

베어팻



황제곰
(emperonics)
곰손도 만든다!

베어팻몰: <https://smartstore.naver.com/bearfabmall>

[프로필](#)

카테고리

전체 보기 (88)

사용가이드 (9)

프로젝트

Learn (75)

라즈베리파이활용 (16)

라즈베리파이GPIO, 카메라 (14)

라즈베리파이통신 (2)

라즈베리파이(리눅스) (5)

아두이노 (28)

마이크로파이썬 (0)

tutorial (0)

전자부품, 전자회로 (7)

하드웨어 (2)

태그

라즈베리파이, 아두이노, raspberrypi, 라즈베리파이4, 라즈베리파이활용, gpio, 3d프린터, Pimoroni, arduino, i2c, octoprint, 레드보드, 파이썬, InkyPhat, pwm

모두보기

게시판 (0)

아두이노 28개의 글

목록열기

아두이노

아두이노로 MP3 모듈 제어하기(DFPlayer Mini)

 황제곰 · 2021. 1. 5. 21:54

URL 복사

+아웃추가

안녕하세요. 이번 포스팅에서는 소형 사이즈의 MP3 디코딩 모듈(재생모듈)인 DFPlayer Mini라는 모듈을 아두이노로 제어하는 방법에 대해서 알아 보도록 하겠습니다.

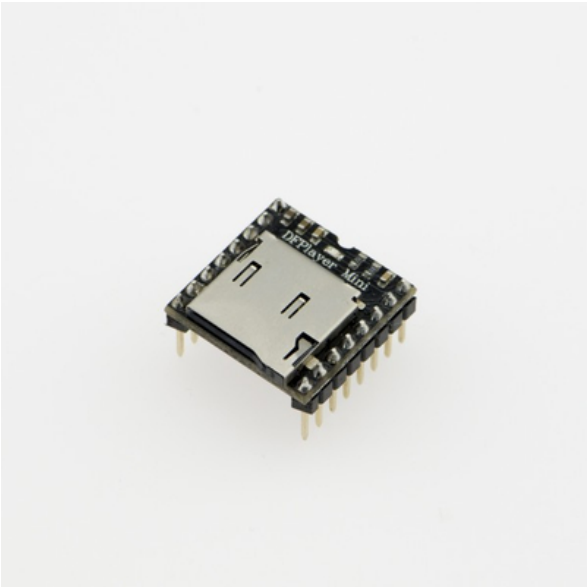




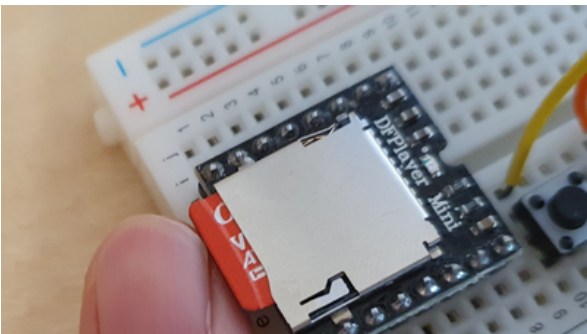
아두이노 MP3 플레이어 모듈<DFPlayer Mini> : 베어팜물
아두이노 MP3 플레이어 모듈<DFPlayer Mini>
smartstore.naver.com

DFPlayer Mini

- DFPlayer Mini 모듈 소개 -



DFPlayer 미니 모듈은 mp3 음악파일을 스피커나 이어폰 잭으로 출력해주는 기능을 갖춘 소형 디코딩 모듈인데요. 사이즈가 작고 가격도 저렴한 편이라서 아두이노로 음악관련 프로젝트를 진행할 때 손쉽게 적용이 가능한 모듈입니다.



사실 마이크로 컨트롤러가 있어야만 이 모듈을 사용할 수 있는건 아니고 저항과 스위치만 몇개 연결하면 별도의 마이크로 컨트롤러 없이 바로 음악 재생 및 몇몇 기능(다음곡, 이전곡 넘기기, 볼륨 조절 등)은 사용이 가능하긴 합니다. 하지만 마이크로 컨트롤러와 시리얼통신을 통해 연결한뒤 정해진 명령어(프로토콜)를 넣으면 기본기능외에 훨씬 더 다양한 기능을 발휘 할 수 있기 때문에 아두이노와 같은 마이크로 컨트롤러를 활용하면 음악재생이 필요한 프로젝트에 폭넓게 사용될 수 있습니다.

- 모듈 스펙 및 기능 -

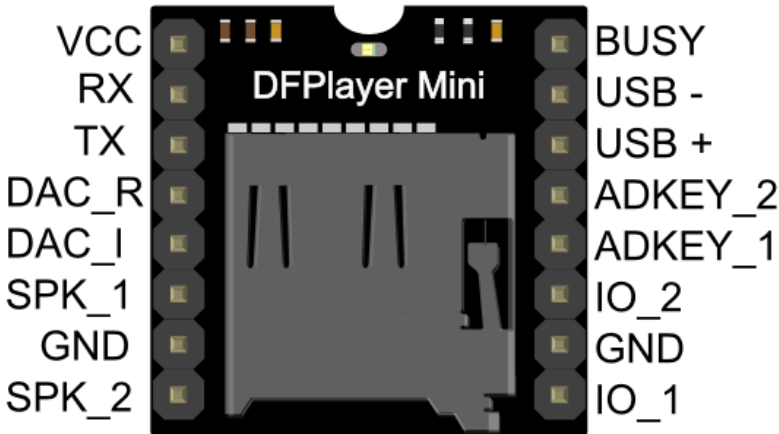
데이터 시트에서 제공하고 있는 모듈의 스펙은 아래와 같습니다.

- ▶ 샘플링속도 (kHz): 8/11.025/12/16/22.05/24/32/44.1/48
- ▶ 24-bit DAC output, support for dynamic range 90dB , SNR support 85dB
- ▶ FAT16, FAT32파일 시스템, 32기가 TF카드, 32기가 Udisk, 64MB의 NORFLASH 지원
- ▶ 다양한 제어방법 지원: 입출력 컨트롤러, 시리얼 제어, AD버튼 제어
- ▶ 어드벌타이즈 모드 지원
- ▶ 폴더별로(100개까지) 255개의 노래를 저장할 수 있음.
- ▶ 30단계로 볼륨조절 가능, 6개의 이퀄라이즈 모드 변경 가능

SD카드는 FAT16과 FAT32파일 시스템을 지원하기 때문에 음악을 넣기 전에 해당 포맷으로 먼저 포맷을 해 주어야 합니다.

