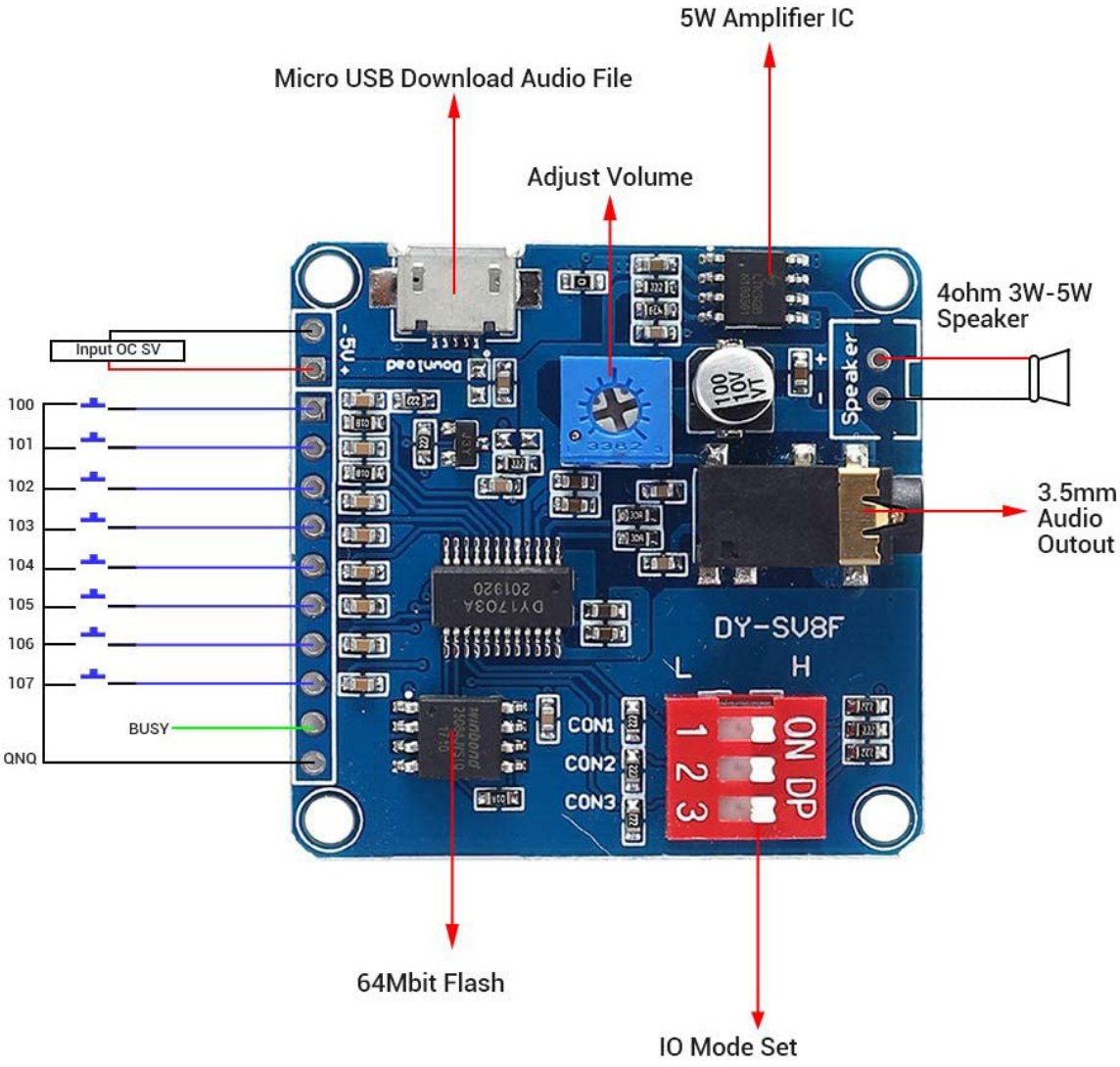
1. Beschreibung:
DY-SV8F ist ein intelligentes Sprach Modul, das von der Division unabhängig entwickelt wurde. Es integriert i/o Unter Abschnitt Triggering, Uart serielle Port-Steuerung, online Single-Bus serielle Port-Steuerung. On Board 5w Klasse d Verstärkerschaltung und kann 4Ohm 3 ~ 5w Lautsprecher direkt fahren. Unterstützung mp3, wav Decodierung Format. Unterstützung 64bbit (8mbyte) Flash-Speicher. Es kann den Computer anschließen, um Audiodateien über USB-Kabel zu aktualisieren. 3,5mm Audio-Interface, Micro-USB-Download-Schnittstelle-Tastenmodul in einem Modul.

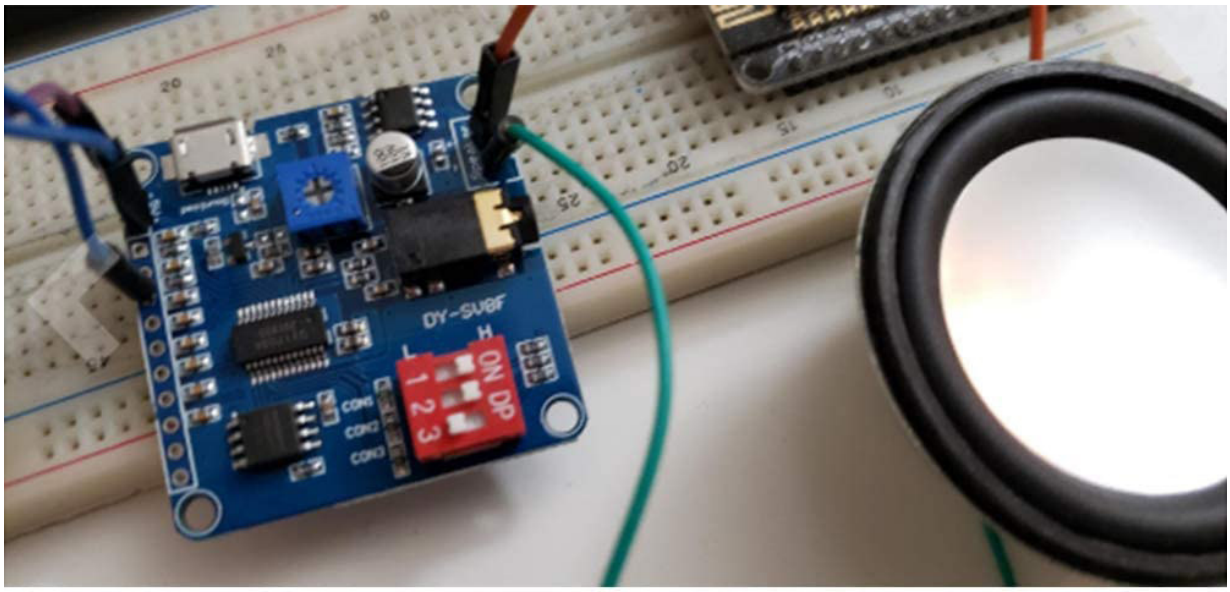
2. features:
1>. Unterstützung mp3 und wav Decoding Format.
2>. Unterstützung der Abtastfrequenz (kHz) : 8/11.025/12/16/22.05/24/32/44.1/48.
3>.24-bit DAC Ausgang, dynamische Palette Unterstützung 90dB, SNR Unterstützung 85dB.
4>. Unterstützung 64Bbit(8Mbyte)-speicher.
5>. Unterstützung Uart serielle Port Control Voice Broadcast-Funktion. Es kann die Wiedergabe, Pause, Auswahl, drehen nach oben und unten Lautstärke und andere Funktionen, die größte Auswahl von 65535 Songs steuern. Die Baudrate beträgt 9600 Bit/s.
6>. Unterstützung i/o Trigger Funktion, 8bit i/o Ports können 8 Musics oder 8 i/o Kombinationen auslösen, um 255 Lieder auszulösen.
7>. Unterstützung Online Single Bus Serial Port Control, die steuern können Wiedergabe, Pause, Auswahl, drehen auf und ab Lautstärke und andere Funktionen.
8>. Unterstützung 3 Konfiguration i/o für Modus Auswahl zu machen 7 Arbeitsmodus.
9>. gebaut in 5w Klasse d Verstärker Schaltung und kann 4Ohm 3 ~ 5w Lautsprecher direkt fahren.

