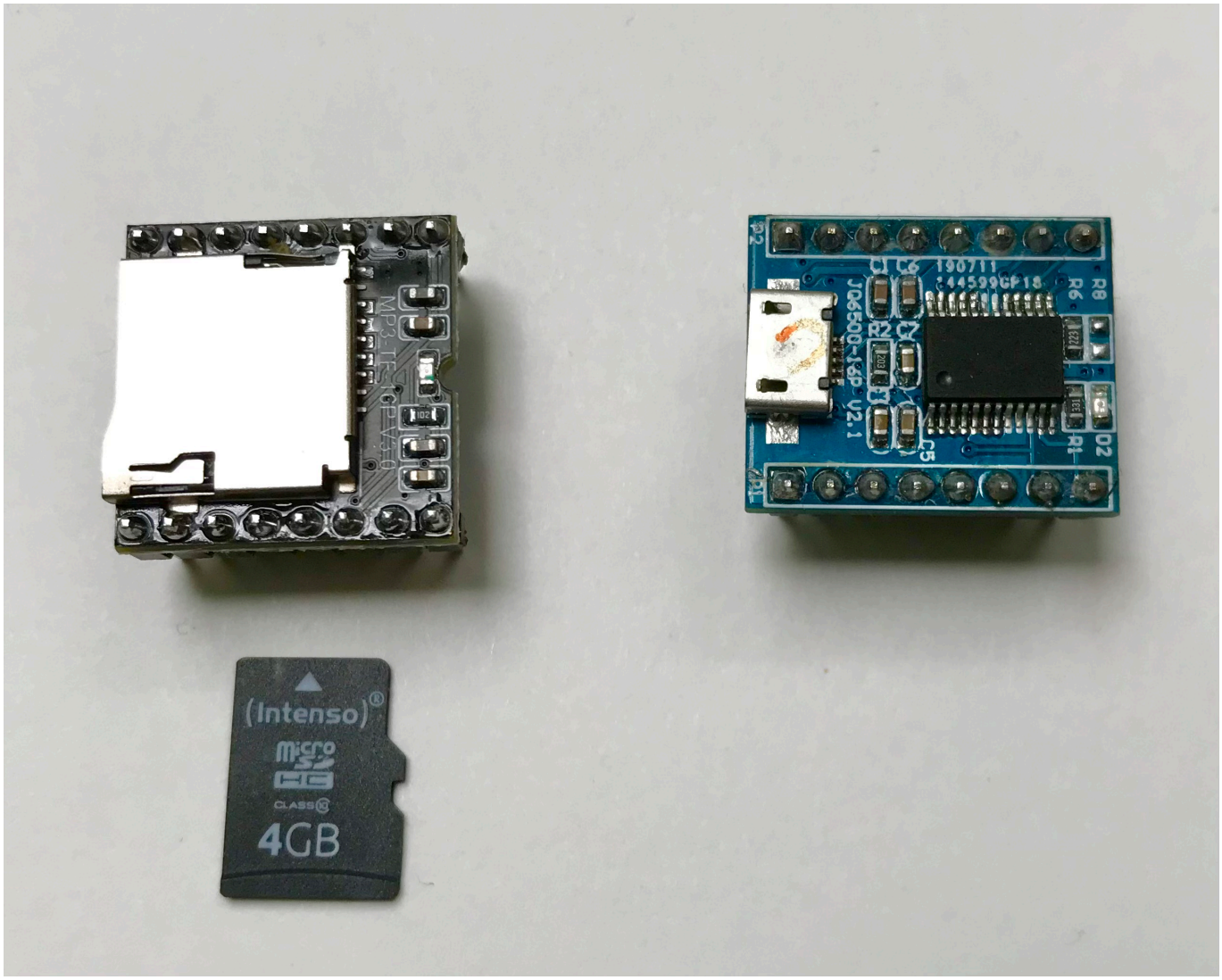
Soundmodule

Je nach Softwareversion werden nur das **JQ6500**-Soundmodul (LocoTurn **V1.0**) oder das **JQ6500**-Soundmodul und der **DFPlayer** unterstützt (ab LocoTurn **V1.1**).

Die Lautstärke kann über das Menü oder über DCC-Befehle eingestellt werden. Der Wert der Laustärke (0 - 30) wird im EEPROM des Nano gespeichert und beim nächsten Einschalten von LocoTurn wieder geladen. Weiterhin kann der Sound generell über das Menü aktiviert/deaktiviert werden.

Hier die beiden Module, links der DF-Player und rechts das JQ6500:



Wo bekomme ich geeignete Sounds her?

Es ist zwar möglich, Soundausschnitte aus YouTube-Videos und anderweitigen Quellen privat zu nutzen; es wäre dann aber rechtlich schwierig, diese zu veröffentlichen. Wer die Möglichkeit und Beziehungen hat, den Ton einer Drehscheibe aufzuzeichnen, der kann sich gerne einbringen.

Hier gibt es gute MP3-Dateien von Drehscheiben-Sounds, die als Ausgangspunkt eine gute Basis

bilden und ggf. bearbeitet/getuned werden müssen:

https://modellbahn.holgerlauer.de/

Link digitalzentrale.de

Wie kann ich Sounds bearbeiten?

Für das Bearbeiten von mp3, wav oder ähnlichen Formaten eignet sich das kostenlose Programm Audacity https://www.audacity.de/. Hierfür gibt es im Netz gute Beschreibungen und Tutorials.

Welche Lautsprecher sind geeignet?

Als Lautsprecher muss ein 8 Ohm, 3 Watt Modell verwendet werden.

Je nach verfügbarem Platz und Einbauort gibt es die unterschiedlichsten Lautsprecher-Größen, von winzig wie einem ESU-Brüllwürfel für den Einbau in Loks bis hin zu größeren Modellen mit mehreren cm Durchmesser.

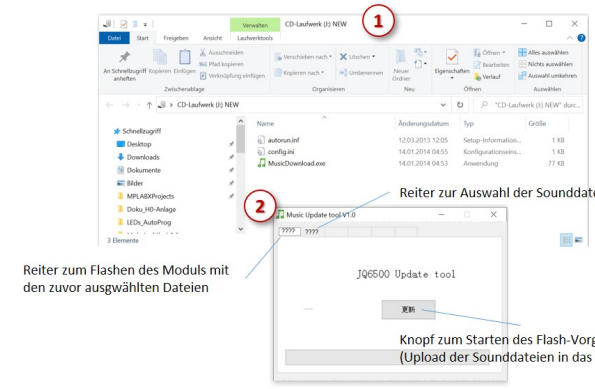
Sketch-Version V1.0

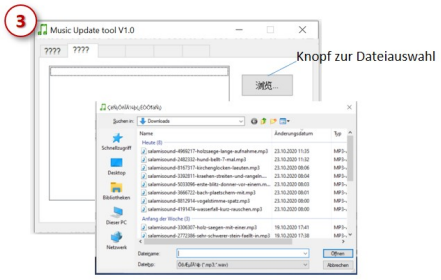
Die Sketch-Version V1.0 unterstützt nur das **JQ6500**-Soundmodul.

