**마이크로프로세서 최종 프로젝트**

**[MP3 플레이어]**

담당 교수 : 안영신

전북대학교 소프트웨어공학과 201911067 김준기

**프로젝트 소개**

이 프로젝트는 저장된 노래를 재생하고 가사를 출력하는 MP3 player이다. 단순해 보이지만 피에조 부저로 음을 출력하고, 텍스트 LCD로 노래 가사를 출력하는 동시에 버튼으로 입력받아야 한다. 구현 방법에 따라 불안정한 시스템이 될 수도 있고, 안정적인 시스템이 될 수도 있다.

이 프로젝트는 시스템이 얼마나 안정적으로 작동하는가(안정성), 사용자가 얼마나 사용하기 편한가(사용자 편의성), 소프트웨어적으로 유지보수하기 용이한가에 중점을 두어 개발을 하였다.

**접근 방법**

1. MP3는 단순해 보이지만 피에조 부저로 음을 출력하고, 노래 가사를 출력하는 동시에 버튼으로 입력을 받아야 한다. 이 프로젝트는 시스템이 얼마나 안정적으로 작동하는가(안정성), 사용자가 얼마나 사용하기 편한 가(사용자 편의성)에 중점을 두어 개발을 하였다.
2. Keil studio를 이용해 ARM Cortext-M4 시스템을 제어
3. LCD1602 I2C 모듈, 피에조 부저, 버튼(2개)을 사용
4. Polling 방식을 사용하며, 여러 장치를 동시에 제어하기 위해 Timer을 사용

**입출력**

* 버튼 1: 사용자의 입력을 받아 다음 노래 선택, 노래 일시 정지 기능을 함
* 버튼 2: 사용자의 입력을 받아 노래 재생, 노래 중지 기능을 함
* 피에조 부저: 노래를 출력
* 텍스트 LCD: 노래 제목, 가사를 출력

**처리**

1. Polling 방식으로 버튼, 피에조 부저, LCD를 동시 제어
2. LCD의 윗줄은 MP3 상태를 출력(menu, playing, pause), 아래줄은 노래 제목, 가사를 출력
3. Music 객체에서 Timer를 사용해 부저로 노래를 출력. 음높이에 따른 주파수를 getFrequency() 함수로 계산
4. 노래 하나를 음높이, 가사, 박자 3개의 배열로 구성. 음높이는 음계와 옥타브 정보가 있으며 스트링으로 저장(“C4” - 4옥타브 도). 노래 전체 가사를 박자 단위로 분리해서 스트링 배열로 저장. 박자는 정수 타입이며 이 시스템에서 박자의 최소 단위로 8분음표를 1로 정의.
5. 노래가 출력되는 동시에 LCD에 가사를 출력. 가사를 입력할 공간이 남아있으면 이전 가사에 이어서 출력을 하고, 만약 가사가 공간을 초과한다면 이전 가사를 지우고 현재 가사를 출력
6. 만약 노래 제목이 LCD가 출력할 수 있는 최대 길이(16)보다 크다면 Timer를 사용해 주기적으로 제목을 왼쪽으로 하나씩 이동하여 사용자가 모든 제목을 볼 수 있도록 처리
7. 버튼을 누를 때마다 wait()을 사용해 디바운싱 작업을 함. 이때 코드 상에서 wait()을 사용자 편의성을 고려하여 위치시킴.

**프로젝트 동기**

Nucleo 보드와 여러 센서, 모듈들을 학습하면서 눈으로 보이고, 귀로 들리는(특히 음계를 출력할 수 있는) 텍스트 LCD와 피에조 부저에 관심이 있었다. 따라서 이 둘을 적절히 조합할 수 있는 MP3를 프로젝트 주제로 선정하였다. MP3는 단순하게 보이지만, 음향 신호를 출력하고, 가사를 출력하고, 사용자의 입력을 받는 작업을 동시에 진행해야 한다. 이에 대한 여러 구현 방법이 있겠지만, 나는 어떻게 하면 사용자 친화적이고 안정적인 임베디드 시스템을 개발할 수 있을 가에 중점을 두어 개발을 진행하였다.