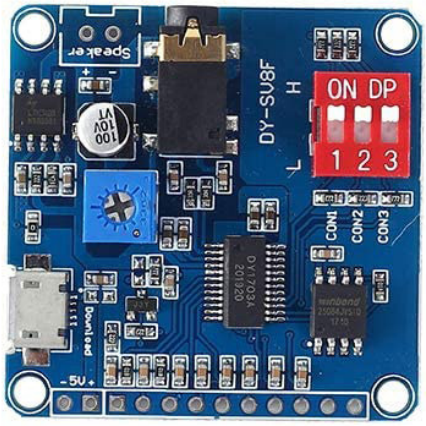
3.Parameter:
1>. Produktname: DY-SV8F Sprach Wiedergabemodul
2>. Produkt Nummer: DY-SV8F
3>. Arbeitsspannung: dc 5v
4>. Arbeitstemperatur Bereich: -20 ℃ ~ 85 ℃
5>. Arbeitsfeuchtigkeit Bereich: 0%-95% rh
6>. Größe: 35*35*4,5mm

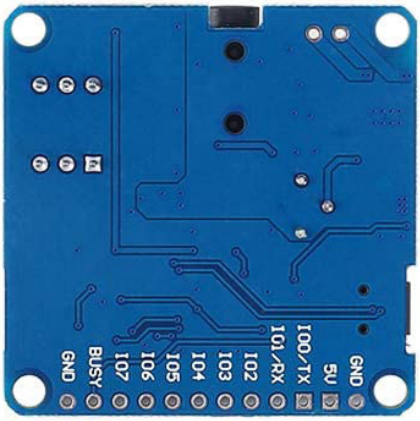
4. Paket:
1> 1pc DY-SV8F Sprach Wiedergabemodul

Front Display



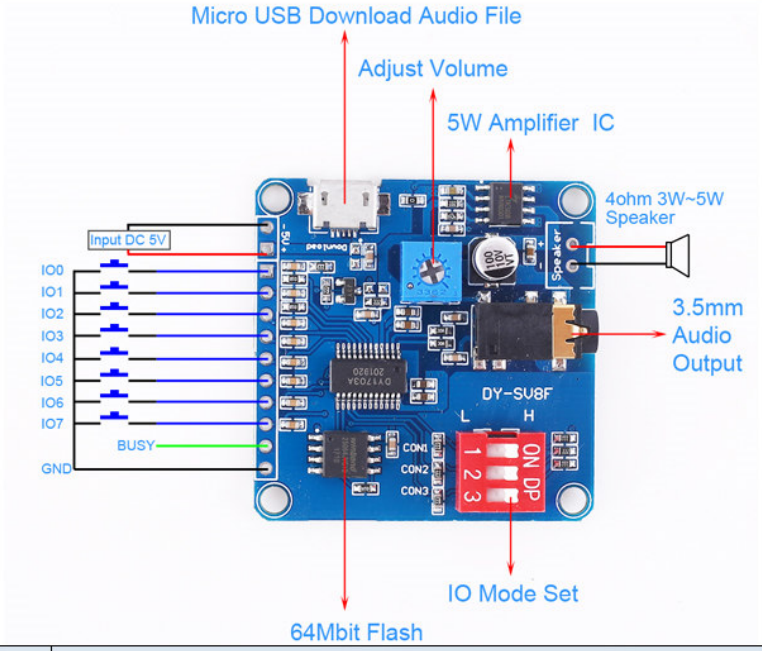

DY-SV8F
ON DP
1 2 3


Speaker
DY-SV8F
ON DP
1 2 3
CON1
CON2
CON3
Download
-5V+


GND
BUSY
IO7
IO6
IO5
IO4
IO3
IO2
IO1/RX
IO0/TX
5V
GND

Buyer show

Pin Definition



No.	Pin Name	Instruction
1	5V+	Work Voltage Positive Pole
2	5V-	Work Voltage Negative Pole
3	TXD/IO0	IO trigger mode is input IO0;UART mode is TX.
4	RXD/IO1	IO trigger mode is input IO1;UART mode is RX.
5	IO2	IO trigger mode input IO2.
6	IO3	IO trigger mode input IO3.
7	IO4/ONE_LINE	IO mode input IO4;One_Line mode data receiver pin.
8	IO5	IO trigger mode input IO5.
9	IO6	IO trigger mode input IO6.
10	IO7	IO trigger mode input IO7.
11	BUSY	Output low level signal(0V) when playing and output high(3.3V) after playing.
12	GND	Ground

Work Mode Configuration

Control Mode	Configuration Pin			I/O Function							
	CON3	CON2	CON1	IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0
I/O Integrated Mode 0	0	0	0	Key combination play, can play 2^8-1(255) Songs.							
I/O Integrated Mode 1	0	0	1	Level combination play, can play 2^8-1(255) Songs.							
I/O Independent Mode 0	0	1	0	Song8	Song7	Song6	Song5	Song4	Song3	Song2	Song1
I/O Independent Mode 1	0	1	1	Song8	Song7	Song6	Song5	Song4	Song3	Song2	Song1
UART Mode	1	0	0							RXD	TXD
One-Line Mode	1	0	0				TXD				
Standard MP3 Mode	1	0	1				RPT	EQ	P/P/MODE	PREV/V-	NEXT/V+

Note:

1>. "key combination play" : Return to the original high level after the corresponding level from I/O0-I/O7 output, similar to the key triggered once.Similar instantaneous switch.

2>. "Level combination play" :The trigger signal remains the same, similar to a self-locking switch.

3>.The difference between "I/O Integrated/Independent Mode 0" and "I/O Integrated/Independent Mode 1" :Mode 0 will continue playing the current song to the end after release level .Mode 1 will stop playing immediately after release level.

I/O Integrated Mode 0 (Key combination playing).

Note: the song must be named for 5bit.

IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	Song
1	1	1	1	1	1	1	0	00001.mp3
1	1	1	1	1	1	0	1	00002.mp3
1	1	1	1	1	1	0	0	00003.mp3
1	1	1	1	1	0	1	1	00004.mp3
1	1	1	1	1	0	1	0	00005.mp3
1	1	1	1	1	0	0	1	00006.mp3
1	1	1	1	1	0	0	0	00007.mp3
.....	.....	.....	.....	.....	.....	.....	.....	.....
0	0	0	0	0	0	0	0	00255.mp3

It will stop playing current song to the end after I/O0-7 release input signal (return to high) at 'I/O Integrated Mode 0'. It will playing new song when get new input signal during playing and stop after end of song. It will play repeatedly if keep input. Busy pin will output valid signal(High) during playing.Music control as following:

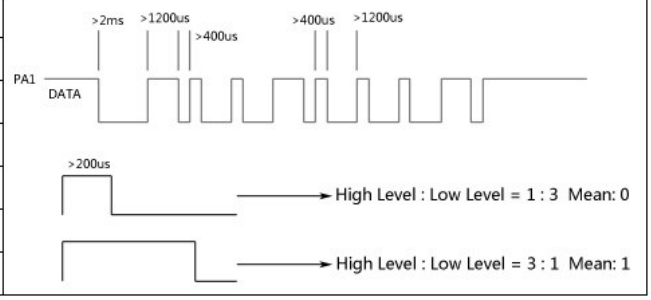
I/O Integrated Mode 1 (Level combination playing)									
IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	Song	It will keep playing current song when get trigger signal.It will stop playing immediately after release level.Busy pin will output valid signal(High) during playing.
1	1	1	1	1	1	1	0	00001.mp3	
1	1	1	1	1	1	0	1	00002.mp3	
1	1	1	1	1	1	0	0	00003.mp3	
1	1	1	1	1	0	1	1	00004.mp3	
1	1	1	1	1	0	1	0	00005.mp3	
1	1	1	1	1	0	0	1	00006.mp3	
1	1	1	1	1	0	0	0	00007.mp3	
.....	.....	.....	.....	.....	.....	.....	.....	.....	
0	0	0	0	0	0	0	0	00255.mp3	
I/O Independent Mode 0 (Key independent controlling)									
IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	Song	I/O0-I/O7 independently controls 8 songs.It will stop playing current song to the end after I/O0-7 release input signal(return to high);It will playing new song when get new input signal during playing and stop after end of song;It will play repeatedly if keep input;Busy pin will output valid signal(High) during playing.
1	1	1	1	1	1	1	0	00001.mp3	
1	1	1	1	1	1	0	1	00002.mp3	
1	1	1	1	1	0	1	1	00003.mp3	
1	1	1	1	0	1	1	1	00004.mp3	
1	1	1	0	1	1	1	1	00005.mp3	
1	1	0	1	1	1	1	1	00006.mp3	
1	0	1	1	1	1	1	1	00007.mp3	
0	1	1	1	1	1	1	1	00008.mp3	
I/O Independent Mode 1 (Level independent controlling)									
IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	Song	I/O0-I/O7 independently controls 8 songs.It will keep play repeatedly specify the triggered song.It will stop playing immediately after release level.Busy pin will output valid signal(High) during playing.
1	1	1	1	1	1	1	0	00001.mp3	
1	1	1	1	1	1	0	1	00002.mp3	
1	1	1	1	1	0	1	1	00003.mp3	
1	1	1	1	0	1	1	1	00004.mp3	
1	1	1	0	1	1	1	1	00005.mp3	
1	1	0	1	1	1	1	1	00006.mp3	
1	0	1	1	1	1	1	1	00007.mp3	
0	1	1	1	1	1	1	1	00008.mp3	

