

Sound Servoplatine

Über die Servo-Platine 510 können drei Soundmodule angesteuert werden. Es können JQ6500 Module oder MP3-TF16-p/DFPlayer Mini Module verwendet werden. Dabei spielt es keine Rolle ob nur JQ6500 oder nur MP3-TF16-p/DFPlayer Mini oder diese beliebig gemischt eingesetzt werden. Die drei Module können auch parallel je einen Sound abspielen.

ATTiny 85

Zunächst muss der ATTiny für die Servoplatine programmiert werden.

Die Anleitung dazu [Programmierung von Attinys für Servo oder Charlieplexing](#) ist im WIKI zu finden.

Im Pattern-Configurator unter spezielle Module **Servo-MP3** auswählen.

Pattern Configurator

Beispiele | **Spezielle Module** | Extras

Diese Seite enthält Funktionen mit denen spezielle, auf dem ATTiny basierte Module programmiert und getestet werden können. [by Hardi](#)

55) Programmieradapter

Ein ATTiny hat keinen USB Anschluss. Darum benötigt man zur Programmierung einen Programmieradapter (In Circuit Programmer). Das kann ein Arduino mit besonderen Programm sein. Mit dem Knopf Links wird das Programm zu Arduino übertragen.

Dieses Programm kann auf dem Tiny_UniProg und auf einen "Nackten" Arduino in einem Steckbrett eingesetzt werden.
Der "HV Reset" ist allerdings nur mit der "Tiny_UniProg" Platine möglich. Er wird für das Programmieren des Servo Programms benötigt.

est Patt

ms,0.5

rnT22(

T22(LE

nur die

ms 250

ms,;

ms,1

20 m

en Zei

50 ms

Charlieplexing | **Servo** | **Servo-MP3**

Prog.Servo/MP3

Man kann das Servo-Modul auf zur Ansteuerung von Soundmodulen nutzen (JQ6500/MP3-TF-16p) nutzen.

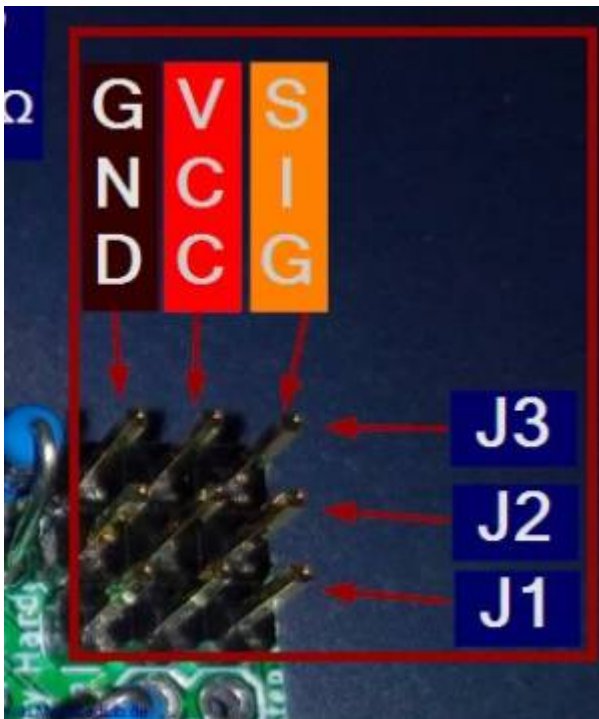
Dabei werden die Soundmodule mit je einem 1k Widerstand an die Signal-Pins der Servoausgänge angeschlossen. Zusätzlich ist die Masse zu verbinden. Die Stromversorgung kann man getrennt lösen - meist funktioniert es aber auch die 5V von den Servoanschlüssen zu nutzen.

Zur Programmierung des ATTiny85 wird ein Programmieradapter benötigt (siehe oben) in der der ATTiny eingesteckt wird.

Achtung: Die Software für das Servo Modul ist noch in der Entwicklung.

Dialog schließen

Das folgende Bild zeigt die Pins am Ausgang der Servo-Platine. Der SIG-J1 - Pin ist mit dem Eingang des ersten Soundmodules zu verbinden, entsprechend J2 mit Modul 2 und J3 mit Modul3.