**하드웨어 분석**

1. 부품이(가) 표시된 사진

   중간 신뢰도로 자동 생성된 설명피에조 부저 : 압전 물질로 소리를 발생시키는 장치이다. 압전 물질이란 압력을 가했을 때 전압이 발생되고, 전압을 가했을 때 압력이 생기는 물질이다. 피에조 부저는 압전 물질에 전압을 가해 진동을 시켜, 판과 부딪혀 소리가 나게 하는 원리이다. 여기서 피에조 부저는 디지털 신호를 이용하여 아날로그 신호를 표현하는 PWM 핀에 연결해 제어하고 있다. PWM은 펄스 폭을 변조하여 전압이나 전류를 제어함으로써 아날로그 값을 표현한다. PWM 핀은 듀티 사이클과 주기로 제어한다. 듀티 사이클은 0~1(0%~100%)로 조절할 수 있다. 만약 듀티 사이클이 0.5라면 주기의 절반 동안 신호가 활성화되고, 나머지 절반 동안 비활성화된다.   
   피에조 부저와 연결하는 핀은 다음과 같다.   
   - (+) : 5V  
   - other : D5(PWM)
2. Text LCD : LCD(Liquid Crystal Display) 기술을 활용하여 문자를 표시하는 장치이다. 여기서 사용되는 Text LCD는 LCD1602로 16칸×2줄로 문자를 표시할 수 있다. 보통 Text LCD는 8개의 핀으로 제어를 해야 하지만 여기서 사용하는 모듈은 I2C 모듈과 연결되어 있어 더 적은 4개의 핀으로 제어할 수 있다. I2C란 직렬 통신 프로토콜로 두가지 신호선 SDA와 SCL로 데이터를 전송한다. SDA (Serial Data)는 I2C 통신에서 실제 데이터를 전송하는 데 사용되는 신호 선이다. 그리고 SCL (Serial Clock)은 I2C 통신에서 클럭 신호를 제공하는 선으로, 모든 디바이스 간에 데이터 전송의 타이밍을 동기화하는 역할을 한다. 연결하는 핀은 다음과 같다.  
   - GND : GND  
   - VCC : 5V  
   - SDA : D14(SDA)  
   - SCL : D15(SCL)  
     
   Text LCD를 쉽게 제어하기 위해선 라이브러리가 필요하다. 다음은 케일 스튜디오를 통해 라이브러리를 적용시키는 방법이다.

* 케일 스튜디오 > file > Add Mbed Library … > 아래 링크 입력
* 링크(http://os.mbed.com/users/oscarvzfz/code/TextLCD/)

TextLCD.h를 사용한 기본적인 코드는 다음과 같다. 이는 해당 라이브러리를 작성한 사람의 Sample 코드를 참고하였다.

|  |
| --- |
| include "mbed.h"  #include "TextLCD.h“  I2C i2c\_lcd(D14,D15); // SDA, SCL  TextLCD\_I2C lcd(&i2c\_lcd, 0x4E, TextLCD::LCD16x2, TextLCD::HD44780);  int main()  {  lcd.setMode(TextLCD::DispOn); //DispOff, DispOn  lcd.setBacklight(TextLCD::LightOff);//LightOff, LightOn  lcd.setCursor(TextLCD::CurOff\_BlkOff);//CurOff\_BlkOff, CurOn\_BlkOff, CurOff\_BlkOn, CurOn\_BlkOn  lcd.printf("Oscar de Jesus \n Vasquez");  } |