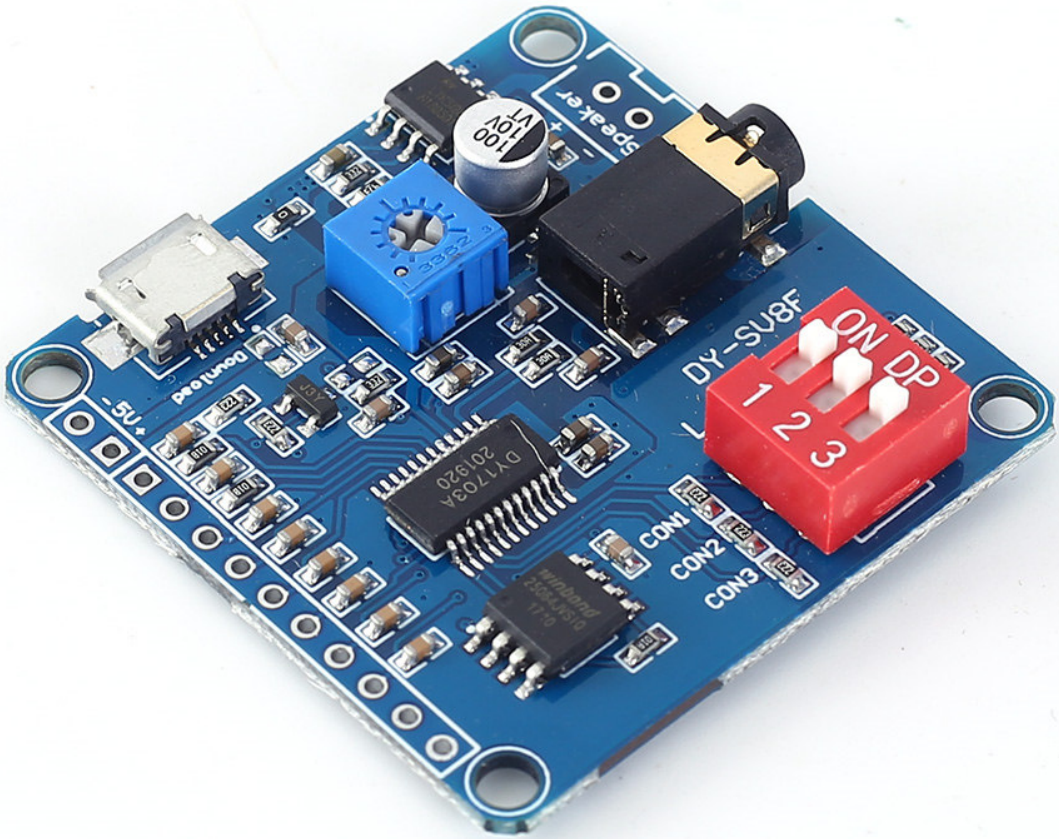
UART Mode					
Communication Format					
Adopt full duplex serial port communication. Baud rate 9600, data bits 8, stop bit 1, check bit N.					
Start Code	Command Type	Data Length (n)	Data 1	Data n	Check Bit (SM)
Command Code: fixed to 0xAA.					
Command Type: used to distinguish the type of command.					
Data Length: the number of bytes of data in an command.					
Data: Relevant data in command, when length of data is 1, means there is only CMD and no data bits.					
Check Bit: Low 8 bits of sum of all bytes. that is, When start code and data are added, take out low 8 bits.					
Data format: Sent data or command, high 8-bit data is in front, low 8-bit is in the back.					
Communication Protocol					
The following is a data definition for the return and identification of the chip.					
A. Playing State definition: the system is on the stop state when power on.					
	00(stop)	01(play)	02(pause)		
B. Disk character definition: it is stopped after the switch disk.					
	USB:00	SD:01	FLASH:02	NO_DEVICE: FF	
C. Volume: the volume is 31grades, 0-30.The default is 20grade.					
D. Play mode: the default is the single stop when power on.					
Cycle for all songs (00) : play the whole songs in sequence and play it after the play.					
Single cycle (01) : play the current song all the time.					
Single stop (02) : Only play current song once and then stop.					
Random play (03) : random play.					
Directory loop (04) :Play in current folder in order, then play by play.Directory don't contain subdirectory.					
Directory random (05): random play in the current folder, and directory does not contain subdirectory.					
Directory order play(06):Play current folder in order & stop after play.Directory not include subdirectory.					
Sequential play (07) : play the whole songs in order and stop after it is played.					
E. EQ definition: the default EQ is NORMAL(00).					
	NORMAL(00)	POP(01)	ROCK(02)	JAZZ(03)	CLASSIC(04)
F. Composition play definition: combination play is combined by filename. The file requirements are stored under the "ZK" file. You can change the name of the file you want to combine to two bytes, which is generally recommended as a number. Such as: 01. Mp3, 02. Mp3.					

UART Communication Command		
Control Command		
Command	Command code	Return
Play	AA 02 00 AC	None
Pause	AA 03 00 AD	None
Stop	AA 04 00 AE	None
Previous	AA 05 00 AF	None
Next	AA 06 00 B0	None
Volume +	AA 14 00 BE	None
Volume -	AA 15 00 BF	None
Previous file	AA 0E 00 B8	None
Next file	AA 0F 00 B9	None
Stop playing	AA 10 00 BA	None
Query Command		
Command	Command Code	Return
Query play status	AA 01 00 AB	AA 01 01, play status, SM
Query current online drive	AA 09 00 B3	AA 09 01, drive, SM
Query current play drive	AA 0A 00 B4	AA 0A 01, drive, SM
Query Number of songs	AA 0C 00 B6	AA 0C 02S.N.H S.N.L SM
Query current song	AA 0D 00 B7	AA 0D 02 S.N.H S.N.L SM
Query folder directory song	AA 11 00 BB	AA 11 02 S.N.H S.N.L SM
Query folder Number of songs	AA 12 00 BC	AA 12 02 S.N.H S.N.L SM

UART Communication Command					
Control Command			Query Command		
Command	Command Code	Return	Command	Command code	Return
Play	AA 02 00 AC	None	Query play status	AA 01 00 AB	AA 01 01, play status, SM
Pause	AA 03 00 AD	None	Query current online drive	AA 09 00 B3	AA 09 01, drive, SM
Stop	AA 04 00 AE	None	Query current play drive	AA 0A 00 B4	AA 0A 01, drive, SM
Previous	AA 05 00 AF	None	Query Number of songs	AA 0C 00 B6	AA 0C 02S.N.H S.N.L SM
Next	AA 06 00 B0	None	Query current song	AA 0D 00 B7	AA 0D 02 S.N.H S.N.L SM
Volume +	AA 14 00 BE	None	Query folder directory song	AA 11 00 BB	AA 11 02 S.N.H S.N.L SM
Volume -	AA 15 00 BF	None	Query folder Number of songs	AA 12 00 BC	AA 12 02 S.N.H S.N.L SM
Previous file	AA 0E 00 B8	None			
Next file	AA 0F 00 B9	None			
Stop playing	AA 10 00 BA	None			