- 핀맵 및 기능 -

모듈에 사용되는 핀 맵과 기능은 아래그림 및 표와 같습니다.



번호	핀	핀 기능	설명
1	VCC	입력 전압	DC3.2V~5V 입력 가능
2	RX	UART 입력	시리얼 통신 입력핀
3	TX	UART 출력	시리얼 통신 출력 핀
4	DAC_R	오디오 출력 오른쪽	

배어팩

이 블로그에서 검색

7	GND	Ground	전원 GND 연결
8	SPK_2	스피커 +	스피커 2개의 선 중 하나 연결
9	IO_1	간편 입력 1	버튼 연결후 클릭시 이전 곡재생 (길게 누르고 있으면 볼륨 감소)
10	GND	Ground	전원 GND 연결
11	IO_2	간편 입력 2	버튼 연결후 클릭시 이전 곡재생 (길게 누르고 있으면 볼륨 증가)
12	ADKEY1	AD키 포트 1	AD키로 수동 제어시 사용
13	ADKEY2	AD키 포트 2	AD키로 수동 제어시 사용
14	USB+	USB+ DP	USB용 포트(이 모듈에서는 SD카드를 사용하므로 거의 사 용할 일이 없음.)
15	USB-	USB+ DM	USB용 포트(이 모듈에서는 SD카드를 사용하므로 거의 사 용할 일이 없음.)
16	BUSY	재생 여부 표시 핀	노래 재생 중일 때LOW, 정지시 HIGH 출력

- SD카드에 노래 넣기 -

SD카드에 파일을 넣기 전에 먼저 SD카드의 파일 포맷을 FAT16또는 FAT 32로 포맷을 해 줍니다. 포맷이 된 SD 카드에 mp3파일을 넣으면 되는데요. 음악을 넣는 방법이 4가지 방법이 있습니다. 특히 **아두이노로 모듈을 제어할 경우 이 mp3파일을 넣는 방법에 따라 제어 방법(사용함수)이 달라지며, 라이브러리를 사용할 때 제어 함수를 잘 못 사용할 경우(예: mp3폴더에서 mp3 파일을 재생하는 함수를 일반 폴더에 보관후 사용했을 경우 등등...) 모듈이 정상 동작하지 않으므로 mp3파일 저장 방법에 따른 재생방법(함수)을 잘 확인해야 합니다.**

▶ "mp3" 폴더에 파일 넣기

SD카드의 최상위 위치에서 **"mp3"** 라는 폴더를 만들고 그 하위에 mp3파일을 넣는 방법입니다. 이 때 음악 파일의 이름은 **"nnnnXXXX"**의 형식이어야 합니다. 여기서 nnnn은 숫자 0001부터 9999까지의 값이고 XXXX 파일이름이며 어떤 값이 들어가도 괜찮습니다. mp3폴더내에 아래 사진과 같이 넣으면 됩니다.

SD://mp3/

이름

수정한 날짜

유형

크기

0001 장범준 - 흔들리는 꽃들 속에서 네 잠부향이 느껴진거야

2020-12-01 오전 10:51

MP3 파일

6,795KB

0002 방탄소년단-Dynamite

2020-12-01 오전 10:51

MP3 파일

7,900KB

0003 임창정-힘든 건 사랑이 아니다

2020-12-01 오전 10:50

MP3 파일

10,905KB

0004 BLACKPINK Lovesick Girls

2020-12-01 오전 10:52

MP3 파일

7,732KB

0005 산들-취기를 빌려

2020-12-01 오전 10:51

MP3 파일

9,122KB

0006 장범준-잠이 오질 않네요

2020-12-01 오전 10:50

MP3 파일

10,975KB

0007 화물원정대-DON'T TOUCH ME

2020-12-01 오전 10:51

MP3 파일

9,046KB

0008 스탠딩 에그-오래된 노래

2020-12-01 오전 10:51

MP3 파일

10,895KB

0009 방탄소년단-Savage Love

2020-12-01 오전 10:51

MP3 파일

7,473KB

0010 마마무-딩가딩가

2020-12-01 오전 10:51

MP3 파일

7,424KB

0011 규현-내 마음이 윙윙했던 순간

2020-12-01 오전 10:51

MP3 파일

9,337KB

공감 2

댓글 18

베어패드

▶ 2차다 숫자도 이루어진 폴더에 음악 파일 넣기(폴더당 최대 9999개 저장 가능)

만약 폴더구조로 음악을 넣고 싶은데 음악의 개수가 999개 이상일 경우는 폴더내의 음악파일의 이름을 "nnnnXXX.mp3" 의 형식으로 저장하면 됩니다. 여기서 nnnn은 0001에서 9999까지의 숫자이고 XXX는 파일이름으로 어떤 값이 들어가도 됩니다.



예시

이럴 경우 폴더의 음악을 재생할 때는 "playLargeFolder(uint8_t folderNumber, uint16_t fileName)" 함수를 사용합니다.

```
//사용예시
player.playLargeFolder(5, 1);    //5번 폴더의 1번 파일 재생
player.playLargeFolder(5, 2);
player.playLargeFolder(5, 211);  // 5번 폴더의 211번 파일 재생
```

▶ 임의의 위치에 파일 넣기

이 방법은 위 방법들에도 적용되는 방법입니다. 파일을 폴더에 넣어도 되고 그냥 SD카드의 최상위 폴더에 넣어도 됩니다.(파일이름도 상관없습니다.) 라이브러리의 play() 함수를 쓰면 SD카드에 mp3파일이 복사된 순서대로 음악파일이 재생됩니다. 파일이름에 상관없이 SD카드에 저장된 순서대로 내부적으로 임의의 인덱스를 생성되어 그 순서대로 음악이 재생됩니다.

* 주의 *

음악 파일을 넣을 때 한가지 주의 해야 할 점이 있는데요. 음악파일을 번호를 잘 지정해서 폴더내 넣었을 때에도 특정운영체제 (특히 맥)에서는 음악파일이 저장된 순서에 따라 별도의 인덱스 파일이나 숨김파일이 생성되는 경우가 있습니다. 이런 인덱스 파일이나 숨김파일이 생기면 내가 번호를 매긴 파일이름보다 숨김파일이 먼저 적용되어 내가 원하는 번호대로 음악이 재생되지 않습니다. 그럴 경우는 숨김파일을 표시해서 해당 파일과 인덱스 파일을 삭제 해 주시면 됩니다.