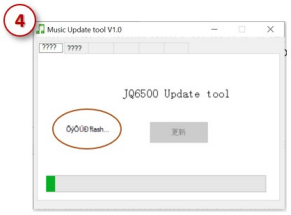
Initialisierung des JQ6500 und Hochladen von Soundfiles

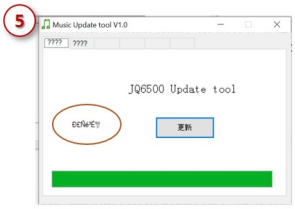
MobaLedLib - Sound-Dateien auf JQ6500 speichern

- Mit etwas Pech muss man das JQ6500 zuerst einmal initialisieren, da es sonst nicht vom PC erkannt wird. Siehe auch MobaLedLib-Thread und diverse Internet-Seiten (Video des Italieners und VMWare).
- JQ6500 via USB an PC anschließen. Wenn man Glück hat, wird das Modul als neues CD-Laufwerk erkannt und kann mit dem Explorer angezeigt werden ①.
- Nach Start der **MusicDownload.exe** erscheint ein spartanisches chinesisches Menü ②.
- Nach Drücken der rechten „????“ können die Sound-Files vom PC ausgewählt werden ③.
- Das Modul hat normalerweise 2 MB Speicher. Überschreitet man diesen mit den ausgewählten Dateien, kann der Upload nicht durchgeführt werden. Es erscheint eine chinesische Fehlermeldung.
- Zurück auf dem linken Reiter „????“ den Knopf in der Mitte drücken, um den Upload-Vorgang zu starten. Es erscheint unten ein grüner Balken. In der Mitte links steht sogar das Wort „flash“ ④.
- Wenn der Upload fertig ist, ändert sich der Text und der grüne Balken ist komplett gefüllt ⑤.
- Dann sollten alle Files auf dem JQ6500 gespeichert sein. Dann die App schließen und das USB-Kabel entfernen.
- Welche Dateien aufgespielt wurden, lässt sich hinterher nicht mehr feststellen. Daher am besten die Dinger durchnummerieren (z.B. mit permanent Stift auf dem USB-Anschluss) und die Namen der Sounds irgendwo dokumentieren.









Wenn ein flashdata.ini Fehler erscheint, muss in den Eigenschaften der .exe.Datei der Kompatibilitätsmodus auf Windows 7 gestellt werden !

Soundfiles auf dem JQ6500

Für Drehscheibenbewegungen sollten 3 Soundfiles auf dem JQ6500 abgespeichert werden, aktuell werden davon nur 1 und 3 genutzt:

- Sound 1 (= file 1) –> gesampeltes File bestehend aus Hupen, Anfahren und Fahren. Je nach Drehgeschwindigkeit und gewünschter Soundlänge (sollte schon ein paar Minuten dauern) muss der Fahrsound mehrfach hintereinander ins File kopiert werden (z.B. mit dem Programm *Audacity*)
- Sound 2 (= file 2) –> Abbremsen und Stopp (aktuell nicht im Einsatz, da Pause zwischen den sounds zu lange!)
- Sound 3 (= file 3) –> Hupen

Folgende #defines sind wichtig:

```
#define SOUND1_FILENR  1  // sound/file number of JQ6500 for turntable start and running (Hupe, Anfahren und Drehen)
#define SOUND2_FILENR  3  // sound for turntable stop (aktuell nur die Hupe)
```

Es können bei Bedarf noch weitere Sounds auf das JQ6500 gespielt werden, diese lassen sich dann über DCC-Befehle abspielen. Die Auswahl erfolgt über die #defines:

```
#define DCC_SOUNDFILE_1    1   // File-Nr. auf dem Soundmodul, das abgespielt wird mit dem entsprechenden DCC-Befehl
#define DCC_SOUNDFILE_2    2   // J6500; Dateien stehen im Rootverzeichnis, Reihenfolge geht nach Reihenfolge des Kopierens auf das Modul
#define DCC_SOUNDFILE_3    3   // DFPlayer: Dateien müssen im \mp3-Folder stehen, Nomenklatur 0001_beliebiger Text (4-stellige Nummer, führende Nullen + sprechender Text)
#define DCC_SOUNDFILE_4    4
#define DCC_SOUNDFILE_5    5
```

```
#define DCC_SOUNDFILE_6    6
```

Sketch-Version ab V1.1

Die Versionen V1.1 und höher verwenden entweder das **JQ6500**-Soundmodul oder den **DFPlayer**.

In der Turntable_config.h muss das gewünschte Soundmodul beim **#define USE_SOUNDMODULE** hinterlegt werden:

```
#define DFPLAYER          1          // mit DS-Kartenslot
#define JQ6500            2          // 2MB Speicher onboard
#define USE_SOUNDMODULE   DFPLAYER   //JQ6500 Auswahl des verwendeten
Soundmoduls
```

Auf das JQ6500-Modul passen 2 MByte mp3-Files, das reicht bei guter Sampling-Rate für mehrere Minuten Sounds.

Der DFPlayer verfügt über einen Micro-SD-Karten-Slot und unterstützt Karten bis 32 GByte (kleinere Karten bekommt man fast nicht mehr 😬). Hier kann man hohe Sampling-Raten verwenden und etwas verschwenderischer mit dem Speicherplatz umgehen.

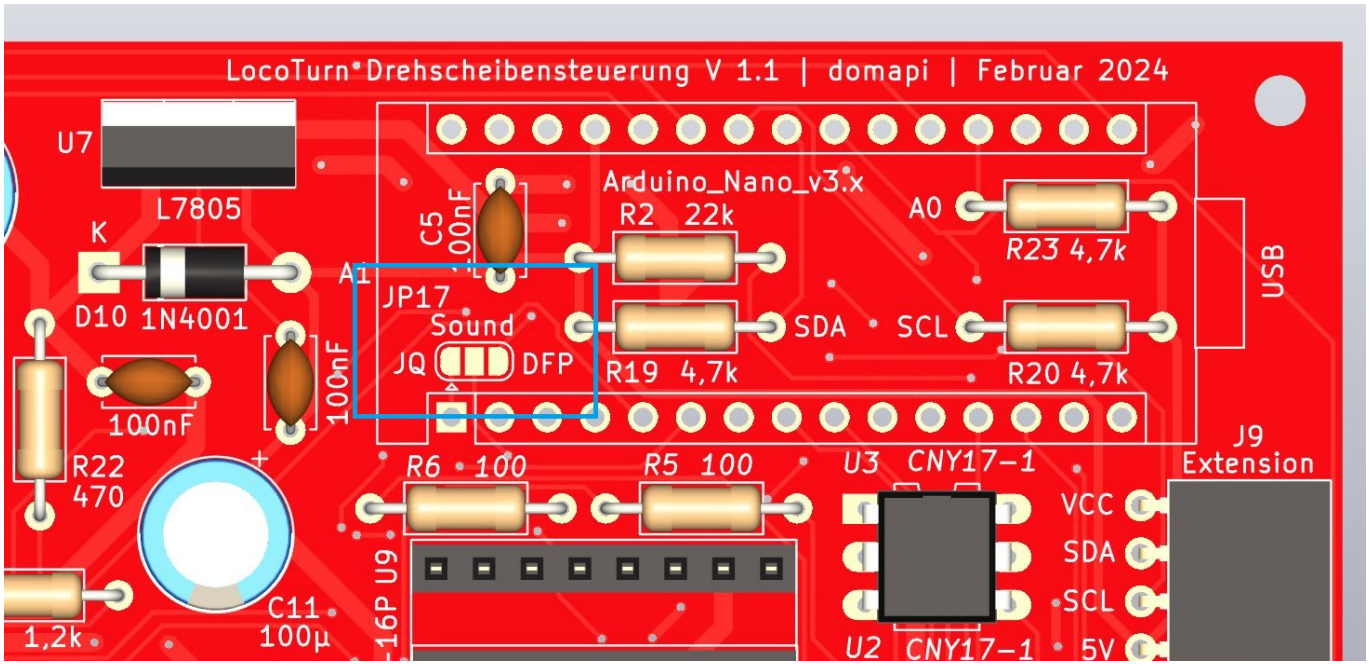
Allerdings sind die DFPlayer etwas heikel: Es gibt verschiedene Versionen mit unterschiedlichem Chipsatz, die auch noch verschiedene Firmware-Versionen aufgespielt haben. Ich habe nicht alle Kombinationen getestet, daher kann es sein, dass das euer eingebautes Modul nicht mit dem Sketch funktioniert! Da hilft nur ausprobieren; bei ein paar € pro DFPlayer hält sich der finanzielle Verlust in Grenzen.