Setting Command			
Command	Command code	Return	Remark
Set Volume	AA 13 01 VOL SM	None	VOL:0x00-0xFF
Set Loop mode	AA 18 01 Loop-mode SM	None	Loop-mode:0x00-0x07
Set Cycle times	AA 19 02 H L SM	None	H:0x00-0xFF L:0x00-0xFF
Set EQ	AA 1A 01 EQ SM	None	EQ:0x00-0x04
Specified Song	AA 07 02 S.N.H S.N.LSM	None	S.N.H:0x00-0xFF S.N.L:0x00-0xFF
Specified Path	AA 08 Length Drive Path SM	None	Length:0x00-0xFF
			Drive:0x00-0xFF
			Path:0x00-0xFF
Switch Specified Drive	AA 0B 01 Drive SM	None	Drive:0x00-0xFF
Specified song to be interplay	AA 16 03 Drive S.N.H S.N.L SM	None	Drive:0x00-0xFF
			S.N.H:0x00-0xFF
			S.N.L:0x00-0xFF
Specified path to be interplay	AA 17 Length Drive Path SM	None	Length:0x00-0xFF
			Drive:0x00-0xFF
			Path:0x00-0xFF
Select but no play	AA 1F 02 S.N.H S.N.L SM	None	S.N.H:0x00-0xFF S.N.L:0x00-0xFF

One_line Single Bus Mode		
Command(HEX)	Function	Note
0x00	No. 0	The number 0-9 has corresponding functions, such as selecting music, setting the volume, setting EQ, setting cycle mode, setting channel, setting the repertoire, and sending the digital at first and then send function command.
0x01	No. 1	
0x02	No. 2	
0x03	No. 3	
0x04	No. 4	
0x05	No. 5	
0x06	No. 6	
0x07	No. 7	
0x08	No. 8	
0x09	No. 9	
0x0A	Number reset	Sent the number of Cleared
0x0B	Confirm choosing song	Cooperate with Numbers to achieve.
0x0C	Volume setting	
0x0D	EQ setting	
0x0E	Loop mode setting	
0x0F	Channel setting	
0x10	Interplay song setting	
0x11	Play	Note: "selection" and "interplay" are played according to the track name, for example, the track is named "00123. Mp3", and the selected data is "0x01", "0x02" "0x03" "0x0B", and the selection is completed.
0x12	Pause	
0x13	Stop	
0x14	Previous	
0x15	Previous directory	
0x16	Next directory	
0x17	SD card selection	
0x18	SD card selection	
0x19	U disk selection	
0x1A	FLASH selection	
0x1B	System sleep	
0x1C	Stop Playing	


Speaker
100uF 10V
DY-SU8F
ON DP
1 2 3
CON1
CON2
CON3
-5V+
Download
DY1703A
201920

Abstracted UART Control of DY-XXXX mp3 modules

Diese Bibliothek abstrahiert alle im Handbuch beschriebenen Features in einer C++-Klasse.

Diese Bibliothek unterstützt das ONE_Line-Protokoll nicht, weitere Informationen.

Obwohl alle Funktionen implementiert sind und theoretisch funktionieren sollten, werden nur die im Beispielverzeichnis getestet.

Bitte erstellen Sie ein Problem, wenn Sie Probleme haben.

Diese Bibliothek wurde hardwareunabhängig geschrieben. Das heißt, es sollte auf jedem Gerät mit serieller Schnittstelle funktionieren, z. alle Arduino-, Espressif-, ARM-basierten Boards, wahrscheinlich sogar jeder Computer.

Für Arduino und ESP-IDF sind Hardware Abstraction Layers (HAL) enthalten, für andere Boards müssen Sie selbst einen bereitstellen (PR ist willkommen!).

Hinweis zum Upgrade von Version 3.x.x auf Version 4.x.x

TL;DR: Alle Vorkommen von Enum-Konstanten von DY::Device, DY::PlayState, DY::Eq, DY::PlayMode und D::Previous Dir von Großbuchstaben auf CamelCase ändern, z. B.: DY::Device:: FLASH zu DY::Device::Flash.

Eines der Probleme mit C++, von denen ich auf die harte Tour erfahren habe, ist, wie der Namespace trotz Namespaceing immer noch verschmutzt werden kann. Da Makros vom Präprozessor verarbeitet werden, wird jede Erwähnung eines Makros in der Quelle, selbst wenn es offensichtlich als Code gedacht ist (gut für einen Menschen), als Aufruf eines Makros interpretiert.

Die Header eingebetteter Geräte sind stark von Makros abhängig und haben im Allgemeinen kein Präfix. Daher berichteten Benutzer von Kollisionen zwischen Konstanten wie USB (es ist eigentlich Teil einer Enum-Klasse: DY::Device::USB, was die Sache noch frustrierender macht) und einem Makro, das für ein Board definiert ist, das einen USB-Anschluss hat, auch USB genannt.

Der einzige Weg, dies zu umgehen, besteht darin, die soziale Konvention (von der wohl bereits abgeraten wurde) der Benennung von Konstanten in Großbuchstaben fallen zu lassen. Der Compiler kümmert sich nicht um diese Konvention, es geschieht normalerweise nur aus Gründen der Lesbarkeit.

Module sollten funktionieren (nicht vollständig)

Model name	Capacity	SD Card support	Amplifier	Voltage	Tested
DY-SV17F	32Mbit	No	3-5W(4Ω/8Ω)	5VDC	**Yes**
DY-SV8F	64Mbit	No	3-5W(4Ω/8Ω)	5VDC	No
DY-HV20T	NA	Yes, Max. 32GB	3-5W(4Ω/8Ω)	5VDC	No
DY-HV8F	8Mbit	No	10W(8Ω)/20W(4Ω)	6-35VDC	No
DY-HV20T	NA	Yes, Max. 32GB	10W(8Ω)/20W(4Ω)	6-35VDC	No
DY-SV5W	NA	Yes, Max. 32GB	3-5W(4Ω/8Ω)	5VDC	**Yes**

HINWEIS: Ich kann nicht garantieren, dass Ihr Board mit der Bibliothek funktioniert. Auch nicht, dass eine bestimmte Funktion funktioniert. Ich habe den DY-SV17F zum Zeitpunkt des Schreibens nur zum Testen in meinem Besitz. Wenn etwas nicht funktioniert, machen Sie ein Issue und/oder senden Sie mir eine Pull-Anfrage.

Verdrahtung des Moduls

Wenn Sie eine Platine mit DIP-Schaltern haben, stellen Sie CON3 auf on, CON1 und CON2 sollten aus bleiben. Wenn Ihre Platine keine DIP-Schalter hat (z. B. DY-SV17F-Platine), müssen Sie 3 Widerstände von 10 KOhm, einen von jedem CON#-Pin, anschließen an:

CON pin	Connect to	Via
CON1	GND	10KOhm
CON2	GND	10KOhm
CON3	3.3V	10KOhm

Der 3,3-V-Pin wird von der Platine freigelegt, sodass Sie ihn nicht selbst bereitstellen müssen.

Stellen Sie außerdem diese Verbindungen her:

Pin	Connect to	Via
V?	V+ (voltage depends on module)	
GND	GND	
IO0/TX	MCU RX	1KOhm if using a 5V board
IO1/RX	MCU TX	1KOhm if using a 5V board
SPK+	Speaker positive lead	
SPK-	Speaker negative lead	

MCU sollte Ihr Board oder Mikroprozessor sein, z. ein Arduino-Board.

Beachten Sie, dass die Logikpegel laut Modulhandbuch 3,3 V betragen oder zumindest durch einen 1-KOhm-Widerstand begrenzt sein sollten, da Sie sonst das Modul beschädigen könnten. Einige Module scheinen Widerstände auf die Platine gelötet zu haben. Überprüfen Sie unbedingt das Datenblatt Ihres Boards.

HAL

Arduino tl;dr; Wenn Sie Arduino verwenden, überspringen Sie dieses Kapitel und gehen Sie zu Arduino.

ESP32 tl;dr; Wenn Sie ESP-IDF verwenden, überspringen Sie dieses Kapitel und gehen Sie zu ESP-IDF.