마이크로컨트롤러(아두이노) 없이 모듈 사용하기

위에서 언급했다시피 이 모듈은 꼭 외부의 마이크로컨트롤러가 있어야만 음악재생을 할 수 있는 것은 아닙니다. 독립된 모듈로 음악을 재생하기 위한 방법이 2가지가 있는데요.

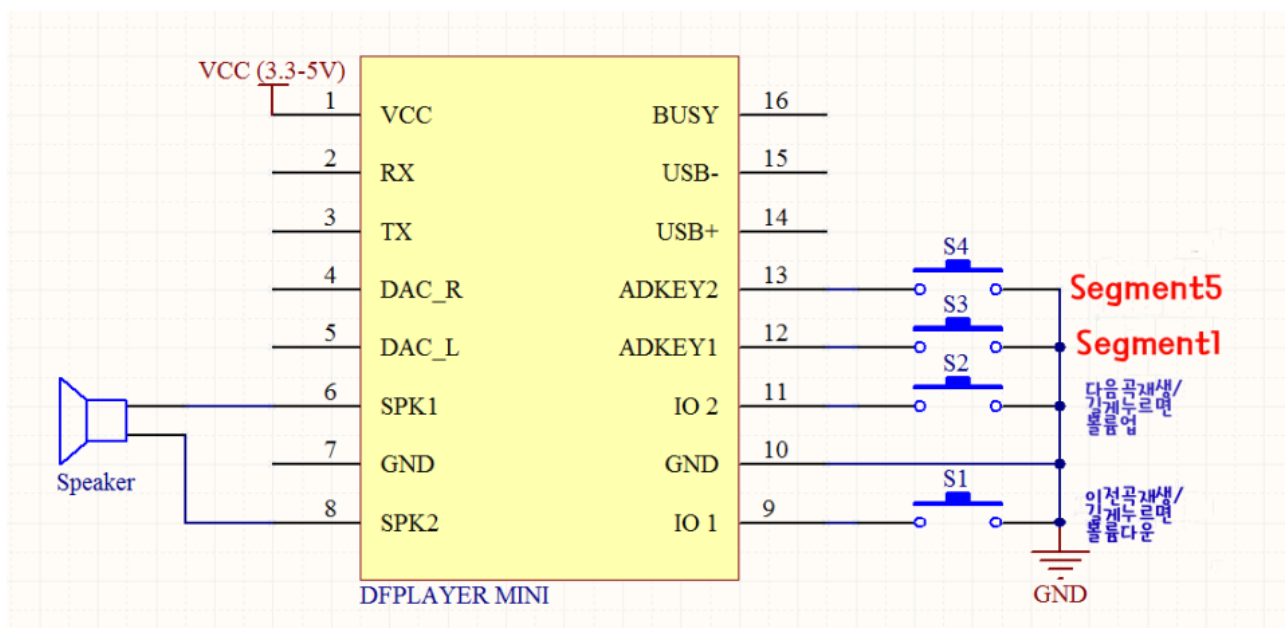
- IO 모드로 사용하기 -

아래와 같이 회로를 연결하면 간단히 4개의 버튼으로 음악을 재생할 수 있습니다. segment1과 연결된 버튼을 누르면 1번 음악파일(첫번째 복사된 파일)이 재생되고 segment5와 연결된 버튼을 누르면 5번 파일(5번째 복사된 파일)이 재생됩니다. S1, S2를 누르면 이전곡 또는 다음곡이 재생되고 길게 누르면 볼륨을 키우거나 줄일 수 있습니다.

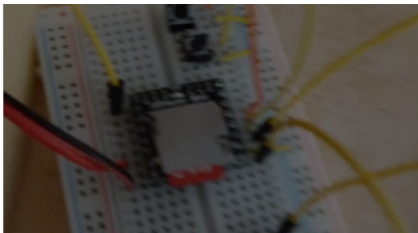
3. I/O Mode

Here comes the most simple way to use this module.

- Refer diagram



DFPlayer Mini I/O 모드
634 2



00:00 00:51

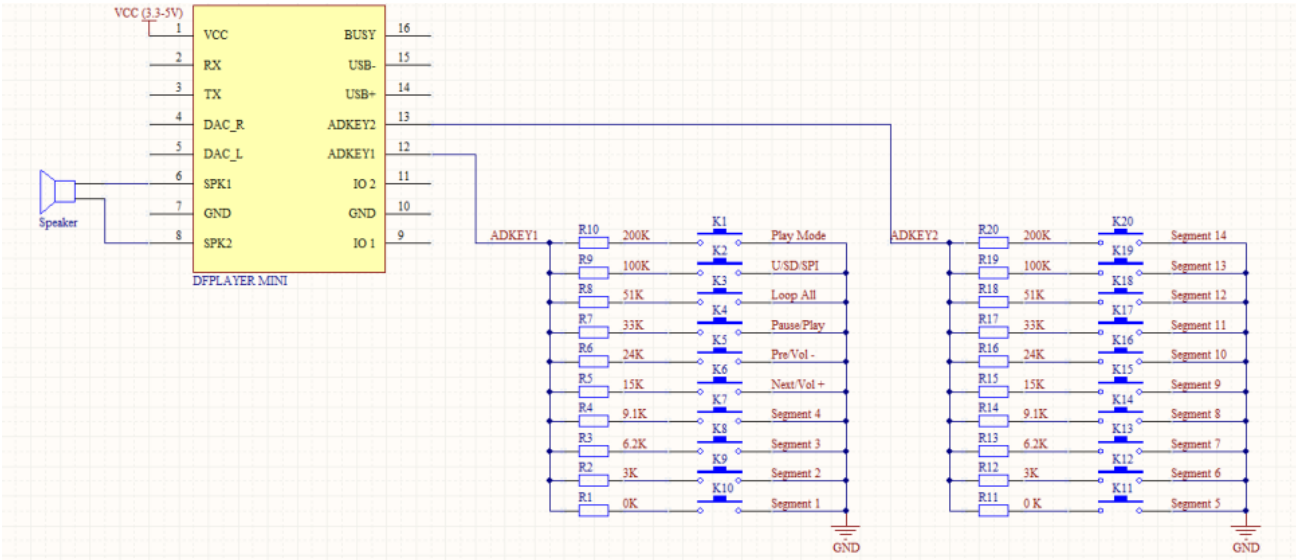
자동

DFPlayer Mini I/O 모드

- AD 키 모드로 사용하기 -

위에서 사용했던 I/O 모드는 간단한 음악 재생 및 볼륨 조절 기능만 사용할 수 있었는데요. AD키 모드로 사용하면 더 많은 기능을 사용할 수 있습니다. 아래 사진과 같이 AD키용 핀 2개에 AD키패드를 연결해 주면 더 많은 기능을 사용할 수 있습니다.