1. 스위치 : 스위치는 디지털 입력을 제어하는 데 사용되는 장치이다. 스위치는 풀업(Pull-Up)과 풀다운(Pull-Down) 방식 중 하나로 사용할 수 있다. 풀업이란 디지털 입력 핀을 기본적으로 HIGH 상태로 유지하는 방식이고, 스위치가 닫혔을 때 LOW 상태가 된다. 반대로 풀다운이란 디지털 입력 핀을 기본적으로 LOW 상태로 유지하고, 스위치가 닫혔을 때 HIGH 상태가 된다. 여기서는 풀다운 방식을 사용한다. 스위치는 4개의 핀이 있는데 대각선 또는 양옆으로 GND와 디지털 핀을 연결하여 사용할 수 있다. 이 프로젝트에서는 두개의 스위치를 사용하는 데 5V 그리고 각각 D3, D4에 연결하였다.   
   
2. 키트와 결선 방법  
   텍스트, 전자제품, 전자 공학, 전기 배선이(가) 표시된 사진

   자동 생성된 설명

**코드 분석**

전체 코드 : https://github.com/ijun17/Mbed-MP3-Player

1. Main.cpp : 아래는 이 프로젝트의 main.cpp파일이다. 기본적으로 “mbed.h” 라이브러리를 포함시키고 있고, 아래에 3개의 헤더 파일이 있다. “Music.h”는 노래 정보를 저장하고, 피에조 부저를 제어하는 클래스를 작성되어 있는 헤더파일이다. “MusicSample.h”는 예시 노래 정보를 담고 있는 파일로 “떴다떴다 비행기”, “반짝반짝 작은별”의 영문 버전의 정보가 작성되어 있다. “MP3Player.h”는 전체 시스템을 제어하는 MP3Player 클래스가 작성된 헤더파일이다.   
   아래에 main 함수는 샘플 노래 정보로 2개의 Music 객체를 만들어 MP3Player 객체에 저장하고, MP3를 실행하는 코드이다.

|  |
| --- |
| #include "mbed.h"  #include "Music.h"  #include "MusicSample.h"  #include "MP3Player.h"  int main()  {      Music music1(name1,pitchs1,lyrics1,beats1,length1);      Music music2(name2,pitchs2,lyrics2,beats2,length2);      MP3Player mp3(D5, D14, D15, D3, D4);      mp3.addMusic(music1);      mp3.addMusic(music2);      mp3.start();  } |

1. MusicSample.h : 앞서 설명했듯 예시 노래 정보를 담고 있는 파일이다. Music 객체를 생성하기 위한 노래 제목, 노래 가사, 음계, 박자, 배열 길이가 작성되어 있다.

|  |
| --- |
| #include <string>  using namespace std;  //Sample1  string name1="Plane In The Sky";  string lyrics1[25]={"Ri","sing ","Ri","sing ","Air","pl","ane ","fly ","a","way ","fly ","a","way ",                      "High","er ","High","er ","fly ","a","way ","Ou","r","air","pl","ane"};  string pitchs1[25]={"E4","D4","C4","D4","E4","E4","E4","D4","D4","D4","E4","E4","E4",                      "E4","D4","C4","D4","E4","E4","E4","D4","D4","E4","D4","C4"};  int beats1[25]={3,1,2,2, 2,2,4, 2,2,4, 2,2,4, 3,1,2,2, 2,2,4, 2,2,3,1, 6};  int length1=25;  //Sample2  string name2="Twinkle, Twinkle, Little Star";  string lyrics2[42]={"Twin","kle, ","Twin","kle, ","lit","tle ","star, ","how ","I ","won","der, ","what ","you ","are! "                      ,"Up ","a","bove ","the ","world ","so ","high, ","like ","a ","dia","mond ","in ","the ","sky. "                      ,"Twin","kle, ","Twin","kle, ","lit","tle ","star, ","how ","I ","won","der, ","what ","you ","are! "};  string pitchs2[42]={"C4", "C4", "G4", "G4", "A4", "A4", "G4", "F4", "F4", "E4", "E4", "D4", "D4", "C4",                      "G4", "G4", "F4", "F4", "E4", "E4", "D4", "G4", "G4", "F4", "F4", "E4", "E4", "D4",                      "C4", "C4", "G4", "G4", "A4", "A4", "G4", "F4", "F4", "E4", "E4", "D4", "D4", "C4"};  int beats2[42]={2,2,2,2,2,2,4,2,2,2,2,2,2,4,                  2,2,2,2,2,2,4,2,2,2,2,2,2,4,                  2,2,2,2,2,2,4,2,2,2,2,2,2,4};  int length2=42; |