Da die Bibliothek hardwareunabhängig ist, müssen Sie möglicherweise einen Hardware Abstraction Layer (HAL) hinzufügen, der die serielle Schnittstelle einrichtet und mindestens 2 Funktionen implementiert `serialWrite()` and `serialRead`(), es ist ein HAL für Arduino enthalten, hier ist eine vereinfachte Version davon:

```
// player.hpp
#include <Arduino.h>
#include "DYPlayer.h"
namespace DY {
  class Player: public DYPlayer {
    public:
      HardwareSerial *port;
      Player();
      Player(HardwareSerial* port);
      void begin();
      void serialWrite(uint8_t *buffer, uint8_t len);
      bool serialRead(uint8_t *buffer, uint8_t len);
  };
}
```

```
//player.cpp
#include "player.hpp"
#include "DYPlayerArduino.h"

namespace DY {
  Player::Player() {
    this->port = &Serial;
  }
  Player::Player(HardwareSerial* port) {
    this->port = port;
  }
  void Player::begin() {
    port->begin(9600);
  }
  void Player::serialWrite(uint8_t *buffer, uint8_t len) {
    port->write(buffer, len);
  }
  bool Player::serialRead(uint8_t *buffer, uint8_t len) {
    // Serial.setTimeout(1000); // Default timeout 1000ms.
    if(port->readBytes(buffer, len) > 0) {
      return true;
    }
    return false;
  }
}
```

Hinweis: Dieser Arduino HAL ist vereinfacht, der enthaltene HAL macht auch SoftwareSerial, was eine gewisse Komplexität hinzufügt, die meiner Meinung nach in einem Beispiel wie diesem nicht sein sollte. Den echten HAL finden Sie hier.

Steps:

1. Definieren Sie eine Klasse, die die Klasse DY::DYPlayer erweitert.
2. Definieren Sie Konstruktoren, die den seriellen Port einrichten. Auf einigen Plattformen müssen Sie Ihren seriellen Anschluss einrichten, nachdem einige andere Dinge initialisiert wurden, z. auf Arduino, dann definieren Sie eine zusätzliche `DY::Player::begin()` (or e.g. `DY::Player::init()`) um die Initialisierung abzuschließen.
3. Definieren Sie Funktionen für `serialWrite()` und `serialRead()` entsprechend dem Board und dem Framework, das Sie verwenden.

Speichernutzung

Diese Bibliothek verwendet so wenig Speicher wie möglich, um gut mit Mikrocontrollern mit kleinem RAM zu spielen, wie z. B. Atmega328 (in vielen Arduino-Boards verwendet), das über 2K RAM verfügt. Um die Speicherauslastung niedrig zu halten, vermeiden Sie die Verwendung von Funktionen, die char *path-Argumente annehmen. Wenn Sie Sounds nicht nach Dateinamen abspielen möchten, können Sie den Rest dieses Kapitels überspringen. Wenn ja, lesen Sie weiter. Die char *path-Argumente verbrauchen natürlich immer mehr RAM als die uint16_t-Argumente, aber dies wird durch eine seltsame Anforderung der Player-Module noch verstärkt. D.h. die Pfade zu Dateien auf Flash/SD-Karte müssen anders als üblich definiert werden, z.B. /SONGS/IN/A/PATH/00001.MP3 sollte wie folgt angegeben werden: /SONGS*/IN*/A*/PATH*/00001*MP3Analysing this:

216 / 5.000

Übersetzungsergebnisse

Übersetzung

• Pfade enden normalerweise mit /, aber ein zusätzliches
* ist erforderlich;
• mit Ausnahme der Root-Ebene.
• Der Punkt in vor der Erweiterung sollte ebenfalls durch * ersetzt werden.
• Der neue Pfad ist 4 Bytes länger als der angegebene Pfad.

Die Konvertierung wird von der Bibliothek durchgeführt, aber das bedeutet, dass der Pfad zweimal im Speicher zugewiesen wird, einmal von Ihnen, einmal von der Bibliothek und letztere muss mehr Kapazität haben (in diesem Fall 4 Bytes). Die Bibliothek kann den Speicher der zweiten Zeichenfolge auf zwei Arten halten: im Heap- oder im Stack-Speicher. Stack-Speicher ist nicht dynamisch, d.h. die erforderliche Menge an Bytes sollte zur Kompilierzeit bekannt sein, was bedeutet, dass bereits mehr als die erwartete Menge an Bytes reserviert werden sollte, was verschwenderisch ist. Abgesehen davon, dass er verschwenderisch ist, kann der Pfad sehr kurz sein (höchstwahrscheinlich z. B. /00001.MP3 oder so etwas wie /SFX/00001.MP3), oder er könnte sehr lang sein. Das Ablegen des Pfads in dynamisch zugewiesenem Heap-Speicher behebt all das, die Bibliothek kann die Menge von / im Pfad zählen und eine Variable genau lang genug für den konvertierten Pfad machen. Wir sollten uns jedoch immer vor Heap-Fragmentierung in Acht nehmen. Kurz gesagt, um Speicher zuzuweisen, muss ein zusammenhängender Block davon verfügbar sein. Das Zuweisen von Speicherblöcken und das anschließende Freigeben hinterlässt Löcher im Speicher, die möglicherweise zu klein sind, um sie später erneut zu verwenden. Das wird nach und nach zu Problemen führen. Möglicherweise müssen Sie Ihr Gerät nach einigen Minuten, Stunden oder Tagen zurücksetzen, da das Programm keinen Heap-Speicher mehr zuweisen kann. Daher reserviert die Bibliothek standardmäßig Stapelspeicher. Der Betrag basiert auf einigen Annahmen: Im Handbuch der Soundmodule steht, dass Pfade und Dateinamen bis zu 8 Zeichen lang sein dürfen.

• Die Bibliothek geht davon aus, dass Sie nicht mehr als 2 Verzeichnisse verschachteln.
• Erweiterungen können .MP3 oder .WAV sein, also immer 4 Bytes.

Also kommen wir zu:`/ dir */ dir */ file * ext`

```
1 + 8 + 2 + 8 + 2 + 8 + 1 + 4 = 34
```

Runden wir das auf 40 auf und Sie könnten sogar noch mehr Verschachtelungen haben, solange die Verzeichnisnamen klein genug sind.

Die Bibliothek definiert daher DY_PATH_LEN als 40, Sie können dies überschreiben, wenn Sie mehr benötigen oder wenn Sie ein paar Bytes an wertvollem Speicher sparen möchten. Beachten Sie, dass 40 die maximale Länge des Pfads nach der Konvertierung in das vom Modul benötigte Funky-Format ist. Die Anzahl der Bytes, die Sie mit dieser Voreinstellung verwenden können, ist auf 36 festgelegt. Wenn Sie über ein leistungsfähigeres Gerät verfügen und/oder virtuellen Speicher verwenden

können, können Sie alternativ DY_PATHS_IN_HEAP so definieren, dass Heap-Speicher anstelle von reserviertem Stack-Speicher verwendet wird.

HINWEIS: Auf Arduino können Sie Ihre Zeichenfolgen in F() umschließen, um dem Compiler mitzuteilen, dass die Zeichenfolge im Flash gespeichert werden soll, im Gegensatz zu RAM (Standard), wodurch Sie noch mehr RAM sparen.

Arduino

Da dies enthalten ist, können Sie auf Arduino einfach die einfügen `DYPlayerArduino.h` Header-Datei und verwenden Sie das Modul.

Sie können jeden seriellen Anschluss auf der Platine verwenden, Sie müssen ihn an den weiterleiten `begin()` function. Like this:

```
DY::Player player(&Serial2);
```

Informationen zur Verwendung des Arduino HAL finden Sie unter PlaySoundByNumber.ino.

ESP-IDF

Da dies enthalten ist, können Sie auf Arduino einfach die Header-Datei DYPlayerESP32.h einfügen und das Modul verwenden.
Sie können jeden seriellen Anschluss auf der Platine und alle 2 Pins für TX und RX verwenden.
Beispiel: Benutzer UART_NUM_2 auf Pins TX: 18, RX: 19:`DY::Player player(UART_NUM_2, 18, 19);`

Ein Beispiel finden Sie hier:PlaySounds.cpp.