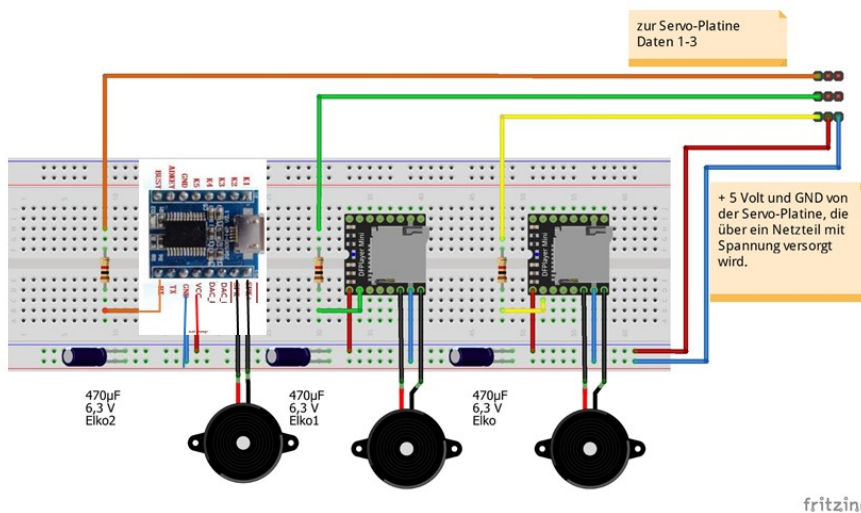
JQ6500 und MP3-TF16-p/DFPlayer Mini



1	+ 5 Volt	BUSY	16
2	Eingang	USB-	15
3	TX	USB+	14
4	DAC_R	ADKEY2	13
5	DAC_L	ADKEY1	12
6	SPK	I/O2	11
7	GND	GND	10
8	SPK	I/O1	9

Schaltung

Die drei Ausgänge der Servo-Platine werden nun über je einen 1 kOhm Widerstand mit den Eingängen der Sound-Module verbunden. Die Versorgungsspannung sollte über eine Verteilerplatine mit angeschlossener stabiler Spannungsversorgung erfolgen, da die Leistung an der Hauptplatine zur Versorgung der Sound-Module nicht ausreicht. Die 470uF Elektrolytkondensatoren dienen als Puffer für die recht hohen Einschaltströme der Soundmodule. Eine separate 5V Spannungsversorgung ist ebenfalls möglich. Dann müssen GND und -5 Volt verbunden werden.



Programm-Generator

Im Prog-Gen gibt es für den Servo-Sound folgende Befehle:

ATTiny85	Soundmodule über ATTiny85
Befehl an Soundmodul	Befehl an Soundmodul über Servoplatine
Titel # abspielen	Track auf Soundmodul über Servoplatine abspielen
Set	MP3-TF-16p einstellen
Titel aus Ordner abspielen	MP3-TF-16p, Track aus Ordner abspielen
Pin MP3-Modul definieren	Anschluss für MP3-Modul auswählen
Soundmodul definieren	Typ für angeschlossenes MP3-Modul einstellen
Titel # aus Hauptverzeichnis abspielen	Track # vom angegebenen Modul abspielen (Rootverzeichnis)
Titel # aus mp3 abspielen	Track # aus mp3 auf Modul abspielen

Beispiel der Soundsteuerung mit den Befehlen:

99														
100	✓	80	Rot											
101		81	Rot											
102	✓	82	Rot											
103														
104	✓	90	Rot											
105	✓	91	Rot											
106	✓	92	Rot											
107														
108	✓	93	Rot											
109	✓	94	Rot											
110	✓	95	Rot											
111	✓	96	Rot											
112														
113														
114														
115														

In den Zeilen 100-102 wird der an die Sound-Platine angeschlossene Modultyp festgelegt. Diese Befehle müssen nur einmalig gesendet werden. Der ATTiny speichert die Einstellung. Eine Änderung ist nur dann notwendig, wenn Änderungen bei den angeschlossenen Modul-Typen vorgenommen werden. Im Beispiel

- Ch1 → JQ6500
- Ch2 → MP3-TF16-p
- Ch3 → MP3-TF16-p

Zeile 104 - 106: legt den Ausgang fest auf den der nächste Befehl gehen soll. Hier:

- Zeile 104 - Ausgang 1, JQ6500.
- Zeile 105 - Ausgang 2, MP3-TF16-p
- Zeile 106 - Ausgang 3, MP3-TF16-p

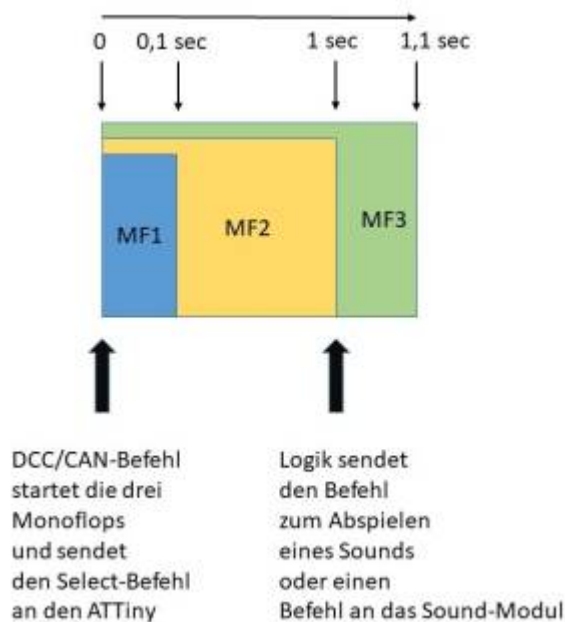
Beispiel:

- DCC Befehl 90 und anschließend 95 spielt Track 1 vom JQ6500 ab.
- DCC Befehl 91 und anschließend 95 spielt Track 1 vom ersten MP3-TF16-p ab.
- DCC Befehl 92 und anschließend 95 spielt Track 1 vom zweiten MP3-TF16-p ab.
- DCC Befehl 91 und anschließend 93 verringert die Laustärke beim zweiten MP3-TF16-p.

Anmerkungen:

- In der Macroauswahl wird der Hinweis „Sendet einen Befehl an ein MP3-TF-16p-Soundmodul welches über SERVO3 an einer Servoplatine angeschlossen ist“ gegeben. Das trifft so nicht zu, da die Befehle an alle drei Ausgänge gesendet werden können und auch für beide Modultypen gelten. Einschränkungen gibt es natürlich beim JQ 6500, da nur fünf Tracks gespeichert werden können.
- Alle Befehle an diese ATTiny Soundplatine müssen über eine LED Adresse laufen, im Beispiel LED 1. Sollte es (ungewollt) eine Verschiebung geben mit dem Befehl „next LED -1“ wieder auf die LED Adresse der Sound-Platine zurück gehen.
- Über den **Kleinen Verteiler** mit der Copy-Funktion, Einstellung über den Jumper, kann man parallel Test-LEDs anschließen und optisch die Funktion überprüfen.

Steuerung über DCC/CAN-Befehle



Vor jedem Sound-Befehl muss das Modul ausgewählt werden auf dem sich die Sound-Datei befindet. So können Sound-Dateien in unterschiedlicher Reihenfolge von den drei Modulen abgespielt werden. Die Logik stellt sicher, dass zunächst über den ATTiny das Modul ausgewählt wird, der Befehl umgesetzt werden kann und dann, mit zeitlichem Verzug, der Track ausgewählt oder eine andere Funktion des Moduls aufgerufen wird.

Beispiel:

Im folgenden Beispiel wird bei Aufruf des:

- DCC-Befehl „3“ der erste Sound des ersten Moduls (JP6500) abgerufen
- DCC-Befehl „4“ der erste Sound des zweiten Moduls (DFPlayer Mini) abgerufen
- DCC-Befehl „5“ der erste Sound des dritten Moduls (DFPlayer Mini) abgerufen