AD 키패드는 아래 사진과 같은 모양의 부품입니다. 각 키에 다른 값의 저항과 연결되어 있어서 이 저항값을 바탕으로 어떤 키가 눌렸는지를 감지할 수 있는 부품입니다.



만약 키패드가 없으면 위 회로도에 나온 저항값과 일반 탭스위치를 연결해서 사용해도 동일한 효과를 발휘할 수 있습니다.

아두이노로 제어하기

사실 요즘은 대부분 스마트폰을 사용해서 음악을 듣기 때문에 위에서처럼 별도로 mp3모듈로 음악을 재생할 일은 별로 없을 것 같네요. 이제 아두이노로 모듈을 제어하는 방법에 대해서 알아 보겠습니다.

- 회로연결 -

어 시리얼핀은 다른 핀으로 지정해도 괜찮습니다.)



DFMINI Player(핀번호)	아두이노(핀번호)
VCC(1)	5.0V 또는 3.3V
RX(2)	소프트웨어시리얼 TX핀(11번)
TX(3)	소프트웨어시리얼 RX핀(10번)
SPK1(6)-스피커와 연결	
SPK2(8)-스피커와 연결	
Ground(7)	아두이노의 GND와 연결

*** 참고 ***

▶ 데이터시트에 노이즈 발생시 위와 같이 1K옴을 아두이노의 소프트웨어 TX핀과 모듈의 RX핀사이에 달아주라고 나와 있는 데요.(저는 우노 모델로도 별도의 저항을 연결하지 않았는데 문제는 없었습니다.) DFMini Player 모듈이 원래 3.3V 신호레벨을 사용하기 때문에 5V를 사용하는 우노모델과 통신 할 경우 아두이노의 5V를 바로 3.3V를 사용하는 모듈로 연결하면 오류가 발생하기 때문인 것으로 보입니다. 3.3V 신호레벨을 사용하는 아두이노의 경우는 별도의 저항 연결없이 통신이 가능합니다.

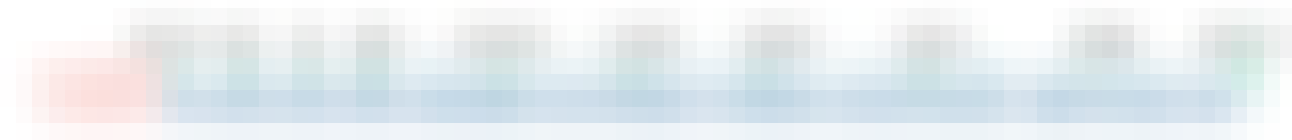
▶ mp3 모듈에 전원이 인가되면 mp3모듈이 초기화되고 음악을 재생할 수 있는 준비를 하게 됩니다. 이 시간이 대략 500ms에서 1500ms(sd카드에 보관한 mp3파일이 많은 경우) 걸리고 준비가 되면 모듈에서 시리얼 통신으로 준비가 되었다는 메시지를 송신합니다. 따라서 아두이노에서 모듈을 제어하려면 이 준비시간 동안 기다려 주거나 시리얼 통신으로 모듈로부터 들어오는 수신메시지를 확인해서 모듈이 준비가 되었을 때 제어를 시작해야 합니다. 또한 모듈이 잘못된 제어명령어를 받거나 전력부족 등 여러가지 이유로 시리얼명령어를 수신하지 못하는 사태가 발생하 수 있습니다. 이 때에도 모듈의 전원을 빼다가 다시 이가

- 시리얼명령어를 통한 제어방법 -

* 이 부분은 동작원리에 대한 설명으로 바로 사용하는 것이 목적이신 분들은 아래 '라이브러리를 통한 제어'로 넘기셔도 됩니다. *

데이터 시트에 보면 어떻게 모듈을 제어할 수 있는지에 대해서 자세히 나와 있는데요. 시리얼 통신으로 9600bps의 속도로 아래 명령어 구문을아두이노에서 송신하면 mp3모듈에서는 수신된 명령어구문을 해독해서 지정된 모드로 동작하게 됩니다.

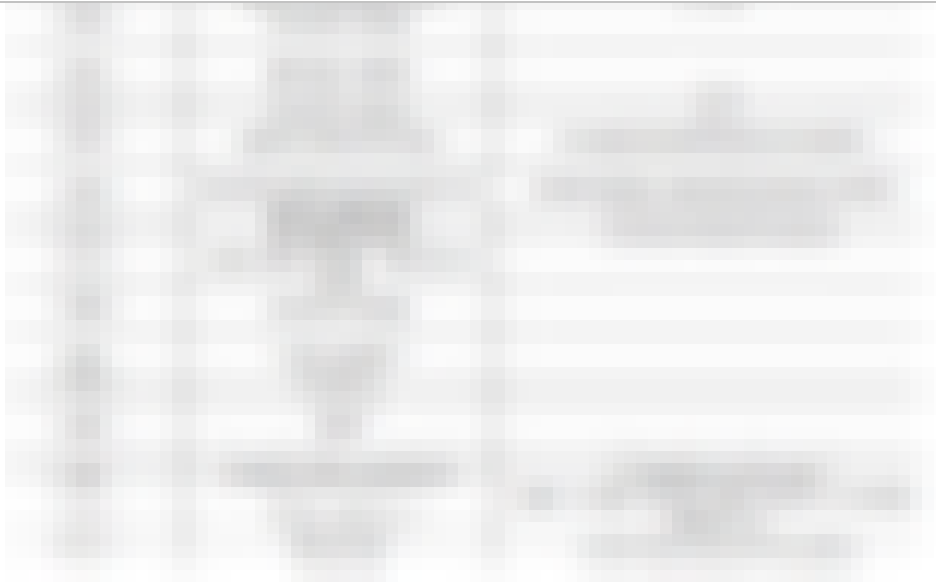
명령어 구문의 형식은 아래와 같이 10개의 요소로 이루어져 있습니다.