API

Die Bibliothek abstrahiert das Senden binärer Befehle an das Modul. Es gibt ein Handbuch für das Modul, das im Internet herumschwirrt, von dem ich nicht sicher bin, ob es urheberrechtlich geschützt ist oder nicht, also werde ich es hier nicht hochladen oder darauf verlinken. Googlen Sie einfach das DY-SV17F-Handbuch und das Datenblatt, und Sie werden es wahrscheinlich finden. Das Handbuch benennt keinen der Befehle eindeutig, also habe ich mir selbst Namen für Klassenmethoden ausgedacht. Das Handbuch enthält eine Befehlsbeschreibung, also habe ich diese in die Liste unten aufgenommen. Insbesondere die "check"-Befehle werden als "get"-Befehle bezeichnet, um anzuzeigen, dass sie Informationen von dem Modul abrufen. Einstellungen, die Sie ändern können, werden durch "Set"-Befehle angewendet.

Methoden festlegen

Das Einstellen oder Abspielen eines Tons wird nicht vom Modul bestätigt. Wenn Sie also eine Bestätigung benötigen, dass das Modul das tut, was erwartet wird, müssen Sie eine der get-Funktionen verwenden, um zu überprüfen, ob es das tut, was Sie ihm gesagt haben. Z.B. Wenn Sie einen Ton abspielen, können Sie folglich anrufen `DY::DYPlayer::getPlayingSound()` um zu überprüfen, ob es funktioniert hat.

Get methods

Get-Methoden senden einen Befehl an das Modul und warten auf eine Antwort. Wenn keine Antwort empfangen wird, läuft ein Timeout ab und anschließend wird 0 für Tonzählungen zurückgegeben.

Command/method list

Befehl (aus dem Handbuch)	byte	Method name
Überprüfen Sie den Wiedergabestatus	0x01	DY::PlayState::play_state_t **DY::DYPlayer::checkPlayState**
Play	0x02	void **DY::DYPlayer::play**
Pause	0x03	void **DY::DYPlayer::pause**
Stop	0x04	void **DY::DYPlayer::stop**
Vorherige Musik	0x05	void **DY::DYPlayer::previous**
Nächste Musik	0x06	void **DY::DYPlayer::next**
Angegebene Musik abspielen	0x07	void **DY::DYPlayer::playSpecified**
Angegebenes Gerät und Spielpfad	0x08	void **DY::DYPlayer::playSpecifiedDevicePath**
Überprüfen Sie das aktuelle Wiedergabegerät	0x0a	DY::Device::device_t **DY::DYPlayer::getPlayingDevice**
Zum ausgewählten Gerät wechseln	0x0b	void **DY::DYPlayer::setPlayingDevice**
Überprüfen Sie die Nummer aller Musik	0x0c	uint16_t **DY::DYPlayer::getSoundCount**
Überprüfen Sie die aktuelle Musik	0x0d	uint16_t **DY::DYPlayer::getPlayingSound**
Vorheriges Ordnerverzeichnis (erstes)	0x0e	void **DY::DYPlayer::previousDir**
Vorheriges Ordnerverzeichnis (letztes)	0x0f	void **DY::DYPlayer::previousDir**
Beenden Sie das Spielen	0x10	void **DY::DYPlayer::stopInterlude**
Überprüfen Sie die erste Musik im Ordner	0x11	uint16_t **DY::DYPlayer::getFirstInDir**
Überprüfen Sie die Anzahl der Musiktitel im Ordner	0x12	uint16_t **DY::DYPlayer::getSoundCountDir**
Lautstärkeeinstellung	0x13	void **DY::DYPlayer::setVolume**
Lautstärke+	0x14	void **DY::DYPlayer::volumeIncrease**
Lautstärke-	0x15	void **DY::DYPlayer::volumeDecrease**
Angegebene Datei zum Zwischenspiel auswählen	0x16	void **DY::DYPlayer::interludeSpecified**
Angegebenen Pfad zum Zwischenspiel auswählen	0x17	void **DY::DYPlayer::interludeSpecifiedDevicePath**
Einstellung des Zyklusmodus	0x18	void **DY::DYPlayer::setCycleMode**
Zykluszeiten einstellen	0x19	void **DY::DYPlayer::setCycleTimes**
EQ einstellen	0x1a	void **DY::DYPlayer::setEq**
Kombinationsspieleinstellung	0x1b	void **DY::DYPlayer::combinationPlay**
Kombinationsspiel beenden	0x1c	void **DY::DYPlayer::endCombinationPlay**
Datei auswählen, aber nicht abspielen	0x1f	void **DY::DYPlayer::select**

UART-Befehle werden gebildet von:

`aa [cmd] [len] [byte_1..n] [crc]`

Wo:

• aa Einen Befehl starten

• [cmd] Ein Befehl aus der Liste, z. 01, um den Wiedergabestatus zu überprüfen.

• [len] Die Länge der als Argument gesendeten Bytemenge.

• [byte_1..n] Alle Bytes werden als Argumente verwendet, einige Argumente sind 1 Byte lang, wie zum Beispiel für die Auswahl des Speichergeräts, einige sind 2 Byte lang, wie zum Beispiel die Angabe eines Songs nach Nummer (uint16_t-Wert, zwischen 0 und 65535), manchmal es ist eine Kombination oder ein Weg.

• [crc] Die Summe aller Bytes im gesamten Befehl als uint8_t.

API documentation

`DY::play_state_t` DY::DYPlayer::checkPlayState(..)

Überprüfen Sie den aktuellen Spielstand, jederzeit abrufbar.

	Type	Name	Description
return	`DY::play_state_t`		Play status: A `DY::PlayState` e.g DY::PlayState::Stopped, DY::PlayState::Playing, etc

`void` DY::DYPlayer::play(..)

Spielen Sie die aktuell ausgewählte Datei von Anfang an ab.

`void` DY::DYPlayer::pause(..)

Setzen Sie den Wiedergabestatus auf Pause.

`void` DY::DYPlayer::stop(..)

Setzen Sie den Wiedergabestatus auf gestoppt.

`void` DY::DYPlayer::previous(..)

Wiedergabe der vorherigen Datei.

`void` DY::DYPlayer::next(..)

Nächste Datei abspielen.

`void` DY::DYPlayer::playSpecified(..)

Spielen Sie eine Sounddatei nach Nummer ab, die Nummer wird als 2 Byte gesendet.

	Type	Name	Description
param	`uint16_t`	`number`	number of the file, e.g. `1` for `00001.mp3`

`void` DY::DYPlayer::playSpecifiedDevicePath(..)

Spielen Sie eine Sounddatei nach Gerät und Pfad ab. Der Pfad kann aus bis zu 2 verschachtelten Verzeichnissen mit einer Länge von 8 Byte und einem Dateinamen mit einer Länge von 8 Byte ohne die Erweiterung mit einer Länge von 4 Byte bestehen. Wenn Ihre Verzeichnisnamen kürzer sind, können Sie mehr Verschachtelungen verwenden. Verwenden Sie für Ihre Pfade nicht mehr als 36 Byte. Wenn Sie mehr benötigen, sehen Sie in der Readme nach, Kapitel: Speichernutzung.

	Type	Name	Description
param	`DY::device_t`	`device`	device A `DY::Device member` e.g `DY::Device::Flash` or `DY::Device::Sd`
param	`char`	`path`	Pfadzeiger auf den Pfad der Datei (asbsolute)

DY::device_t DY::DYPlayer::getPlayingDevice(..)

Holen Sie sich das Speichergerät, das derzeit zum Abspielen von Sounddateien verwendet wird.

	Type	Name	Description
return	DY::device_t		a DY::Device member e.g DY::Device::Flash or DY::Device::Sd

void DY::DYPlayer::setPlayingDevice(..)

Stellen Sie die Gerätenummer ein, die das Modul verwenden soll. Versucht, das Gerät einzustellen, aber es wird keine Garantie gegeben, verwenden `getDevice()`, um das aktuelle Speichergerät zu überprüfen.

	Type	Name	Description
param	DY::device_t	device	device A DY::Device member e.g DY::Device::Flash or DY::Device::Sd

uint16_t DY::DYPlayer::getSoundCount(..)

Rufen Sie die Menge der Sounddateien auf dem aktuellen Speichergerät ab.

	Type	Name	Description
return	uint16_t		Anzahl Sounddateien

uint16_t DY::DYPlayer::getPlayingSound(..)