Aktiv	Filter	Adresse oder Name	Typ	Startwert	Beschreibung	Verteiler-Nummer	Stecker-Nummer	Leuch	Name	Beleuchtung, Sound, oder andere Effekte	Start LedNr	LEDs	InCh	Loc InCh	LED/ Sound Kanal
105					Modul 1 JQ6500										
108	✓	3	Rot		startet MonoFloP (MF) 1				! Mono-Flop	MonoFloP(MF11, #InCh, 0.1 Sek)				1	0
109	✓	MF11	Rot		sendet Select Befehl für Modul 1				Pin MP3-Modul definieren	MP3_SELECT_MODULE(LEd, #InCh, 1)	1	^ C1-1		1	0
110	✓	3	Rot		startet MonoFloP (MF) 2				! Mono-Flop	MonoFloP(MF12, #InCh, 1 Sek)				1	0
111	✓	3	Rot		startet MonoFloP (MF) 3				! Mono-Flop	MonoFloP(MF13, #InCh, 1.1 Sek)				1	0
112	✓	MF12	Rot		logische Verknüpfung der drei MFs				! Logische Verknüpfung	Logic(MF12u13, NOT #InCh AND MF13)				1	0
113	✓	MF12u13	Rot		spielt Track 1 vom JP6500 ab (Welding)				! Titel # abspielen	MP3_TRACK(LEd, #InCh, MP3_PLAY_TRACK, 1)	1	^ C1-1		1	0
114															
115					Modul 2 DFPlayer Mini										
116	✓	4	Rot		startet MF1				! Mono-Flop	MonoFloP(MF21, #InCh, 0.1 Sek)				1	0
117	✓	MF21	Rot		sendet Select Befehl für Modul 2				Pin MP3-Modul definieren	MP3_SELECT_MODULE(LEd, #InCh, 2)	1	^ C1-1		1	0
118	✓	4	Rot		startet MF2				! Mono-Flop	MonoFloP(MF22, #InCh, 1 Sek)				1	0
119	✓	4	Rot		startet MF3				! Mono-Flop	MonoFloP(MF23, #InCh, 1.1 Sek)				1	0
120	✓	MF22	Rot		logische Verknüpfung der drei MFs				! Logische Verknüpfung	Logic(MF22u23, NOT #InCh AND MF23)				1	0
121	✓	MF22u23	Rot		spielt Track 1 aus Root von DFPlayer 1 ab				! Titel # abspielen	MP3_TRACK(LEd, #InCh, MP3_PLAY_TRACK, 1)	1	^ C1-1		1	0
122															
123					Modul3 DFPlayer Mini										
124	✓	5	Rot		startet MF1				! Mono-Flop	MonoFloP(MF31, #InCh, 0.1 Sek)				1	0
125	✓	MF31	Rot		sendet Select Befehl für Modul 3				Pin MP3-Modul definieren	MP3_SELECT_MODULE(LEd, #InCh, 3)	1	^ C1-1		1	0
126	✓	5	Rot		startet MF2				! Mono-Flop	MonoFloP(MF32, #InCh, 1 Sek)				1	0
127	✓	5	Rot		startet MF3				! Mono-Flop	MonoFloP(MF33, #InCh, 1.1 Sek)				1	0
128	✓	MF32	Rot		logische Verknüpfung der drei MFs				! Logische Verknüpfung	Logic(MF32u33, NOT #InCh AND MF33)				1	0
129	✓	MF32u33	Rot		spielt Track 1 aus Root von DFPlayer 2 ab				! Titel # abspielen	MP3_TRACK(LEd, #InCh, MP3_PLAY_TRACK, 1)	1	^ C1-1		1	0
130															
131					Welding Modul 1 JQ6500										
132	✓	MF12u13	Rot		Imported_Pattern (pc)				Muster Pattern_Configurator	// Activation: N_ButtonsInCh_to_TapVar1(#InCh, 1)Pattern26	2	1	1	0	0
133															
134															

Über parallel angeschlossene Test-LEDs, Stichwort Mini-Verteiler, kann der Ablauf optisch sehr gut verfolgt werden. Andere/kürzere Zeitintervalle für die MonoFlops sind möglich und ggf. durch Tests zu ermitteln.

In Zeile 133 wird über die Variable MF12u13 zeitgleich mit dem dazu gehörigen Geräusch vom Sound-Modul 2 ein Schweißlicht ausgelöst. Geräuschlänge und Länge des Lichts können leicht durch Anpassung des **Schweisslicht** über den Pattern-Configurator angepasst werden.

Damit man den Schweißler nicht immer persönlich wecken muss, hier eine Lösung mit der Zufallsschaltung (Random-Funktion).