1. Music.h : 노래 정보를 저장하고 피에조 부저를 제어해 노래를 재생하는 코드이다. 양이 많아 Music 클래스의 함수 중 주파수를 계산하는 함수와 피에조 부저를 다루는 함수만 설명하겠다.  
   아래는 주파수를 계산하는 getFrequency() 함수이고 그 외에는 계산을 보조하는 배열/함수이다. 여기서 주파수를 계산하는 이유는 음높이에 따라 정해진 주파수가 있기 때문이다. 이 음높이에 따른 주파수는 pitches.h 라이브러리에서 제공을 하지만, pitches에서 제공하는 음계들은 정수로 소수점을 무시하고 있다. 따라서 옥타브와 음계를 더 정확히 표현하기 위해 주파수를 계산하는 getFrequency()로 함수를 만들었다.

|  |
| --- |
| static const int PITCH[7];//{9,11,0,2,4,5,7}={A,B,C,D,E,F,G}      //pitchs[i]에서 옥타브 추출      int getOctave(int i){return pitches[i][pitches[i].length()-1]-'0';}      //pitchs[i]에서 음계 추출      int getPitch(int i){return PITCH[pitches[i][0]-'A'] + (pitches[i][1]=='#');}      // 옥타브와 음계로 주파수 계산      float getFrequency(int i) {          const float baseFreq = 32.70; //기준 주파수(1옥타브 도)          return baseFreq \* pow(2, getOctave(i)+getPitch(i)/12.0); // 주파수 반환      } |

아래 그림은 옥타브와 음계에 따른 주파수를 정리한 표이다. 아래 표를 보면 알 수 있듯이 옥타브가 한단계 높아질수록 주파수가 2배가 된다. 그리고 각 음계는 2의 1/12승 배로 증가한다.



그 다음은 주파수를 이용해 피에조 부저로 음향 신호를 방생시키는 update() 함수이다. 이 함수는 MP3Player가 while 루프를 돌며 실행된다. 만약 timer가 일정 시간(다음 음계를 출력할 시간)이 되었다면, 피에조 부저의 주기를 변화시킨다. 앞서 피에조 부저는 듀티 사이클과 주기로 제어한다고 하였다. 듀티 사이클은 ‘=’ 연산자로 설정할 수 있고, period 함수로 주기를 설정할 수 있다. 여기서 주기는 1/주파수이다.

|  |
| --- |
| bool update(PwmOut& buzzer){          if(playID==-1 || (isPlaying() && getTime(playID) < timer.read())){              buzzer = 0.5;              playID++;              timer.reset();              if(isPlaying()){                  buzzer.period(1.0 / getFrequency(playID));                  timer.start();                  return true;              }else return false;          }          return false;      } |

1. MP3Player.h : 이 파일은 전체 시스템을 제어하는 MP3Player 클래스가 작성된 파일이다. 이 클래스의 주요 기능으로 버튼 입력이 있다.

먼저 LCD 제어기능이다. 이 클래스는 LCD에 노래 제목, 노래 가사를 출력한다. 이 클래스는 TextLCD.h 라이브러리 사용하고 있는데 나는, 더 제어를 더 쉽게 하기 위하여 다음과 같이 추상화를 하였다. 아래는 두 문자열을 입력받아 LCD에 출력하는 lcdupdate() 함수이다.