Holen Sie sich die aktuell wiedergegebene Datei nach Nummer.

	Type	Name	Description
return	uint16_t		Nummer der aktuell wiedergegebenen Datei

void DY::DYPlayer::previousDir(..)

Wählen Sie das vorherige Verzeichnis und starten Sie die Wiedergabe des ersten oder letzten Titels.

	Type	Name	Description
param	playDirSound_t	song	song Play DY::PreviousDir::FirstSound o DY::PreviousDir::LastSoun

uint16_t DY::DYPlayer::getFirstInDir(..)

Rufen Sie die Nummer des ersten Songs im aktuell ausgewählten Verzeichnis ab.

	Type	Name	Description
return	uint16_t		Nummer des ersten Titels im aktuell ausgewählten Verzeichnis

uint16_t DY::DYPlayer::getSoundCountDir(..)

Rufen Sie die Anzahl der Sounddateien im aktuell ausgewählten Verzeichnis ab. HINWEIS: Dateien in Unterverzeichnissen ausschließen.

	Type	Name	Description
return	uint16_t		Anzahl der Sounddateien im aktuell ausgewählten Verzeichnis

void DY::DYPlayer::setVolume(..)

Stellen Sie die Wiedergabelautstärke zwischen 0 und 30 ein. Standardlautstärke, wenn nicht eingestellt: 20.

	Type	Name	Description
param	uint8_t	volume	einzustellende Lautstärke (0-30

void DY::DYPlayer::volumeIncrease(..)

Die Lautstärke erhöhen.

void DY::DYPlayer::volumeDecrease(..)

Verringern Sie die Lautstärke.

void DY::DYPlayer::interludeSpecified(..)

Spielen Sie eine Zwischendatei nach Gerät und Nummer ab, die Nummer wird als 2 Byte gesendet. Hinweis aus dem Handbuch: „Musik-Zwischenspiel“ hat nur Stufe 1. Fortlaufendes Zwischenspiel überdeckt das vorherige Zwischenspiel (das Zwischenspiel wird sofort gespielt). Wenn das Zwischenspiel beendet ist, kehrt es zum ersten Unterbrechungspunkt des Zwischenspiels zurück und spielt weiter.

	Type	Name	Description
param	DY::device_t	device	device A DY::Device member e.g DY::Device::Flash or DY::Device::Sd
param	uint16_t	number	Nummer der Datei, e.g. 1 for 00001.mp3

void DY::DYPlayer::interludeSpecifiedDevicePath(..)

Spielen Sie ein Zwischenspiel nach Gerät und Pfad. Hinweis aus dem Handbuch: „Musik-Zwischenspiel“ hat nur Stufe 1. Fortlaufendes Zwischenspiel überdeckt das vorherige Zwischenspiel (das Zwischenspiel wird sofort gespielt). Wenn das Zwischenspiel beendet ist, kehrt es zum ersten Unterbrechungspunkt des Zwischenspiels zurück und spielt weiter.

Der Pfad kann aus bis zu 2 verschachtelten Verzeichnissen mit einer Länge von 8 Byte und einem Dateinamen mit einer Länge von 8 Byte ohne die Erweiterung mit einer Länge von 4 Byte bestehen. Wenn Ihre Verzeichnisnamen kürzer sind, können Sie mehr Verschachtelungen verwenden. Verwenden Sie für Ihre Pfade nicht mehr als 36 Byte. Wenn Sie mehr benötigen, sehen Sie in der Readme nach, Kapitel: Speichernutzung.

	Type	Name	Description
param	DY::device_t	device	device A DY::Device member e.g DY::Device::Flash or DY::Device::Sd
param	char	path	Pfadzeiger auf den Pfad der Datei (asbsolute)

void DY::DYPlayer::stopInterlude(..)

Stoppen Sie das Zwischenspiel und spielen Sie weiter. Stoppt auch die Wiedergabe des aktuellen Tons, wenn das Zwischenspiel nicht aktiv ist.

void DY::DYPlayer::setCycleMode(..)

Legt den Zyklusmodus fest. Sehen DY::play_mode_t für Modi und Bedeutung.

	Type	Name	Description
param	play_mode_t	mode	mode Der einzustellende Zyklusmodus

void DY::DYPlayer::setCycleTimes(..)

Stellen Sie ein, wie viele Zyklen in den Cycle-Modi 0, 1 oder 4 (Repeat-Modi) gespielt werden sollen.

	Type	Name	Description
param	uint16_t	cycles	Zyklen Die Zyklenanzahl für Wiederholungsmodi

void DY::DYPlayer::setEq(..)

Stellen Sie die Equalizer-Einstellung ein. Sehen DY::eq_t für Einstellungen.

	Type	Name	Description
param	DY::eq_t	eq	eq Die Equalizer-Einstellung

void DY::DYPlayer::select(..)

Wählen Sie eine Sounddatei aus, ohne sie abzuspielen.

	Type	Name	Description
param	`uint16_t`	`number`	Nummer der Datei, e.g. `1` for `00001.mp3`

`void` DY::DYPlayer::combinationPlay(..)

621 / 5.000

Übersetzungsergebnisse

Übersetzung

Mit der Kombinationswiedergabe können Sie eine Wiedergabeliste aus mehreren Sounddateien erstellen.
Sie könnten dies verwenden, um Zahlen zu kombinieren, z. B.: "fourthy-two", wo Sie Beispiele für "fourthy" und "two" haben.
Diese Funktion hat einen besonders merkwürdigen Parameter, Sie müssen die Sounddateien nach Namen angeben, sie müssen mit 2 Zahlen und einer Erweiterung benannt werden, z. B.: 01.mp3 und mit 01 angegeben. Sie sollten sie als Array-Zeiger übergeben. Sie müssen die Dateien in einem Verzeichnis ablegen, das DY, ZH oder XY heißen kann, Sie müssen das mit Ihrem Modul gelieferte Handbuch lesen oder alle ausprobieren. Vielleicht gibt es noch mehr Kombinationen! Siehe auch Loading sound files.

E.g.

```
const char * sounds[2][3] = { "01", "02" };
DY::DYPlayer::combinationPlay(sounds, 2);
```

	Type	Name	Description
param	`char`	`sounds`	sounds Ein Array von char[2], das die Namen der Sounds enthält, die der Reihe nach abgespielt werden sollen
param	`uint8_t`	`len`	len Die Länge des übergebenen Arrays

`void` DY::DYPlayer::endCombinationPlay(..)

Kombinationsspiel beenden

.typedef enum class DY::device_t

Vom Modul gemeldete Speichergeräte, aus denen Sie bei der Auswahl eines Speichergeräts auswählen können.

const	value	description
`DY::Device::Usb`	`0x00`	USB Storage device.
`DY::Device::Sd`	`0x01`	SD Card.
`DY::Device::Flash`	`0x02`	Onboard flash chip (usually winbond 32, 64Mbit flash).
`DY::Device::Fail`	`0xfe`	UART failure, can't be `-1` (so this can be uint8_t).
`DY::Device::NoDevice`	`0xff`	No storage device is online.

Diese Aufzählungsklasse basiert auf uint8_t.

typedef enum class DY::play_state_t

Der aktuelle Wiedergabestatus des Moduls.

const	value	description
`DY::PlayState::Fail`	`-1`	UART-Fehler, kann ein Verbindungs- oder CRC-Problem sein.
`DY::PlayState::Stopped`	`0`	
`DY::PlayState::Playing`	`1`	
`DY::PlayState::Paused`	`2`	

Diese Aufzählungsklasse basiert auf int8_t.

typedef enum class DY::eq_t

Equalize settings.

const	value
DY::Eq::Normal	0x00
DY::Eq::Pop	0x01
DY::Eq::Rock	0x02
DY::Eq::Jazz	0x03
DY::Eq::Classic	0x04

This enum class is based off uint8_t.

typedef enum class DY::play_mode_t

Wiedergabemodi sind im Grunde alles, was Sie normalerweise auf einem Mediaplayer finden, i.e.: Repeat 1, Repeat all, Repeat list (dir), playlist (by dir), random play.