각각의 요소에 대한 더 자세한 내용은 아래 표와 같습니다.

번호	요소	설명
①	\$S(시작바이트)	명령어 구문의 시작을 알림, 값은 0x7E
②	VER(버전)	버전 입력 기본값은 0xFF
③	LEN(길이)	전체 명령어 구문의 길이 입력, 시작바이트, 종료바이트, 체크섬은 빼기 때문에 값은 0x06
④	CMD(명령어)	MP3 모듈에 지정할 명령어 입력, 명령어의 내용은 하단의 표 참고
⑤	FEEDBACK(피드백여부)	명령어 구문 송신후 수신확인 피드백을 받을 것인지 지정 확인을 받기 위해서는 0x01, 안 받을 거면 0x00
⑥	PARAM1(파라미터1)	명령어와 더불어 지정해줄 파라미터가 있을 때 넣어줄 추가 파라미터의 상위 바이트
⑦	PARAM2(파라미터2)	명령어와 더불어 지정해줄 파라미터가 있을 때 넣어줄 추가 파라미터의 하위 바이트
⑧	CHECKSUM1(체크섬1)	전체 명령어 구문의 오류 확인을 위한 체크섬 상위 바이트
⑨	CHECKSUM2(체크섬2)	전체 명령어 구문의 오류 확인을 위한 체크섬 하위 바이트
⑩	\$O(종료 바이트)	명령어 구문의 종료를 알리기 위한 바이트 값은 0xEF

CMD(명령어)에 들어갈 수 있는 값들은 아래와 같습니다.



언뜻 보면 굉장히 복잡해 보이는데요. 명령어의 구조를 한번만 파악하고 나면 CMD에 해당하는 실제 명령어만 변경하면 되기 때문에 그다지 복잡하지 않습니다.

간단히 볼륨을 15으로 지정하고 2번 트랙을 재생하는 명령어 구문을 보내는 방법을 예시로 들어 보겠습니다.

기본상태에서 이 모듈의 볼륨은 30으로 지정되어 있습니다. 외부 전원을 사용하지 않고 USB전원으로 연결된 아두이노의 5V로 모듈을 구동시 가끔 전력이 부족해서 그런지 음악재생중 드르륵 소리가 나면서 모듈이 정상 동작하지 않는 경우가 발생합니다. 이를 방지하기 위해서 먼저 모듈의 볼륨을 15으로 지정해 보겠습니다. 위 명령어 중 볼륨을 조절하는 명령어는 **"0x06" Specify Volume**에 해당합니다. 그리고 이 명령어는 0-30의 파라미터를 추가로 입력 가능한데요. 이 파라미터에 지정하고 싶은 볼륨값을 입력하게 됩니다.(우리는 15였습니다.) 따라서 위 명령어 구문에 이 값을 넣으면 전송할 명령어 구문의 값은 순서대로 아래의 값을 전송하면 됩니다.

0x7E(시작바이트), 0xFF(버전), 0x06(명령어 구문의 길이), **0x06(볼륨조절 명령어)**, 0x00(피드백 안받음), 0x00(파라미터 상위바이트 없음), **0x0F(볼륨지정값, 15를 16진수로표현)**, **체크섬 상위바이트**, **체크섬 하위바이트**, 0xEF(종료바이트)

체크섬을 계산하는 방법은 아래 공식을 통해 계산할 수 있습니다.

체크섬(2바이트) = 0xFFFF-(버전바이트+명령어길이+CMD+피드백여부 + 파라미터1+파라미터2)+1

따라서 위 볼륨조절을 기준으로 체크섬을 계산해 보면

$(0xFF+0x06+0x06+0x00+0x00+0x0F) = 0x011A$ 이므로

체크섬 = $0xFFFF - 0x011A + 1$

체크섬 = 0xFEE6

이 중 체크섬의 상위바이트는 0xFE, 하위바이트는 0xE6 가 됩니다.

체크섬까지 계산해서 볼륨을 15로 지정하기 위해 보낼 전체 명령어 구문은 아래와 같습니다.

0x7E 0xFF 0x06 0x06 0x00 0x00 0x0F 0xFE 0xE6 0xEF

2번 트랙을 재생하기 위한 방법도 위 볼륨조절과 동일합니다. 위 명령어 표 중 트랙 넘버를 선택하는 명령어는 **"0x03" Specify tracking(NUM)**에 해당합니다. 그리고 이 명령어는 파라미터로 0-2999의 값을 넣을 수 있는데요. 이 파라미터가 트랙번호가 됩니다. 따라서 2번 트랙을 지정할 땐 2번을 넣어주면 되겠죠? 이를 바탕으로 체크섬까지 계산해서 명령어 전체 구문을 계산하면 아래의 구문이 완성됩니다.

```
0x7E 0xFF 0x06 0x03 0x00 0x00 0x02 0xFE 0xF6 0xEF
      ----          -----
      play          2번   체크섬
```

이 명령어 구문을 순서대로 시리얼 통신을 통해 mp3모듈로 보내면 됩니다.

위 방법을 아두이노 코드를 통해 직접 확인 해 보도록 하겠습니다.

```
#include "SoftwareSerial.h"          // 소프트웨어 시리얼 라이브러리

SoftwareSerial mySerial(10, 11);      // 소프트웨어 시리얼 통신 핀 지정

# define Start_Byte 0x7E              // 시작 값
# define Version_Byte 0xFF            // 버전 값
# define Command_Length 0x06          // 길이 값
# define End_Byte 0xEF                // 종료 값
# define Acknowledge 0x00             // 피드백 미 수신

void setup()
{
    mySerial.begin(9600);              // 소프트웨어 시리얼 통신 개시
    delay(2000);                      // 2초 대기
    specify_Volume(15);                // 볼륨을 15로 지정
    specify_Track(3);                 // 3번 트랙 재생
}

void loop()
{
}

void specify_Volume(byte level)        // 볼륨 조절 함수
{
    execute_CMD(0x06, 0x00, level);    //볼륨조절 명령어 0x06과 볼륨레벨을 파라미터로 전달
}

void specify_Track(int16_t track)       // 트랙 지정 함수
{
    execute_CMD(0x03, highByte(track), lowByte(track));    // 트랙재생 명령어 0x03과 재생될 트랙을 파라미터로 전달
}

void execute_CMD(byte CMD, byte Par1, byte Par2)    // 시리얼 통신을 통해 실제 명령어구문을 전달하는 함수
{
}
```

```
for (byte k=0; k<10; k++)    // 명령어 구문을 시리얼 통신을 송신
{
    mySerial.write(Command_line[k]);
}
}
```

mp3모듈에 전원이 인가되어 있는 상태에서 아두이노에 위 코드를 업로드 하면 3번 트랙의 노래가 15의 볼륨으로 재생되는 것을 확인 할 수 있습니다.