Auswahl des Soundmoduls auf der Platine

Je nach Platinenversion muss das verwendete Soundmodul entweder über einen Lötjumper oder einen Workaround ausgewählt werden.

Lötjumper (Platine V1.1)

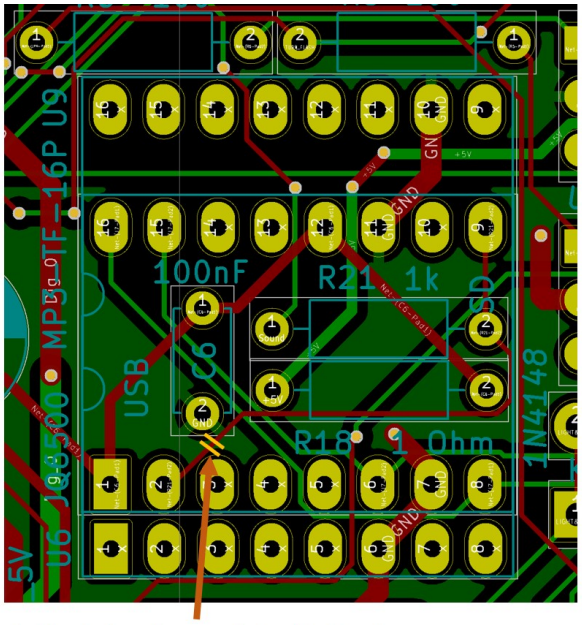
Ab Platinenversion V1.1 gibt es einen 3-fach Jumper JP17 „Sound“. Dieser liegt unter dem Nano-Steckplatz. Der Jumper muss entsprechend des verwendeten Soundmoduls mit Lot überbrückt werden. Entweder die linken beiden Lötpunkte (JQ6500) oder die rechten beiden (DFPlayer).



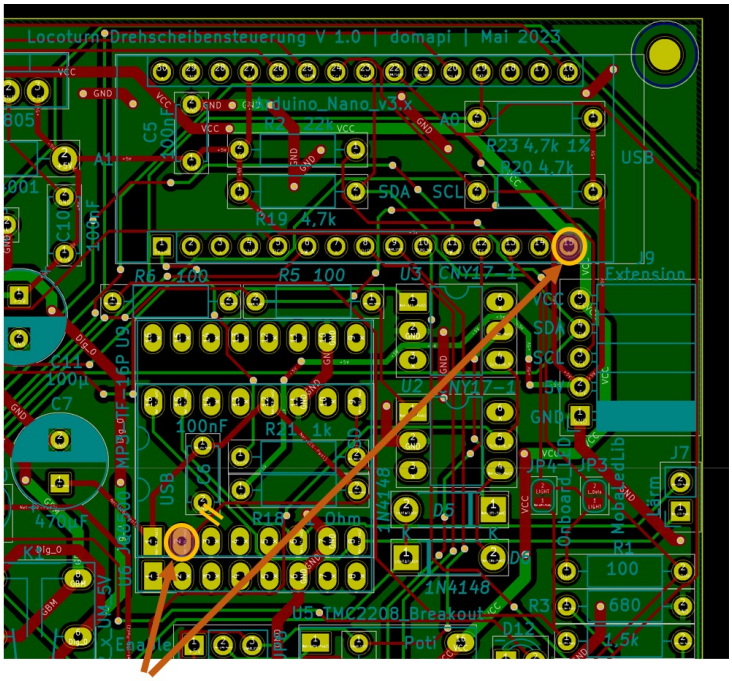
Notwendiger Workaround für Verwendung des DFPlayers (Platine V1.0)

Die Platine V1.0 unterstützt standardmäßig nur das JQ6500-Modul. Daher muss mann bei der Platine V1.0 den DFPlayer mit einem Workaround anschließen, da er nicht wie der JQ6500 an der hardware serial, sondern an einer software serial Schnittstelle betrieben wird:

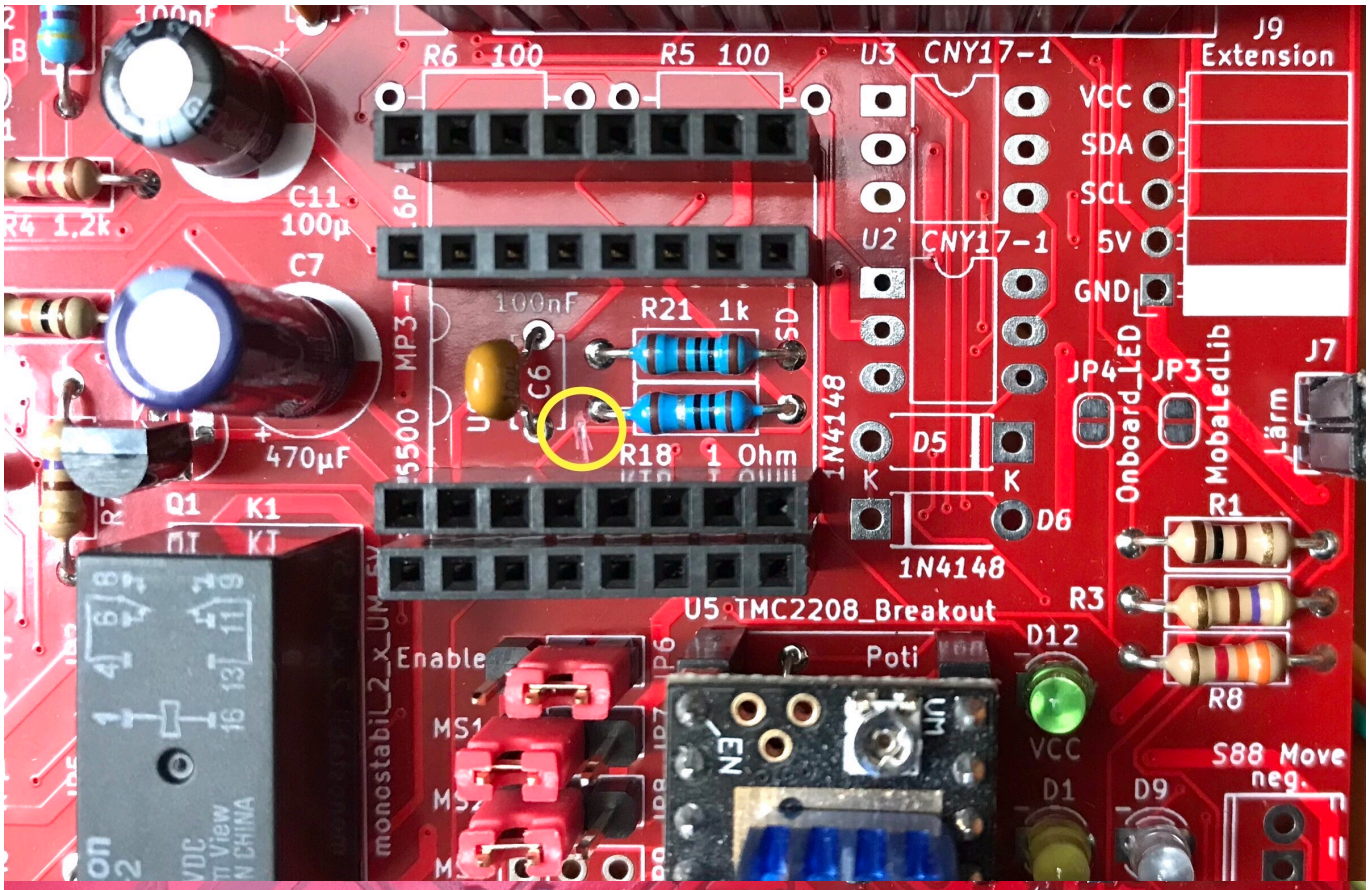
1. Hierfür muss entweder der 1kOhm Widerstand (R21 - dieser würde für das JQ6500 benötigt) von den RX-Pins der beiden Soundmodule entfernt werden (bzw. man baut ihn von Anfang an nicht ein) oder man unterbricht eine Leiterbahn (siehe Abbildung).
2. Weiterhin ist ein separater 1k-Widerstand zur Verbindung des SoftwareSerial TX Pins (D12) des Nanos und des RX-Pins von DFPlayer erforderlich. Diesen lötet man am besten auf der Rückseite auf (Vorsicht, dass dabei kein Kurzschluss entsteht!)