Aktiv	Filter	Adresse oder Name	Typ	Startwert	Beschreibung	Verteiler-Nummer	Stecker-Nummer	Leuch	Name	Beleuchtung, Sound, oder andere Effekte	Start LedNr	LEDs	InCh	Loc InCh	LED/ Sound Kanal
106					Modul 1 JQ6500										
109	✓	3	AnAus	0					! Zufallsschaltung 1 Ausgang	Random(WL1, #InCh, RN_NORMAL, 3 sec, 15 sec, 1 sec, 1 sec)				1	0
111	✓	WL1	Rot		startet MonoFloP (MF) 1				! Mono-Flop	MonoFloP(MF11, #InCh, 0.1 Sek)				1	0
112	✓	MF11	Rot		sendet Select Befehl für Modul 1				Pin MP3-Modul definieren	MP3_SELECT_MODULE(LEd, #InCh, 1)	1	^ C1-1		1	0
113	✓	WL1	Rot		startet MonoFloP (MF) 2				! Mono-Flop	MonoFloP(MF12, #InCh, 1 Sek)				1	0
114	✓	WL1	Rot		startet MonoFloP (MF) 3				! Mono-Flop	MonoFloP(MF13, #InCh, 1.1 Sek)				1	0
115	✓	MF12	Rot		logische Verknüpfung der drei MFs				! Logische Verknüpfung	Logic(MF12u13, NOT #InCh AND MF13)				1	0
116	✓	MF12u13	Rot		spielt Track 1 vom JP6500 ab (Welding)				! Titel # abspielen	MP3_TRACK(LEd, #InCh, MP3_PLAY_TRACK, 1)	1	^ C1-1		1	0
117	✓	MF12u13	Rot		Imported_Pattern (pc)				Muster Pattern_Configurator	// Activation: N_ButtonsInCh_to_TapVar1(#InCh, 1)Pattern26	2	1	1	0	0
118															
119	✓								! LED Nummer manipulieren	// Next_LED(-2)	3	-2	0	0	0
120															
121					Modul 2 DFPlayer Mini										
122	✓	4	Rot		startet MF1				! Mono-Flop	MonoFloP(MF21, #InCh, 0.1 Sek)				1	0
123	✓	MF21	Rot		sendet Select Befehl für Modul 2				Pin MP3-Modul definieren	MP3_SELECT_MODULE(LEd, #InCh, 2)	1	^ C1-1		1	0
124	✓	4	Rot		startet MF2				! Mono-Flop	MonoFloP(MF22, #InCh, 1 Sek)				1	0
125	✓	4	Rot		startet MF3				! Mono-Flop	MonoFloP(MF23, #InCh, 1.1 Sek)				1	0
126	✓	MF22	Rot		logische Verknüpfung der drei MFs				! Logische Verknüpfung	Logic(MF22u23, NOT #InCh AND MF23)				1	0
127	✓	MF22u23	Rot		spielt Track 1 aus Root von DFPlayer 1 ab				! Titel # abspielen	MP3_TRACK(LEd, #InCh, MP3_PLAY_TRACK, 1)	1	^ C1-1		1	0
128															
129					Modul3 DFPlayer Mini										
130	✓	5	Rot		startet MF1				! Mono-Flop	MonoFloP(MF31, #InCh, 0.1 Sek)				1	0
131	✓	MF31	Rot		sendet Select Befehl für Modul 3				Pin MP3-Modul definieren	MP3_SELECT_MODULE(LEd, #InCh, 3)	1	^ C1-1		1	0
132	✓	5	Rot		startet MF2				! Mono-Flop	MonoFloP(MF32, #InCh, 1 Sek)				1	0
133	✓	5	Rot		startet MF3				! Mono-Flop	MonoFloP(MF33, #InCh, 1.1 Sek)				1	0
134	✓	MF32	Rot		logische Verknüpfung der drei MFs				! Logische Verknüpfung	Logic(MF32u33, NOT #InCh AND MF33)				1	0
135	✓	MF32u33	Rot		spielt Track 1 aus Root von DFPlayer 2 ab				! Titel # abspielen	MP3_TRACK(LEd, #InCh, MP3_PLAY_TRACK, 1)	1	^ C1-1		1	0
136															
137															

From:
<https://wiki.mobaledlib.de/> - MobaLedLib Wiki

Permanent link:
https://wiki.mobaledlib.de/anleitungen/spezial/sound_servoplatine?rev=1639692082

Last update: 2021/12/16 22:01

