Holzfäller und Baum bewegt mit Sound

Der Holzfäller, wie programmieren?

Zunächst ein paar Worte zum Holzfäller vorab.
Das Programm ist für meine Hardwarespezifikation entstanden und muss bei Verwendung von anderen Komponenten evtl. angepasst werden.
Den Sound habe ich auf meiner SD-Karte z.B. auf Position 6 stehen, wie nachfolgend noch erläutert wird.

Die Vorlage für dieses Projekt hat mir ein Bekannter von unserem Winnender Stummi-Stammtisch zukommen lassen.
Hier findet man die originale Beschreibung, samt Sounddatei und Schaltplan zum Download:

https://erster-maerklin-club.de/wp-content/uploads/2021/11/FallenderBaum-Bauanleitung.zip

Mein erste Gedanke war, als ich es gelesen habe, das muss sich doch mit der MobeLedLib umsetzen lassen.

So, nun geht es los.

Hardwarekonfiguration/ Ausgangslage

Grundplatine 100 bestückt in Grundversion, bzw. mit der Taster-Erweiterung für Funktionstests und Servoeinstellung (https://wiki.mobaledlib.de/anleitungen/bauanleitungen/100de_hauptplatine_v1-6_erweiterungen)

Adapterplatine 200 (https://wiki.mobaledlib.de/anleitungen/bauanleitungen/verteilerplatine_200de), daran sind angeschlossen:

- 1 Servoplatine mit 2 Micro-Servos an Stecker 1 (https://wiki.mobaledlib.de/anleitungen/bauanleitungen/510de_modul_servo)
- 1 Soundmodul MP3-TF-16P an Stecker 2 (https://wiki.mobaledlib.de/anleitungen/bauanleitungen/501de_sound_mp3tf16_v1-1)

Was soll passieren?

- Kettensäge startet und wird angelegt (Baumfäller bewegt sich leicht)
- Mann setzt mit Kettensäge nach (Baumfäller bewegt sich kurz, Baum wackelt leicht)

- 2. Mal nachsetzen der Kettensäge (Baumfäller bewegt sich kurz, Baum wackelt)
- Baum wackelt
- Baumfäller dreht sich weg und warnt per Ruf „Baum“
- Der Baum knickt weg
- Der Baum bleibt kurz an einem anderen Baum hängen
- Der Baum fällt und bleibt liegen
- eine Weile passiert nichts
- zu guter Letzt richtet sich der Baum wieder auf und der Ablauf beginnt von vorne

Wer möchte kann den ganzen Ablauf natürlich noch weiter verfeinern.

Was muss programmiert werden?

1. Pattern-Ablauf Servo 1, der Baumfäller
2. Pattern-Ablauf Servo 2, der Baum
3. Pattern-Ablauf Sound (für das Soundmodul)

Was müssen wir für die Einstellung des Sounds wissen?

Das unter dem Soundmodul befindlichen WS2811-Modul wird über die Kanäle „Rot“ und „Grün“ angesteuert und gibt
die Schalterbefehle über die Helligkeitswerte an das Soundmodul weiter.
Dabei werden widerstandskodierte Tasten simuliert, mit denen das Soundmodul normalerweise angesteuert wird.

So nun müssen wir noch die Helligkeitswerte für die Sounds rausbekommen.
Dies „erforschen“ wir über die „MobaLedLib.h“ Bibliothek.
Wir finden die Datei unter Dokumente/Arduino/libraries/MobaLedLib/src/MobaLedLib.h

Diese öffnen wir mit z.B. dem Texteditor und scrollen bis zum

```
//------------------------------------- MP3-TF-16P Sound modul -----------
---------------------------------\\
```

	Helligkeit	**ADKey1**	**ADKey2**
#define SOUND_ADKEY10	11	Play Mode	14
#define SOUND_ADKEY9	17	U/SD/SPI	13
#define SOUND_ADKEY8	22	Loop All	12
#define SOUND_ADKEY7	25	Pause/Play	11
#define SOUND_ADKEY6	29	Prev/Vol-	10
#define SOUND_ADKEY5	37	Next/Vol+	9
#define SOUND_ADKEY4	49	4	8
#define SOUND_ADKEY3	70	3	7
#define SOUND_ADKEY **2**	**134**	2	**6**