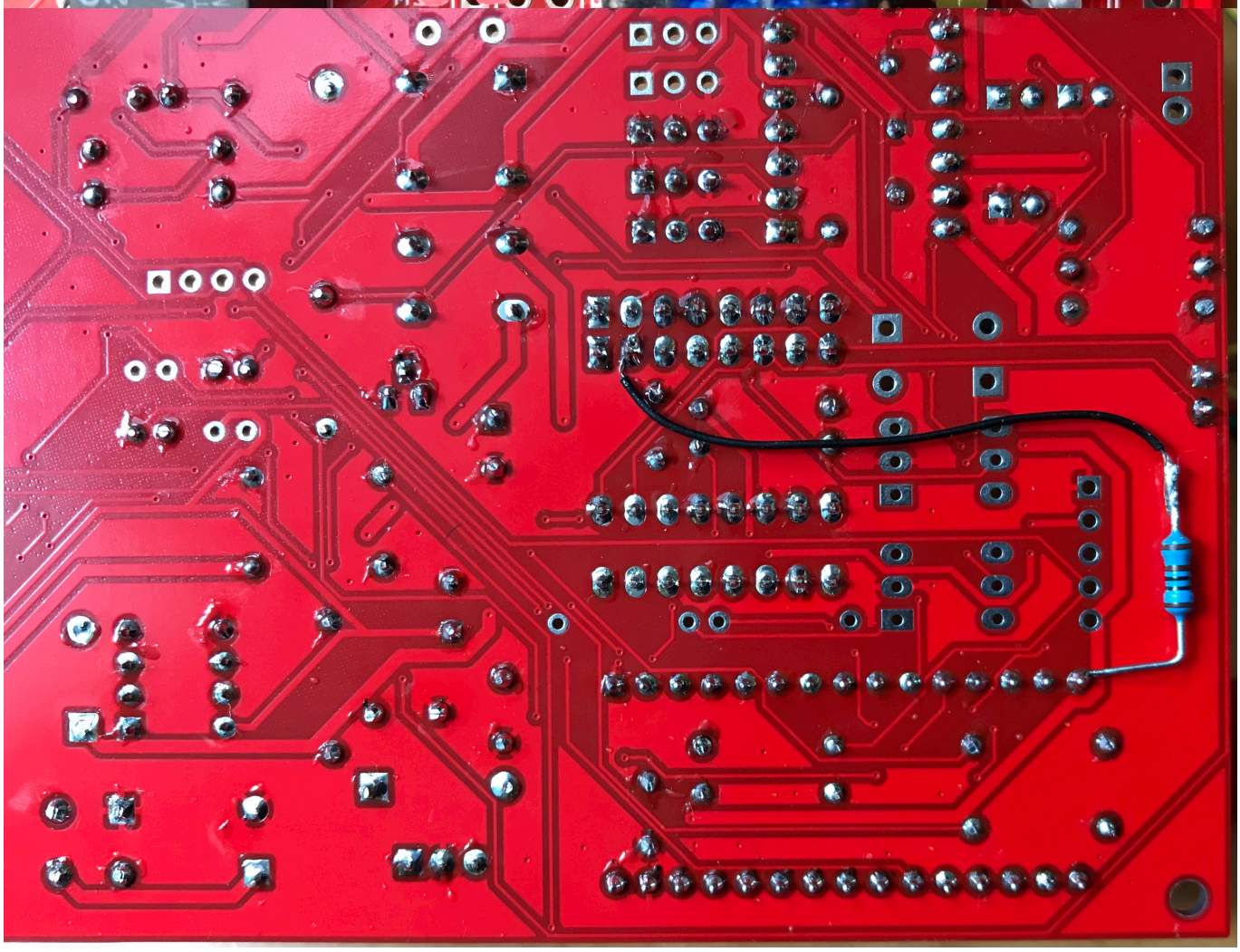


Leiterbahn oben auf der Platine trennen (mit Schraubenzieher, Messer oder Skalpell), alternativ R21 (1k) auslöten



1k Widerstand zwischen Nano-Pin D12 und DFPlayer Pin 2 einlöten


100nF
R6 100
R5 100
U3 CNY17-1
J9
Extension
VCC
SDA
SCL
5V
GND
C11
100µ
R4 1,2k
C7
470µF
100nF
R21 1k
U2 CNY17-1
JP4
JP3
J7
Lärm
Onboard_LED
MobaLedLib
R18 1 Ohm
1N4148
D5
D6
K1
Q1
U5 TMC2208_Breakout
R3
R1
R8
D12
Enable
Poti
MS1
VCC
S88 Move
neg.
D1
D9



Der DFPlayer wird softwaremäßig über eine serielle Schnittstelle angesteuert. Es kann sein, dass sich die Ansteuerung mit der MobaLedLib-Schnittstelle in die Quere kommt. Dies wurde bislang nicht getestet!