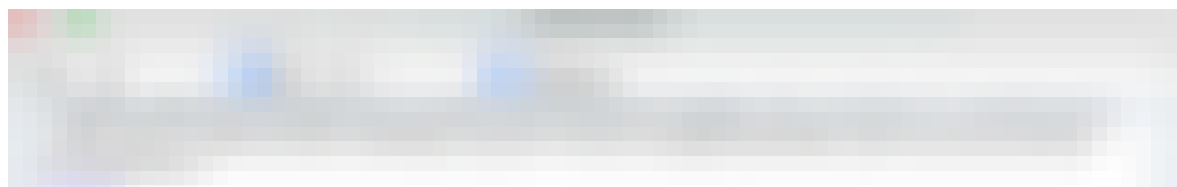
다른 명령어들도 위와 같이 새로 함수를 만들어서 사용할 수 있겠죠? ㅎㅎ

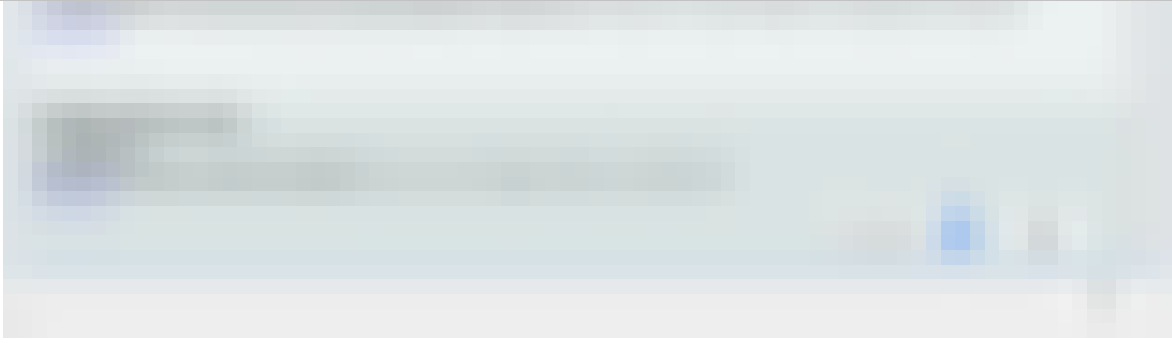
- 라이브러리 설치 -

사실 위 방법처럼 일일이 새로 함수를 작성할 필요없이 이미 개발되어 있는 여러 라이브러리를 사용하면 손쉽게 모듈을 제어 할 수 있습니다.



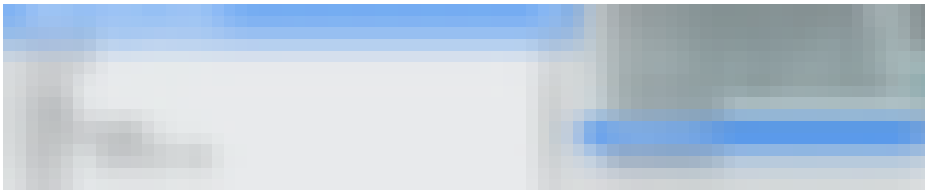
라이브러리 설치를 위해 아두이노의 IDE에서 [스케치]-[라이브러리 포함하기]-[라이브러리 관리]를 눌러 줍니다. 라이브러리 매니저에서 **dfplayer**을 검색하면 여러 라이브러리가 나오는데요. 이중 이번 포스팅에서는 DFROBOT사에서 제작한 **DFRobotDFPlayerMini** 라이브러리를 설치해 줍니다.



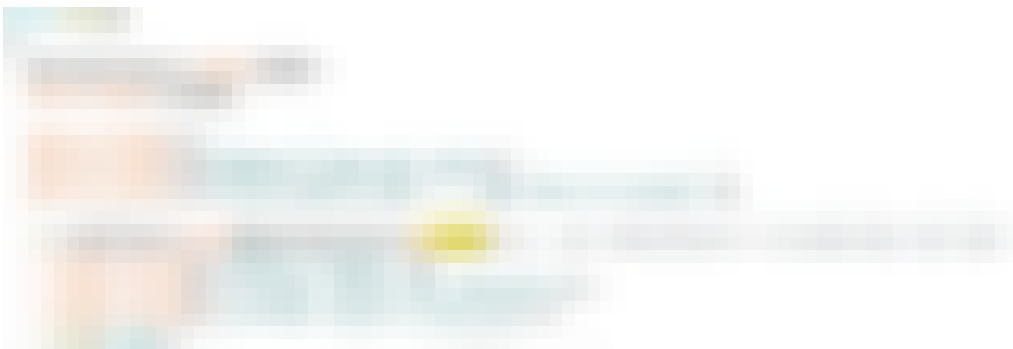


- 라이브러리 예제 실행해 보기 -

라이브러리 설치를 위해 아두이노의 IDE의 [파일]-[예제]-[DFRobotDFPlayerMini]-[GetStarted] 를 열어 줍니다.



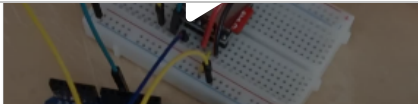
예제 파일에 보면 객체를 초기화 하는 begin 함수부분이 있는데 기본 상태에서는 라이브러리의 생성자에 feedback 메시지를 수신하는 것으로 설정이 되어 있습니다. 저는 feedback을 수신하는 상태에서는 자꾸 오류가 발생해서 수신하지 않도록 아래와 같이 false를 인자로 넣어주었더니 제대로 동작했습니다. 혹시 오류가 발생하면 아래와 같이 false를 넣어서 사용해 보시길 바랍니다.