|  |
| --- |
| void lcdUpdate(string s1,string s2){          const int MAX\_LENGTH=16;          if(s1.length()==0)s1=lcdBuffer[0];          if(s2.length()==0)s2=lcdBuffer[1];          lcdBuffer[0]=s1;          lcdBuffer[1]=s2;          s1=(s1+string(' ',16)).substr(0,MAX\_LENGTH);          s2=(s2+string(' ',16)).substr(0,MAX\_LENGTH);          lcd.printf((s1+s2).data());      } |

위에 함수를 이용하여 제목과 가사를 출력하는 만약 제목이 LCD의 출력가능 범위보다 길면 짤리게 된다. 따라서 만약 제목이 길면 1초에 한번씩 제목을 왼쪽으로 한 문자 씩 이동시키도록 하였다. 아래는 코드는 start() 함수의 while루프 안에서 실행된다.

|  |
| --- |
| //긴 제목 오른쪽 시프트  if(mp3\_state==menu && selectedMusicName.length()>14){  if(int(nameTimer) >= selectedMusicName.length())restartNameTimer();  lcdUpdate("[MP3] menu", "> "+selectedMusicName.substr(int(nameTimer),14));  } |

그리고 아래는 가사를 출력시키는 코드이다. 이 코드도 start 함수의 while 루프 안에서 실행된다. 가사는 기본적으로 이전 가사에 이어서 출력이 되는데 만약 LCD 출력 범위를 초과한다면 이전 가사를 지우고 새 가사를 출력한다.

|  |
| --- |
| //가사 출력  if(mp3\_state==playing){  if(musicList[selectMusicId].update(buzzer)){  if(lcdBuffer[1].length()+musicList[selectMusicId].getLyric().length() <=16)  lcdUpdate("[MP3] playing", lcdBuffer[1]+musicList[selectMusicId].getLyric());  else lcdUpdate("[MP3] playing", "> "+musicList[selectMusicId].getLyric());  }  if(!musicList[selectMusicId].isPlaying()){  stopMusic();  restartNameTimer();  }  } |

아래는 버튼 입력을 받아 Music 객체를 제어하는 코드이다. 버튼 입력은 polling 방식으로 확인하며, start() 함수의 while 루프에서 실행된다. 두개의 버튼을 사용하는데 mp3의 상태에 따라 입력 기능이 달라 총 4개의 제어 기능이 있다. 각 기능마다 함수가 있다. Mp3상태는 메뉴, 노래 재생, 노래 일시 정지이다.

|  |
| --- |
| //버튼 1 : 다음 노래, 일시 중지  if(button1.read()){  if(mp3\_state==menu)nextMusic();  else pauseMusic();  restartNameTimer();  }  //버튼 2 : 재생, 중지  if(button2.read()){  if(mp3\_state==menu)playMusic();  else stopMusic();  restartNameTimer();  } |

아래는 버튼의 4가지 기능에 대한 함수이다. 각 함수에는 wait 함수가 들어있는데 그 이유는 스위치를 닫았다가 열었을 때 발생하는 “바운싱” 현상에 영향을 받지 않기 때문이다. 바운싱 현상이란 스위치를 닫았다가 열었을 때 정확하게 HIGH→LOW로 상태가 변하는 것이 아닌, 일정시간 동안 전압 요동치는 현상을 말한다. 이 노이즈 때문에 실제로 버튼을 한 번만 눌렀더라도 여러 번 누른 듯한 현상이 발생한다. 여기에 따라서 버튼을 닫았다가 열었을 때 wait을 해줌으로서 바운싱의 영향을 받지 않게 한다. 이를 디바운싱이라 한다.   
또한 함수 내에서도 wait의 위치가 각자 다르다. 이는 사용자 편의성을 위해서이다. 사용자는 입력을 했을 때 즉각적인 반응을 원한다. 따라서 lcd를 가장 먼저 수정한다. 그후에는 시스템이 정확한 출력을 할 수 있도록 하는 위치에 wait을 한다. 일시정지 기능을 다루는 pause 함수에서는 먼저 wait을 하고 일시정지 해제를 해야한다. 이 이유는 다음 음 재생까지 남은 시간이 0.1초보다 작은 경우 일시정지 해제를 먼저 한다면 wait에 의해 0.1초동안 현재 음이 재생되기 때문이다.