Sound-Möglichkeiten

Beim Abspielen von Sounds bestehen zwei Möglichkeiten, die über das **#define ADVANCED_SOUND** gesteuert werden:

- Die „einfache“ Version (**#define ADVANCED_SOUND 0**) spielt zwei Sounds ab, einen langen beim Starten der Bewegung (das soundfile beinhaltet Hupen, Anfahren und Fahren und sollte schon ein paar Minuten dauern) und den zweiten beim Stoppen (Hupen); ein separater Bremssound ist nicht verfügbar. Dies liegt daran, dass das JQ6500 dummerweise eine kurze Pause zwischen 2 unterschiedlichen Sounds macht (mehrere ms). Diese Pause ist wahrnehmbar und fällt bei Abspielen des Bremssounds auf, ist aber bei einem abschließenden Hupen verschmerzbar.
- Die erweiterte „advanced“ Version (**#define ADVANCED_SOUND 1**) berechnet anhand der Fahrtdauer (= Entfernung zwischen Start- und Zielport) das abzuspielende Soundfile und startet einen entsprechend langen Sound, hierfür brauchen wir mehrere passgenaue Files auf dem Soundmodul. Jedes File enthält Hupen, Anfahren, Fahren, Bremsen bis zum Stopp und abschließendes Hupen. Der variable Anteil ist das Fahrgeräusch, das muss so lang sein, dass es für die entsprechende Fahrtstrecke reicht. Da hier lediglich ein MP3-Player verbaut und kein Loksounddekoder mit komplexen Soundstrukturen in einem Soundprojekt, muss man etwas in die Trickkiste greifen, um einen bewegungssynchronen Sound zu erreichen.

Empfehlung:
Für die einfache Version reicht ein JQ6500 aus. Um erste Erfahrungen zu sammeln, sollte man damit starten. Wer unbedingt einen Bremssound beim Abbremsen der Bühne kurz vor dem Zielport haben möchte, kommt um die advanced Version nicht herum. Diese ist etwas aufwendiger, da mehrere passende Soundfiles erzeugt werden müssen. Aufgrund des Speicherbedarfs ist hierfür nur der DFPlayer zu empfehlen.

Einfache Sound-Version

Für Drehscheibenbewegungen sollten 3 Soundfiles auf dem Soundmodul abgespeichert werden, aktuell werden davon nur 1 und 3 genutzt:

- Sound 1 (= file 1) –> gesampeltes File bestehend aus Hupen, Anfahren und Fahren. Je nach Drehgeschwindigkeit muss der Fahrsound mehrfach hintereinander ins File kopiert werden (z.B. mit dem Programm *Audacity*). Die Dauer sollte mehrere Minuten betragen, damit der Sound

auch beim Endlosdrehen mit Poti einige Zeit ertönt.
- Sound 2 (= file 2) –> Abbremsen und Stopp (aktuell nicht im Einsatz, da Pause zwischen den Sounds zu lange!)
- Sound 3 (= file 3) –> Hupen

Folgende #defines sind wichtig:

```
#define SOUND1_FILENR  1  // sound-file number of JQ6500/DFPlayer for turntable start and running (Hupe, Anfahren und Drehen)
#define SOUND2_FILENR  3  // sound for turntable stop (aktuell nur die Hupe)
```

Es können bei Bedarf noch weitere Sounds auf den DFPlayer gespielt werden, diese lassen sich dann über DCC-Befehle abspielen.

Advanced Sound-Version

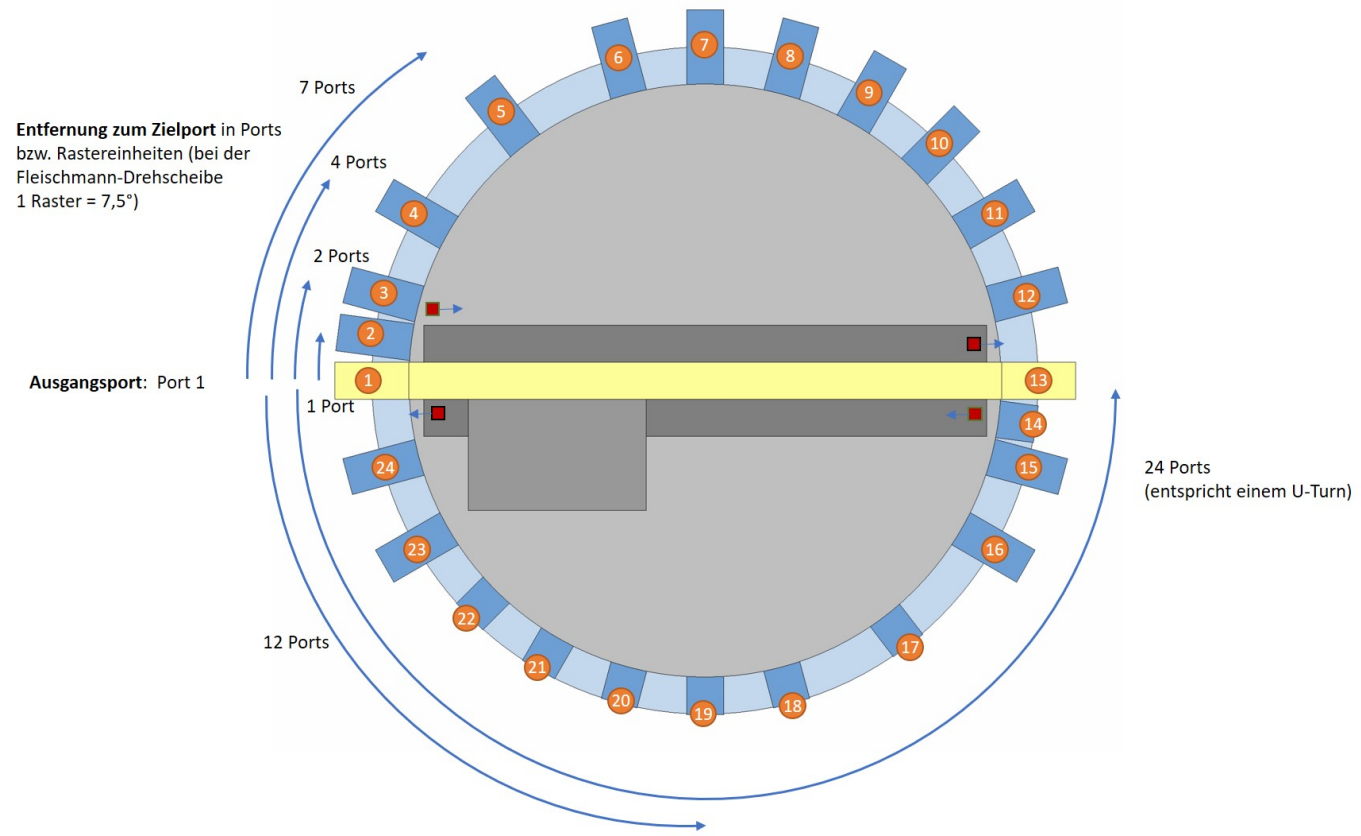
Für die advanced Version benötigt man mehrere Soundfiles und zwar je ein File für jede Fahrmöglichkeit zwischen Start- und Zielport. Aufgrund der benötigten Datenmenge bei annehmbaren Sampling-Raten ist diese Version nur mit dem DFPlayer und einer entsprechenden Speicherkarte sinnvoll zu nutzen!

Wieviele Soundfiles braucht man?

Üblicherweise liegen die Ports einer Drehscheibe in einem bestimmten Raster. Die folgenden Ausführungen beziehen sich auf eine Fleischmann-Drehscheibe. Diese besitzt maximal 48 Gleisabgänge, die im 7.5°-Raster angeordnet sind. Das Raster, auf das sich die Entfernungen zwischen Start- und Zielport beziehen, ist in **#define MAX_PORTS** festgelegt (bei Fleischmann = 48).

Die Fahrtdauer ist immer ein ganzzahliges Vielfaches der Fahrtzeit, die die Bühne für eine Rastereinheit von 7,5°, d.h. einen Port benötigt.

Die Abbildung zeigt einige beispielhafte Fahrtdauern zwischen Start- und Zielport.