이제 소스코드를 업로드 하면 아래 영상과 같이 3초에 한번씩 전체 파일을 순회하면서 음악이 재생되는 것을 확인 할 수 있습니다.(혹시 동작이 잘 안되면 소스코드 업로드 후 mp3 모듈의 전원을 분리했다가 다시 연결해 주면 잠시후 동작합니다.)

아두이노 mp3 모듈 예제
324 0





00:00 00:20

자동

아두이노 mp3 모듈 예제

- 라이브러리 함수 알아 보기 -

이제 라이브러리의 주요 함수에 대해 알아 보겠습니다. **[파일]-[예제]-[DFRobotDFPlayerMini]-[FullFunction]** 예제 파일을 실행해 보면 거의 모든 함수에 대해 설명이 나와 있기 때문에 해당 파일을 살펴보면 대부분의 기능을 알 수 있습니다.

```
#include "Arduino.h"
#include "SoftwareSerial.h"
#include "DFRobotDFPlayerMini.h"

SoftwareSerial mySoftwareSerial(10, 11); // 소프트웨어 시리얼 용 핀 지정
DFRobotDFPlayerMini myDFPlayer;          // 객체 생성
void printDetail(uint8_t type, int value);

void setup()
{
    mySoftwareSerial.begin(9600);          // 소프트웨어 시리얼 통신 개시
    Serial.begin(115200);

    Serial.println();
    Serial.println(F("DFRobot DFPlayer Mini Demo"));
    Serial.println(F("Initializing DFPlayer ... (May take 3~5 seconds)"));

    if (!myDFPlayer.begin(mySoftwareSerial), false) { //객체 초기화
        Serial.println(F("Unable to begin:"));
        Serial.println(F("1.Please recheck the connection!"));
        Serial.println(F("2.Please insert the SD card!"));
        while(true);
    }
    Serial.println(F("DFPlayer Mini online.));

    myDFPlayer.setTimeout(500); //시리얼 통신용 타임아웃 시간 지정

    ----볼륨조절----
    myDFPlayer.volume(10); //0~30사이의 값을 인수로 입력.
    myDFPlayer.volumeUp(); //볼륨을 1단계씩 키울 때 사용
    myDFPlayer.volumeDown(); //볼륨을 1단계씩 내릴 때 사용

    ----이퀄라이즈 모드 지정시 사용 ----
```




```

myDFPlayer.EQ(DFPLAYER_EQ_CLASSIC); //클래식 모드
myDFPlayer.EQ(DFPLAYER_EQ_BASS); // BASS모드

---디바이스 모드 지정시 사용: 위 모듈 사용시 별도로 지정하지 않아도 SD카드 모드로 지정되기 때문에 사용하지 않아도 됨 ---
myDFPlayer.outputDevice(DFPLAYER_DEVICE_U_DISK);
myDFPlayer.outputDevice(DFPLAYER_DEVICE_SD);
myDFPlayer.outputDevice(DFPLAYER_DEVICE_AUX);
myDFPlayer.outputDevice(DFPLAYER_DEVICE_SLEEP);
myDFPlayer.outputDevice(DFPLAYER_DEVICE_FLASH);

----mp3 모듈 제어 모드----
myDFPlayer.sleep(); //슬립모드
myDFPlayer.reset(); //슬립모드로부터 복귀
myDFPlayer.enableDAC(); //음악파일에서 디코딩 상태로 설정
myDFPlayer.disableDAC(); //음악파일을 디코딩 하지 않음
myDFPlayer.outputSetting(true, 15); //이 모듈에서는 적용되지 않음

----Mp3 재생----
myDFPlayer.next(); //다음 곡 재생
delay(1000);
myDFPlayer.previous(); //이전 곡 재생
delay(1000);
myDFPlayer.play(1); //1번째 노래 재생(SD카드에 파일이 저장된 순서대로 재생됨)
delay(1000);
myDFPlayer.loop(1); //첫번째 노래 반복재생
delay(1000);
myDFPlayer.pause(); //일시 정지
delay(1000);
myDFPlayer.start(); //다시 재생
delay(1000);
myDFPlayer.playFolder(15, 4); //지정한 폴더의 트랙 재(15번 폴더의 4번 트랙재생)
delay(1000);
myDFPlayer.enableLoopAll(); //전체 MP3파일 재생(SD카드에 파일이 저장된 순서대로 쭉 재생)
delay(1000);
myDFPlayer.disableLoopAll(); //전체 MP3파일 재생 중지(enableLoopAll 함수 이후 실행해야 효과 적용 됨)
delay(1000);
myDFPlayer.playMp3Folder(4); //mp3폴더에 저장된 트랙을 재생(SD:/MP3/0004.mp3 재생) 트랙번호는 (0~65535) 지정 가능
delay(1000);
myDFPlayer.advertise(3); // 광고모드를 실행해서 SD:/ADVERT/0003.mp3 파일을 실행, 밑에 추가 설명 참고
delay(1000);
myDFPlayer.stopAdvertise(); //광고 모드 해제
delay(1000);
myDFPlayer.playLargeFolder(2, 999); //폴더내에 mp3파일이 여러개 여서 0001.mp3형식으로 저장했을 때 사용
delay(1000);
myDFPlayer.loopFolder(5); //지정한 폴더엔 있는 모든 음악 순회재생(SD:/05번 폴더 순회 재생)
delay(1000);
myDFPlayer.randomAll(); //모든 음악파일 랜덤 재생
delay(1000);
myDFPlayer.enableLoop(); //현재 재생 중인 트랙을 반복 재생 모드로 지정
delay(1000);
myDFPlayer.disableLoop(); //현재 반복 재생 트랙을 반복재생 모드에서 해제
delay(1000);

//----Read information----
Serial.println(myDFPlayer.readState()); //mp3 모듈 상태 읽어오기
Serial.println(myDFPlayer.readVolume()); //현재 볼륨값 읽어오기

```

```

}

void loop()
{
    static unsigned long timer = millis();

    if (millis() - timer > 3000) {
        timer = millis();
        myDFPlayer.next(); //다음 곡 재생
    }

    if (myDFPlayer.available()) {
        printDetail(myDFPlayer.readType(), myDFPlayer.read()); //Print the detail message from DFPlayer to handle d
    }
}

```

주석만 봐도 대부분의 기능을 알수 있기 때문에 추가적으로 별도의 설명은 필요 없을 것 같네요 ㅎ

주석을 봐도 헛갈릴만한 기능만 추가적으로 몇가지 더 알아 보겠습니다.