Die Standardeinstellung ist vielleicht etwas unerwartet: `DY::PlayMode::OneOff`. Oft werden diese Module in Spielzeug oder Informationsdisplays verwendet, wo Sie einen Knopf drücken und einen entsprechenden Ton hören können. Um das Standardverhalten des Mediaplayers zu erhalten, sollten Sie wahrscheinlich festlegen `DY::PlayMode::Sequence` um einfach mit dem nächsten Lied fortzufahren, bis alle gespielt oder übersprungen sind, und dann aufhören.

const	value	description
DY::PlayMode::Repeat	0x00	Spielen Sie alle Musikstücke nacheinander ab und wiederholen Sie sie.
DY::PlayMode::RepeatOne	0x01	Repeat current sound.
DY::PlayMode::OneOff	0x02	Play sound file and stop.
DY::PlayMode::Random	0x03	Play random sound file.
DY::PlayMode::RepeatDir	0x04	Repeat current directory.
DY::PlayMode::RandomDir	0x05	Play random sound file in current folder.
DY::PlayMode::SequenceDir	0x06	Play all sound files in current folder in sequence, and stop.
DY::PlayMode::Sequence	0x07	Play all sound files on device in sequence, and stop.

Loading sound files

Normal Playback

Für die normale Wiedergabe von Sounddateien empfehle ich Ihnen, Ihre Dateien der Reihe nach zu benennen und sie im Stammverzeichnis des Laufwerks abzulegen, es sei denn, Sie brauchen etwas Ausgefalleneres. Die Nummerierung sollte wie folgt sein:

```
00001.mp3
00002.mp3
00003.mp3
...
65535.mp3
```

Wichtig: Dateien werden nicht in der benannten Reihenfolge abgespielt

Die `DY::DYPlayer::playSpecified` Die Funktion spielt nicht unbedingt die Sounddatei mit dem Namen 00001.MP3 ab, wenn sie aufgefordert wird, Sounddatei 1 abzuspielen. Das Modul sucht nach der ersten Sounddatei, die im Dateisystem gefunden wird. Es verwendet weder den Dateinamen noch sortiert es nach Dateinamen. Um sicherzustellen, dass die Sounddateien in der erwarteten Reihenfolge abgespielt werden:

1. Leeren Sie das Speichergerät.

2. Stellen Sie sicher, dass sich keine Papierkorbdateien auf dem Speichergerät befinden.

3. Kopieren Sie die Dateien in der Reihenfolge, in der sie abgespielt werden sollen (es hilft, sie nacheinander zu benennen, nach Namen zu sortieren und die Dateien zu kopieren, normalerweise werden sie der Reihe nach kopiert.

Bitte melden Sie keine Fehler, die angeben, dass ein Fehler in der ist `DY::DYPlayer::playSpecified` funktionieren, weil `00001.MP3` spielt nicht, wenn 1 an das Modul gesendet wird. Es handelt sich nicht um einen Fehler in dieser Bibliothek, sondern um die Funktionsweise des Moduls. Wenn Sie feststellen, dass die obigen Informationen nicht korrekt sind, oder wenn Sie Verbesserungsvorschläge haben, erstellen Sie bitte ein Problem.

Nach Dateipfad

Sie können auch Pfade und Dateinamen verwenden, aber weder Verzeichnis noch Dateiname sollten 8 Byte überschreiten. Sie können die Dateien in Verzeichnisse ablegen, aber machen Sie nicht viele verschachtelte Verzeichnisse. Dateien im Stammverzeichnis werden wiedergegeben, indem der Pfad als angegeben wird `/00001.MP3` (note the `/`).

Verwenden Sie normale Pfade

Im Handbuch des Moduls finden Sie, dass Pfade durch */ und * anstelle von ., Großbuchstaben usw. getrennt werden müssen. Vergessen Sie all das, geben Sie Pfade einfach so an, wie Sie es gewohnt sind, ohne Berücksichtigung der Groß- und Kleinschreibung , keine lustigen Zeichenersetzungen. Die Bibliothek übernimmt die erforderlichen Konvertierungen.

Kombinationsspiel

Für das Kombinationsspiel müssen Sie Dateien mit Namen aus 2 Zeichen plus der Erweiterung hinzufügen,e.g.: `01.mp3` in ein aufrufbares Verzeichnis `XY`, `ZH`, DY, müssen Sie das mit Ihrem Modul gelieferte Handbuch lesen oder alle ausprobieren. Vielleicht gibt es noch mehr Möglichkeiten! Die meisten Handbücher behaupten, dass es so sein sollte DYaber die meisten Leute haben das berichtetXY funktioniert eigentlich, so `try`:/DY/##.MP3` Erste.

Wenn Sie herausfinden, was die Buchstabenkombination auf Ihrem Board ist, fügen Sie bitte ein Problem mit dem Verzeichnisnamen und dem Modulmodellnamen hinzu, damit ich es hier hinzufügen kann, um anderen zu helfen.

ONE_LINE support?

Diese Bibliothek unterstützt derzeit nur UART. Es sollte relativ einfach sein, das ONE_Line-Protokoll hinzuzufügen, wie es im Handbuch zu finden ist. Beachten Sie jedoch, dass sich das gesamte Kommunikationsprotokoll vom UART-Protokoll unterscheidet. Es hat also keinen besonderen Vorteil, es dieser Bibliothek hinzuzufügen. Wenn Sie jedoch vorhaben, es zu implementieren, könnten wir der Einfachheit halber darüber sprechen, es mit dieser Bibliothek zusammenzuführen.

Runnning this library on PIC

Es wird nicht unterstützt, aber dort finden Sie einen Port here by @DoubleTop12, was beweist, dass es mit etwas Aufwand möglich ist.

Troubleshooting

Keine Kommunikation mit dem Vorstand

- Prüfen Sie, ob das Modul mit Strom versorgt wird.

• Überprüfen Sie, ob TX auf Ihrem Arduino-Board mit RX auf dem Modul verbunden ist und umgekehrt, überprüfen Sie, ob Sie die seriellen Leitungen über Widerstände verbunden haben, wenn Ihr Arduino mit 5 V betrieben wird. Sehen Wiring the module.

• Überprüfen Sie die Konfiguration der Platine, indem Sie entweder Widerstände mit dem richtigen Wert an die richtigen CON#-Pins anschließen oder, falls Ihre Platine solche hat, die DIP-Schalter auf den richtigen Wert einstellen. Sehen Wiring the module.

• Wenn Sie den seriellen Anschluss zwischen dem Computer zum Flashen und dem Anschluss an das Modul teilen, trennen Sie das Kabel (FTDI- oder USB-Kabel) von der Platine, bevor Sie sie booten. Alles auf der TX-Leitung des Moduls außer Ihrem Arduino-Board verhindert den Empfang von Daten.

• Überprüfen Sie, ob auf dem Speichergerät des Moduls kompatible Sounddateien vorhanden sind, möglicherweise verbindet es sich, hat aber keinen Sound zum Abspielen, verwenden Sie es `DY::DYPlayer::playSpecified` Sounddatei abzuspielen `00001.MP3` weil es fehlertoleranter ist als durch Pfadfunktionen.

Kein Ton

• Überprüfen Sie die Lautsprecherverbindung.

• Überprüfen Sie den Lautstärkeregler, einige Platinen haben ein Potentiometer, das Sie drehen können.

• Testen Sie den Lautsprecher auf einem anderen Gerät und prüfen Sie, ob er betriebsbereit ist.

• Verwenden Sie die `DY::DYPlayer::getPlayingDevice` Funktion, um zu sehen, ob das Gerät reagiert. Siehe die loading sound files Kapitel und wenden Sie die einfachste Dateistruktur an, um zu sehen, ob die Verkabelung usw. korrekt ist, bevor Sie Probleme einreichen.

• Überprüfen Sie, ob auf dem Speichergerät des Moduls kompatible Sounddateien vorhanden sind. Möglicherweise hat es keinen Sound zum Abspielen, verwenden Sie ihn `DY::DYPlayer::playSpecified` um die Sounddatei 00001.MP3 abzuspielen, weil sie fehlertoleranter ist als durch Pfadfunktionen.

Unerwartetes Verhalten

• Ich habe das Modul aufgefordert, 1 abzuspielen, aber 00001.MP3 wird nicht abgespielt, ich höre einen anderen Ton. Dies ist kein Fehler in dieser Bibliothek, überprüfen Sie: Important: files do not play in order of named sequence