Man muss nun prüfen, welche Entfernung zwischen allen möglichen Start- und Zielportkombinationen liegen, also wieviele Raster bzw. potentielle Gleisabgänge die Drehscheibe jeweils fahren muss. Das erinnert ein bisschen an frühere ADAC-Entfernungstabellen zwischen Städten.

Bei der Drehscheibe in der Abbildung hat man folgende Fahrtdauern:

Start	Ziel	Dauer in Ports (7,5°-Raster)
1	2	1
1	3	2
1	4	4
1	5	7
…	…	…
2	3	1
2	4	3
2	5	6
…	…	…

Die Drehscheibe wählt bei LocoTurn immer den kürzesten Weg zum Zielport. Daher dreht sich die Bühne maximal um 180°, also eine halbe Umdrehung. Es reicht daher aus, 24 fahrtdauerabhängige Sounds zu definieren, um alle Fahrtmöglichkeiten abzudecken.

Als Maximalausprägung braucht man bei einer Fleischmann-Drehscheibe mit 48 Abgängen folglich 24 fahrtdauerabhängige Sounds plus die 3 Standard-Soundfiles. Wer zu faul ist zu zählen, welche Kombinationen auf der Drehscheibe überhaupt auftreten, ist mit 24 +3 Sounds auf der sicheren Seite.

Bei der kleinen Fleischmann-Drehscheibe mit 24 Gleisabschlüssen braucht man 12 + 3 = 15 Sounds.

Folgende Sounds werden benötigt:

Nr.	Zweck
1	Hupen, Anfahren, Fahren (lange Version (5 min) für Endlos-Drehen)
2	*Bremsen (wird aktuell nicht benutzt!)*
3	Hupen
4	Hupen, Anfahren, Fahren, Bremsen, Hupen; Fahrtdauer 1 x 7,5°(1 Port)
5	Hupen, Anfahren, Fahren, Bremsen, Hupen; Fahrtdauer 2 x 7,5° = 15° (2 Ports)
6	Hupen, Anfahren, Fahren, Bremsen, Hupen; Fahrtdauer 3 x 7,5° = 22,5° (3 Ports)
…	…
26	Hupen, Anfahren, Fahren, Bremsen, Hupen; Fahrtdauer 23 x 7,5° = 172,5° (23 Ports)
27	Hupen, Anfahren, Fahren, Bremsen, Hupen; Fahrtdauer 24 x 7,5° = 180° (24 Ports) entspricht halber Drehung (dieses File wird auch auch bei U-Turn verwendet)

Sonderfälle

- Wird eine DS-Bewegung mittendrin abgebrochen (durch Drücken der Drehencoder-Taste oder via DCC-Befehl), ertönt die Hupe. Es gibt keinen Bremssound. Die Bühne bleibt irgendwo stehen.
- Fährt man wieder zu einem anderen Port nach einem solchen Stopp, ist u.U. nicht klar, wie lange die Fahrt dauert, da die DS ja im Nirwana, d.h. auch genau zwischen 2 Ports stehen kann. In diesem Fall wird beim Starten das lange File 1 abgespielt und beim Zielport lediglich die Hupe. Es gibt keinen Bremssound. Steht die DS in der Nähe eines Ports (+/- 10 Mikrosteps), wird jedoch der fahrtdauerabhängige Sound abgespielt.
- dito. bei Poti-Bewegung und anschließender neuer Fahrt mit Encoder oder via DCC etc.
- Bei Endlosdrehen CW/CCW wird immer das lange Sound-File Nr. 1 abgespielt (5 min ohne Bremsen und ohne Hupe am Ende). Beendet man das Endlosdrehen, wird die Hupe abgespielt.
- Bei U-Turns wird immer das File Nr. 27 gestartet, da die Bühne immer 24 Ports (= halbe Umdrehung) fährt.

Abhängigkeiten

Wenn man advanced sound verwendet, dann sollte man das **#define ENCODER_LOGIC** auf 1 setzen.

- ENCODER_LOGIC = 0: macht hier keinen Sinn, der Sound würde nach jedem Encoder-Drehen neu starten (Soundfile 4, Bewegung um ein Raster), was keinen bewegungssynchronen Sound ergeben würde.
- ENCODER_LOGIC = 1: da hier die DS erst nach Bestätigung losfährt, kann auch der richtige Sound berechnet und abgespielt werden!

Erzeugung der Soundfiles

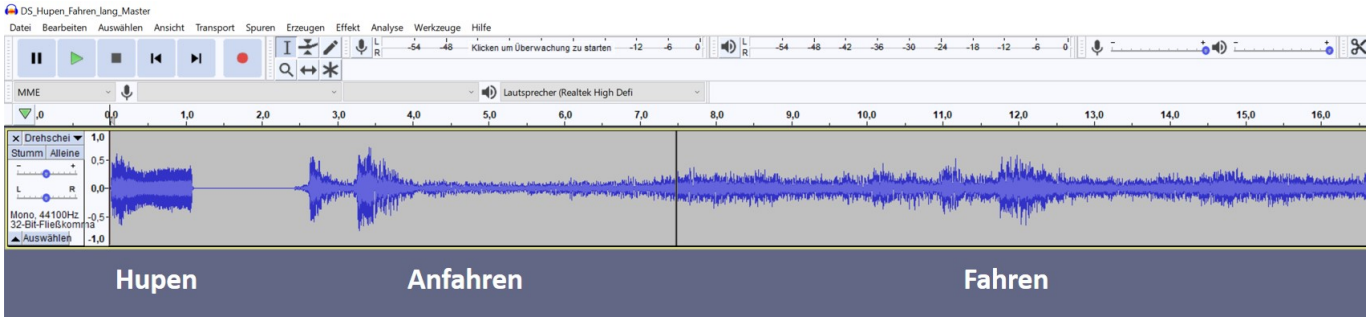
Zur Erzeugung der Soundfiles braucht man:

- geeignete Drehscheiben-Sounds (siehe die Links oben)
- eine (kostenlose) Software zur Bearbeitung von mp3, wav etc. files, z.B. Audacity
- Geduld zum Zusammenstellen der notwendigen Sounds

Zunächst bastelt man sich ein sehr langes File zusammen, bestehend aus:

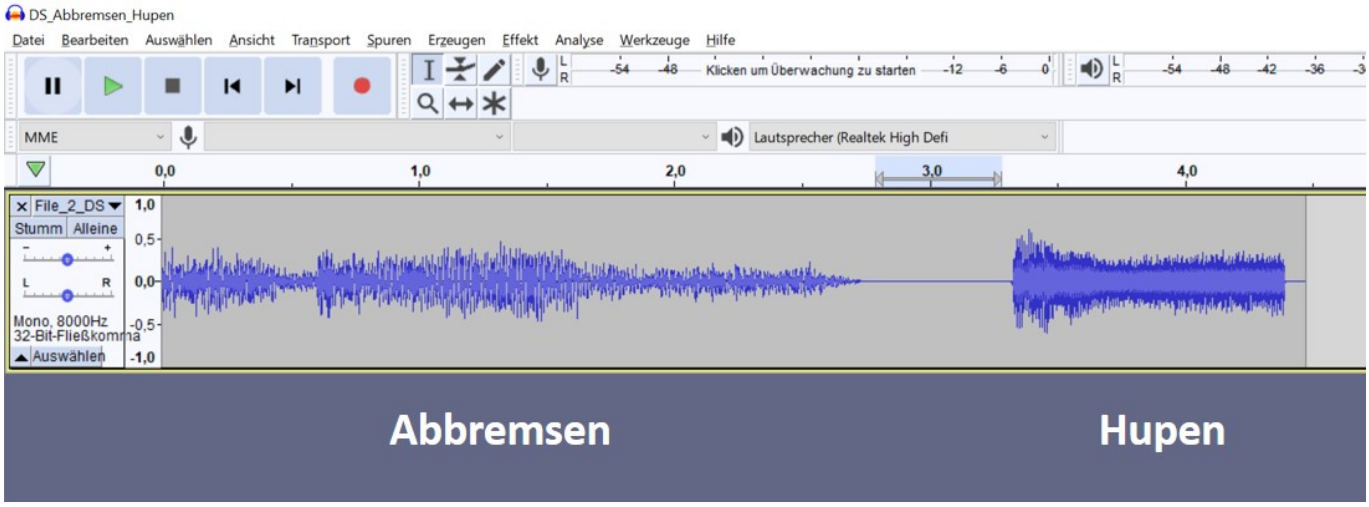
- Hupen
- Anfahren/Beschleunigen
- Fahren –> das ist der variable Teil !

Das Fahrgeräusch kopiert man hierfür mehrfach hintereinander in das lange Soundfile, damit es ca. 5 min lang dauert. Dieses File brauchen wir später auch als File 1 (siehe oben).



Weiterhin brauchen wir ein zweites File, das folgendes enthält:

- Abbremsen
- und ggf. abschließendes Hupen



Dauer der Soundfiles

Dann misst man auf der realen Drehscheibe, wie lange der Sound dauern muss, um die jeweilige Fahrt vom Start- zum Zielport abzudecken. Dies hängt ab von der Steppermotor-Geschwindigkeit (**#define MOVE_SPEED1**) und der Anfahr-/Bremsrampe (**#define STEPPER_RAMP_LENGTH**). Bevor man die Sounds erzeugt, sollte man diese beiden Parameter final festgelegt haben, sonst muss man die ganze Soundbastelarie erneut durchführen.

Ändert man später im Betrieb die Geschwindigkeit (z.B. über der seriellen Monitor oder DCC-Kommandos), dann passen die soundfiles natürlich nicht mehr zum Bewegungsablauf. Der Sound wird folglich entweder zu früh fertig sein, oder zu spät aufhören!

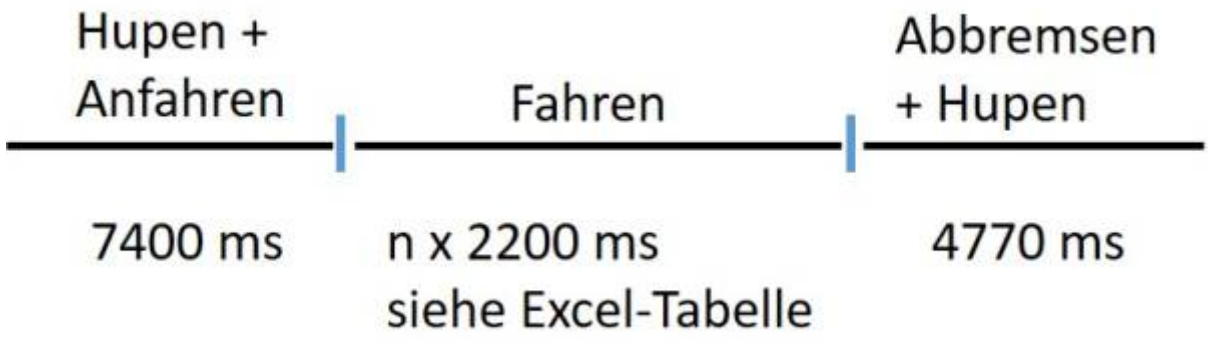
Zur Messung der echten reinen Fahrtdauern kann man im Turntable_config.h das **#define FAHRTDAUER_MESSEN** aktivieren, um die Dauer im seriellen Monitor der Arduino IDE ausgeben zu lassen. Die Fahrtdauer misst die Zeit, in der sich die Bühne bewegt, also inklusive des Abbremsens. Hierfür muss man ein paar #defines einstellen:

```
#define USE_DCC 0
#define USE_SERIAL_INPUT 0
#define ENABLE_DPRINTF 1
#define FAHRTDAUER_MESSEN 1
```

Anschließend speichert man die Mess-Werte am besten in einem Excel und kann dann recht einfach berechnen, welche Soundanteile erforderlich sind, um eine bestimmte Fahrtdauer abzudecken.

Da es nicht auf die ms ankommt, werden die Dauern in Spalte D manuell gerundet. Das reine Fahren sollte einen linearen Verlauf haben und proportional zur Anzahl fahrender Drehscheiben-Raster sein. Im Excel erkennt man, dass die Fahrtdauer um 2200 ms von Zeile zu Zeile, d.h. für jeden zusätzlichen Port, zunimmt. Von der gerundeten Fahrtdauer zieht man die Zeit für das Abbremsen ab (2770 ms) und erhält die Werte in Spalte F. Hierzu addiert man das Hupen und Hochfahren des Motors (7400 ms) und erhält die Spalte G.

Bei mir dauern die einzelnen Soundbestandteile z.B.:



Der Sound für das Abbremsen dauert im Beispiel 2770 ms und das Hupen zusätzlich 2000 ms, macht rechts die 4770 ms.