|  |
| --- |
| void nextMusic() { //다음 노래를 선택          if (++selectMusicId == musicList.size())selectMusicId = 0;          lcdUpdate("", "> "+musicList[selectMusicId].getName());          selectedMusicName = musicList[selectMusicId].getName();          wait(0.1);      }      void pauseMusic() { //노래를 일시정지          if(mp3\_state==playing){              lcdUpdate("[MP3] pause", "");              musicList[selectMusicId].pause(buzzer);              mp3\_state=pause;              wait(0.1);          }else{              lcdUpdate("[MP3] playing", "");              wait(0.1);              musicList[selectMusicId].unpause(buzzer);              mp3\_state=playing;            }      }      void stopMusic() { //노래를 중지          lcdUpdate("[MP3] menu", "> "+musicList[selectMusicId].getName());          musicList[selectMusicId].stop(buzzer);          mp3\_state=menu;          wait(0.1);      }      void playMusic() {          lcdUpdate("[MP3] playing", "> ");          wait(0.1);          musicList[selectMusicId].play();          mp3\_state=playing;      } |

**실행 결과**

보고서에서는 음향 신호를 나타낼 수 없기 때문에 기본 화면, 긴 제목 시프트 기능, 가사 출력 기능, 노래 일시정지만 설명하였다.

1. 기본 화면  
   스크린샷, 텍스트, 전자제품, 전자 공학이(가) 표시된 사진

   자동 생성된 설명
2. 긴 제목 시프트  
   스크린샷, 텍스트, 회로, 전자제품이(가) 표시된 사진

   자동 생성된 설명
3. 가사 출력 기능

텍스트, 스크린샷, 멀티미디어 소프트웨어, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

스크린샷, 텍스트, 전자제품, 전자 공학이(가) 표시된 사진

자동 생성된 설명

1. 노래 일시정지  
   전자제품, 전자 공학, 케이블, 전기 배선이(가) 표시된 사진

   자동 생성된 설명

**마치며**

이 프로젝트는 안정성, 사용자 편의성, 유지보수성에 중점을 두고 진행을 하였다.

스위치에서 디바운싱 작업을 하였으며, 임계 구역 문제가 발생할 수 있는 thread 대신 timer를 사용하였다.

긴 제목은 1초에 한번 왼쪽으로 이동시켜 사용자가 전체 제목을 볼 수 있게 하였고, 가사를 출력 범위에 따라 이어서 출력하거나 새로 출력하는 등에 작업을 하였고, wait() 함수를 적절히 위치시켰다.

코드에서 Music 클래스와 MP3Player 클래스를 만들어 객체지향적으로 작성하여 유지보수가 용이하게 하였다.

하지만 이 프로젝트는 피에조 부저로는 사람의 음성을 출력하긴 어렵다는 점, Text LCD로는 한글을 출력하기 어렵다는 점이 한계이다. Text LCD는 패널 중 한 칸에 문자를 출력하는데 한 칸은 5\*8 크기다. 만약 “떴다떴다 비행기”에서 “떴”을 출력한다면 아래와 같이 출력할 수 있는데 보기는 좋지 않다. TextLCD에서 한글을 출력할 수 있는 라이브러리를 찾아보았지만 모두 한글을 출력하는데 2칸 이상을 사용하고 있었다. 2칸 이상을 사용해서 한글을 출력해도 가독성은 좋지 않았다.