▶ mp3 모듈 제어모드

```

myDFPlayer.sleep(); //슬립 모드
myDFPlayer.reset(); //슬립모드로부터 복귀 (동작안됨ㅠ)
myDFPlayer.enableDAC(); //음악파일에서 디코딩 상태로 설정
myDFPlayer.disableDAC(); //음악파일을 디코딩 하지 않음
myDFPlayer.outputSetting(true, 15); //이 모듈에서는 적용되지 않음

```

위 제어 모드 중 슬립모드는 모듈로 음악을 재생하지 않을 때 전력을 최소화하기 위한 모드입니다. 해당 모드로 진입한 뒤에는 음악재생 명령을 내려도 음악이 재생되지 않는데요. 다시 일반 모드로 복귀하기 위해서는 `reset()` 함수로 복귀하면 된다고 되어 있는데 해당 함수를 사용해 봤는데 일반모드로 돌아 오지 않더군요ㅠ 일반 모드로 복귀하기 위해

`myDFPlayer.outputDevice(DFPLAYER_DEVICE_SD);` 함수를 사용하니 다시 일반모드로 돌아오는 것을 발견했습니다.

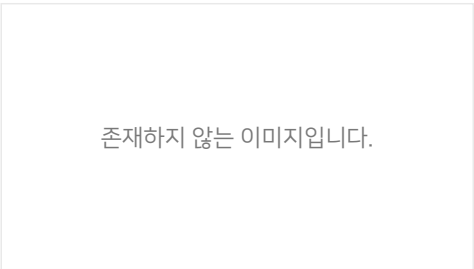
`enableDAC()` 함수는 DAC모드로 동작시킨다는 의미인데요. 기본 상태에서는 DAC모드입니다. 음악트랙을 재생중인 상태에서 `disableDAC()`; 함수를 실행하면 내부적으로는 음악이 재생은 진행되고 있는데 디코딩이 안되기 때문에 소리가 안나는 상태가 됩니다. 다시 `enableDAC()` 함수를 실행하면 그 시점부터 다시 디코딩이 진행됩니다. 이 함수는 MUTE 모드를 만들때 사용하면 좋을 것 같네요.

`ouptPutSetting()` 함수는 사용해 봤는데 별 효과가 없는 것 같습니다.

▶ 광고(Advertise) 모드

위 함수들 중 `advertise()`라는 함수와 `stopAdvertise()`라는 함수가 있는데요. 이 함수는 광고모드를 실행하거나 해제할 때 사용되는 함수 입니다. 이 모듈은 광고모드를 지원하는데요. 광고모드는 특정 음악 재생중에 광고모드를 실행하면 먼저 재생중 이던 음악을 일시중지 후 광고모드로 실행한 음악파일의 먼저 재생하고 다시 일시중지했던 음악파일의 재생하는 모드입니다.

전 음악을 재생하게 됩니다.

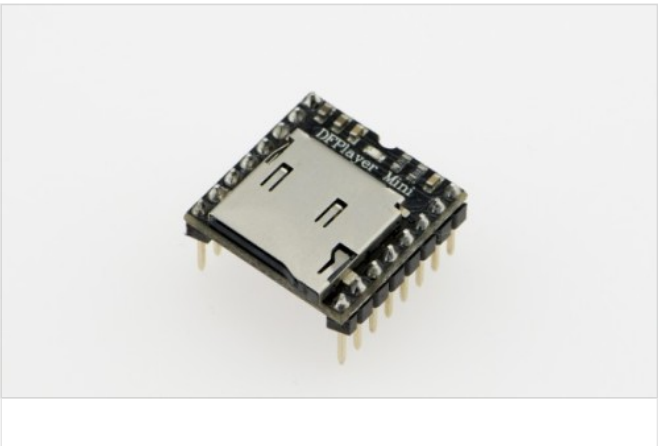


이 밖의 추가기능에 대해서는 데이터 시트를 통해 확인하면 될 것 같네요. 이제 재밌는 음악관련 프로젝트에 이 모듈을 잘 활용 할 수 있겠죠?^^



참고자료 및 더 알아보기

- 모듈설명서: https://wiki.dfrobot.com/DFPlayer_Mini_SKU_DFR0299



wiki.dfrobot.com

- 데이터 시트

DFPlayer Mini Manual.pdf

- 코드 참고: <https://www.electronics-lab.com/project/mp3-player-using-arduino-dfplayer-mini/>

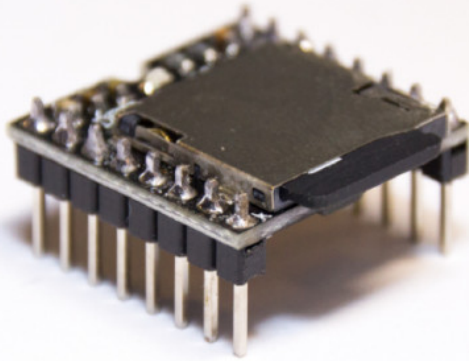
A photograph of a DFPlayer Mini MP3 module. It is a small, rectangular circuit board with a silver-colored metal shield on top. The board has several gold-colored pins along the bottom edge and a black plastic connector on the right side. The text "DFPlayer Mini" is printed on the board.

MP3 player using Arduino and DFPlayer mini - Electroni...

Hi guys, welcome to this tutorial. Today, we will build an mp3 pl...

www.electronics-lab.com

- MP3파일 넣는 방법 참고: <https://circuitjournal.com/how-to-use-the-dfplayer-mini-mp3-module-with-an-arduino>

A photograph of a DFPlayer Mini MP3 module, showing the underside of the silver shield. The board has several gold-colored pins along the bottom edge and a black plastic connector on the right side. The text "DFPlayer Mini" is printed on the board.

Wiring DFPlayer Mini (MP3 Module) to Arduino, Stereo...

Wiring DFPlayer Mini (MP3 Module) to Arduino, Stereo/Mono D...

circuitjournal.com