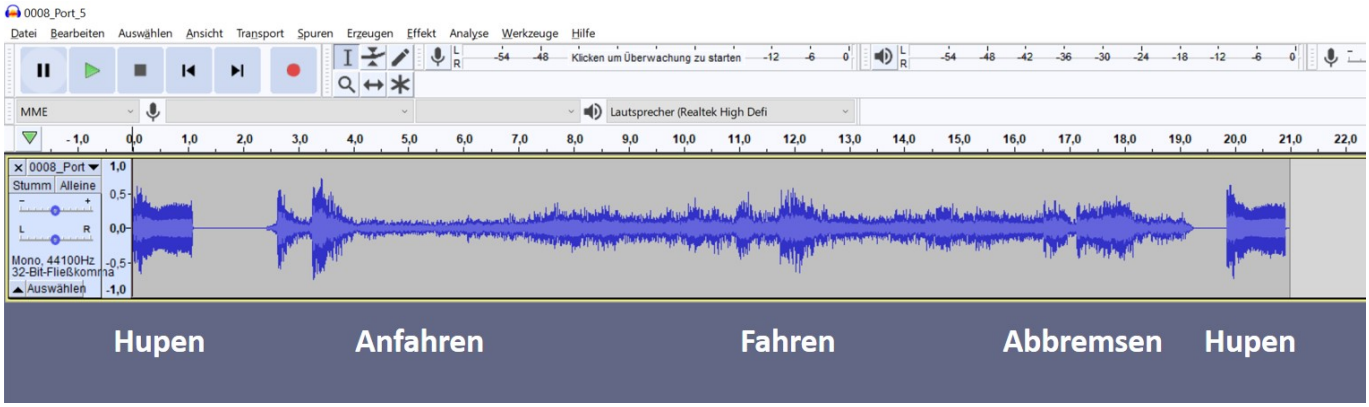
	A	B	C	D	E	F	G
1	**Ports to go**	**Sound-File #**	**Dauer im Sketch ms**	**Gerundete Dauer**	**Dauer Abbremsen**	**Dauer Fahrsound**	**Länge ab Start**
2	**1**	Sound-Nr: 4 2855	2855	3000	2770	230	7630
3	**2**	Sound-Nr: 5 5089	5089	5200	2770	2430	9830
4	**3**	Sound-Nr: 6 7305	7305	7400	2770	4630	12030
5	**4**	Sound-Nr: 7 9519	9519	9600	2770	6830	14230
6	**5**	Sound-Nr: 8 11757	11757	11900	2770	9130	16530
7	**6**	Sound-Nr: 9 13968	13968	14100	2770	11330	18730
8	**7**	Sound-Nr: 10 16189	16189	16300	2770	13530	20930
9	**8**	Sound-Nr: 11 18421	18421	18500	2770	15730	23130
10	**9**	Sound-Nr: 12 20637	20637	20700	2770	17930	25330
11	**10**	Sound-Nr: 13 22855	22855	23000	2770	20230	27630
12	**11**	Sound-Nr: 14 25088	25088	25200	2770	22430	29830
13	**12**	Sound-Nr: 15 27305	27305	27400	2770	24630	32030
14	**13**	Sound-Nr: 16 29522	29522	29600	2770	26830	34230
15	**14**	Sound-Nr: 17 31753	31753	31900	2770	29130	36530
16	**15**	Sound-Nr: 18 33972	33972	34100	2770	31330	38730
17	**16**	Sound-Nr: 19 36171	36171	36300	2770	33530	40930
18	**17**	Sound-Nr: 20 38423	38423	38500	2770	35730	43130
19	**18**	Sound-Nr: 21 40638	40638	40700	2770	37930	45330
20	**19**	Sound-Nr: 22 42854	42854	43000	2770	40230	47630
21	**20**	Sound-Nr: 23 45069	45069	45200	2770	42430	49830
22	**21**	Sound-Nr: 24 47301	47301	47400	2770	44630	52030
23	**22**	Sound-Nr: 25 49536	49536	49600	2770	46830	54230
24	**23**	Sound-Nr: 26 51755	51755	51900	2770	49130	56530
25	**24**	Sound-Nr: 27 53969	53969	54100	2770	51330	58730
26							
27							
28	Stepper-Geschwindigkeit 600 Steps/10 s, Rampe für Beschleunigung 150						

Das Excel-File steht hier zur Verfügung:

berechnung_fahrtdauerabhaengige_sounds.xlsx

Aus dem langen Maximal-File (Hupen + Anfahren + Fahren) nimmt man dann den jeweiligen Anteil in ms (im Excel Spalte „Länge ab Start“), kopiert ihn in ein neues Audacity file und fügt am Ende den Abbremsen + Hupen Sound aus dem zweiten File ein. Dann noch als mp3 File abspeichern, mono reicht aus und schon hat man ein entsprechendes Soundfile, um von A nach zu B zu fahren. Das ganze macht man nun noch 23 Mal, bis man alle Möglichkeiten abgedeckt hat.

Hier als Beispiel das Soundfile, um 5 Ports weit zu fahren (also 5 x 7,5°):



Das #define **DELAY_TURN_START_SOUND** verzögert die DS-Bewegung bis zum Zeitpunkt, wann das reine Fahrgeräusch beginnt. In meinem Beispiel sind das 7400 ms; bei Verwendung eines DFPlayers muss man ca. 300 ms dazu addieren, da dieses Soundmodul verzögert über softwareserial startet. In Summe habe ich daher das #define auf 7700 gesetzt.

Speicherung der Soundfiles

Die Dateien speichert man auf einer SD-Karte in einem Verzeichnis „Laufwerk:\mp3\“ ab.

Die Dateien werden durchnummeriert, 4-stellig, führende Nullen, Rest vom Dateinamen kann anderweitig genutzt werden, um dem Sound einen sprechenden Namen zu geben:

Name	Änderungsdatum	Typ	Größe
0001_DS_Hupen_Fahren_lang_Master.mp3	28.04.2023 13:25	MP3-Audioformat	4.502 KB
0002_DS_Bremsen.mp3	09.04.2022 08:44	MP3-Audioformat	9 KB
0003_DS_Hupe.mp3	12.03.2023 10:18	MP3-Audioformat	15 KB
0004_Port_1.mp3	28.04.2023 12:11	MP3-Audioformat	131 KB
0005_Port_2.mp3	28.04.2023 12:11	MP3-Audioformat	147 KB
0006_Port_3.mp3	28.04.2023 12:12	MP3-Audioformat	196 KB
0007_Port_4.mp3	28.04.2023 12:05	MP3-Audioformat	228 KB
0008_Port_5.mp3	28.04.2023 12:06	MP3-Audioformat	261 KB
0009_Port_6.mp3	28.04.2023 12:08	MP3-Audioformat	294 KB
0010_Port_7.mp3	28.04.2023 12:17	MP3-Audioformat	325 KB
0011_Port_8.mp3	28.04.2023 12:16	MP3-Audioformat	358 KB
0012_Port_9.mp3	28.04.2023 12:18	MP3-Audioformat	392 KB
0013_Port_10.mp3	28.04.2023 12:21	MP3-Audioformat	425 KB
0014_Port_11.mp3	28.04.2023 12:23	MP3-Audioformat	457 KB
0015_Port_12.mp3	28.04.2023 12:24	MP3-Audioformat	489 KB
0016_Port_13.mp3	28.04.2023 12:57	MP3-Audioformat	521 KB
0017_Port_14.mp3	28.04.2023 13:00	MP3-Audioformat	555 KB
0018_Port_15.mp3	28.04.2023 13:01	MP3-Audioformat	589 KB
0019_Port_16.mp3	28.04.2023 13:03	MP3-Audioformat	621 KB
0020_Port_17.mp3	28.04.2023 13:04	MP3-Audioformat	653 KB
0021_Port_18.mp3	28.04.2023 13:20	MP3-Audioformat	684 KB
0022_Port_19.mp3	28.04.2023 13:19	MP3-Audioformat	719 KB
0023_Port_20.mp3	28.04.2023 13:17	MP3-Audioformat	787 KB
0024_Port_21.mp3	28.04.2023 13:16	MP3-Audioformat	785 KB
0025_Port_22.mp3	28.04.2023 13:15	MP3-Audioformat	817 KB
0026_Port_23.mp3	28.04.2023 13:10	MP3-Audioformat	847 KB
0027_Port_24.mp3	28.04.2023 13:09	MP3-Audioformat	883 KB

Es können bei Bedarf noch weitere Sounds auf den DFPlayer gespielt werden, diese lassen sich dann über DCC-Befehle abspielen. Die Auswahl erfolgt über die #defines:

```
#define DCC_SOUNDFILE_1   1   // File-Nr. auf dem Soundmodul, das abgespielt
wird mit dem entsprechenden DCC-Befehl
#define DCC_SOUNDFILE_2   2   // J6500; Dateien stehen im Rootverzeichnis,
Reihenfolge geht nach Reihenfolge des Kopierens auf das Modul
#define DCC_SOUNDFILE_3   3   // DFPlayer: Dateien müssen im \mp3-Folder
stehen, Nomenklatur 0001_beliebiger Text (4-stellige Nummer, führende Nullen
```

```
+ sprechender Text)
#define DCC_SOUNDFILE_4   28
#define DCC_SOUNDFILE_5   29
#define DCC_SOUNDFILE_6   30
```
