if (vNote <= 'c' && vNote >= 'a')
          vNote = vNote - 'a' + 10;
        else if(vNote == ',')
          vNote = -1; // rest
        else if(vNote == '>')
          // > 는 연주하지 않는다
          vNote = -1;
        playNote(vNote, vDur);
        if(eep==1){
```

앞에서 eep가 1인 경우는 eeprom(0)이 1인 경우 임으로 note와 dur을 eeprom(1)번부터 차례로 저장한다.

```
          EEPROM.write(i, vNote);
          EEPROM.write(i+1, vDur);
          i+=2;
          EEPROM.commit();
        }
      }
    }
}
```