	Helligkeit	ADKey1	ADKey2
#define SOUND_ADKEY1	255	1	5

Nachdem wir den **Sound 6** brauchen (hatte ich ja ganz oben geschrieben, dass meine Sounddatei auf Platz 6 der SD-Karte liegt),
sind für uns die Werte ADKEY2 – Helligkeit 134 – Sound 6 (in der Tabelle rot hinterlegt) interessant.
Steht der Sound auf anderer Stelle auf der SD-Karte, so muss man natürlich dann die entsprechenden Werte aus der Tabelle raussuchen.

Um die Werte zum Soundmodul schicken zu können, benötigen wir noch die Schaltzeiten,
diese sind **200 ms** und **10 ms**.

Nun können wir den Editor wieder schließen.
Fall versehentlich was geändert wurde ***nicht*** speichern!!!

Was müssen wir über die Servoansteuerung wissen?

Die Servo-Platine setzt die von der Hauptplatine über den Verteiler kommenden RGB-Signale um.
Somit wird ein Kanal einem Servo auf der Servo-Platine zugeordnet (z.B. Kanal 1 = Rot)
Die Bewegung wird über den Helligkeitswert des Kanals (z.B. rot) in den Werten von 10-210 gesteuert.
(für diejenigen, die es genauer wissen möchten: Servo State Diagram)

Die Servo-Einrichtung für die Geschwindigkeit und die Drehwinkel ist im Wiki beschrieben,
hier gehe ich nicht gesondert drauf ein. Hier der Link zur Anleitung:
https://wiki.mobaledlib.de/anleitungen/spezial/tiny-uniprog?do=export_pdf

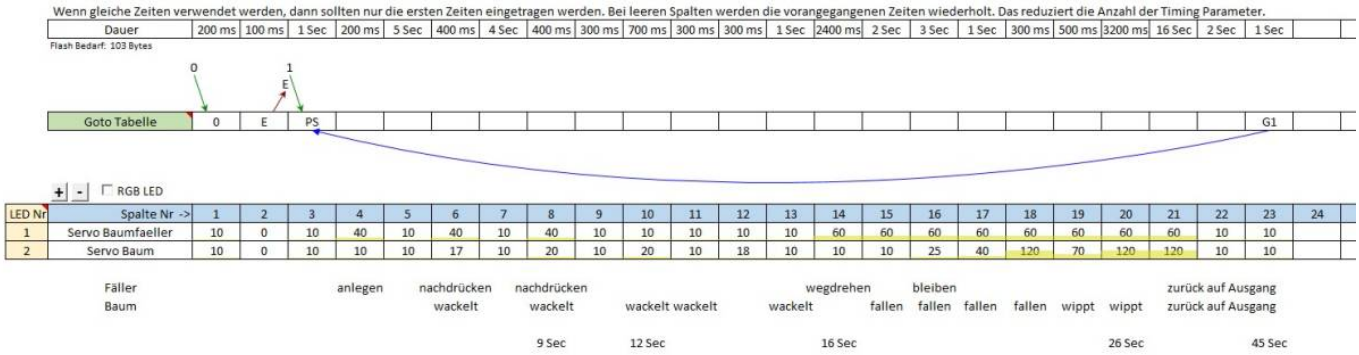
Analog gilt dies natürlich auch für Servo 2 und Servo 3, hier sind dann die Farben „Grün“ und „Blau“

Nun geht es an die Programmierung des Pattern 1 (die Servos bewegen)

Wir öffnen das Programm Pattern_Generator, dort machen wir als erstes ein neues Blatt
(Einstellungen übernehmen? Nein)
Dann geben wir dem Registerblatt einen Namen, wie zum Beispiel: „Baumfäller bewegen“

Nun geben wir im gelben Kasten folgende Werte ein:

Als nächstes benennen wir das Makro mit einem aussagekräftigem Namen, wie zum Beispiel: „Baumfaeller_Baum“
Jetzt geht es ans ausfüllen der Tabelle:



Die Zeile „Spalte Nr.“ benennt dabei unsere Servos (im Beispiel „Servo1“ wird „Servo Baumfaeller“, „Servo2“ wird „Servo Baum“)

- Spalte Nr.1 sorgt dafür, dass der Servo in die Ausgangsposition fährt
- Spalte Nr.2 schaltet den Servo ab, damit er nicht brummt

In die Goto-Tabelle geben wir den Wert „E“ für Ende ein, dies beendet die erste Startsequenz, welche wir später für den Sound brauchen.
Die weiteren Spalten dienen dazu, den Servo1 = Servo Baumfaeller und den Servo2 = Servo Baum abgestimmt mit dem Soundablauf zu steuern.
Der Sound hat eine Gesamtlänge von 26 Sekunden.
Ich habe mir die Mühe gemacht, nachdem es ja nur eine Sounddatei für den ganzen Ablauf ist, diese für mich in einzelne Abschnitte/ Ereignisse zu zerlegen. Das Ergebnis sind die Zeiten in der obigen Tabelle.

Um die Servobewegung auszuführen benötigen wir die Werte zwischen 10 und 210 (minimaler bzw. maximaler Ausschlag).
Den tatsächlichen Ausschlag bestimmt man im Farbtestprogramm über die Servoeinstellungen.
Die Werte 10 und 210 liegen außerhalb des normalen Bewegungsbereichs, was gemacht wird, damit der Servo nicht brummt.
Das Brummen entsteht, da die Messung der WS2811-Helligkeiten und die Messung der Servoposition geringfügige Störungen haben.

Wichtig ist in die Goto-Tabelle in Spalte 3 den Wert „PS“ einzugeben (P = Position Start und S = Start).
Sowie am Ende den Wert G1 (Goto Wert 1, also zu PS), damit der programmierte Prozess ab der Stelle PS von vorne beginnt.
Der Ablauf läuft so lange immer wieder durch, bis wir ihn beenden, z.B. durch drücken eines Tasters.

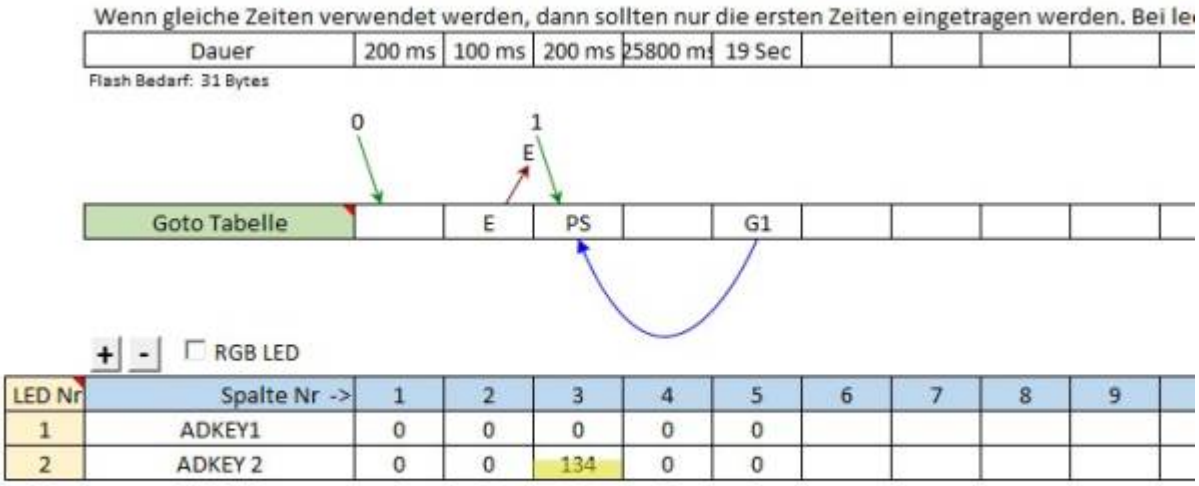
Nun müsste die Tabelle wie oben ausgefüllt aussehen.
Die unter der Tabelle angegebenen Hinweise dienen zur Erklärung des Ablaufs, um zu erkennen wann was gemacht wird.

Speichern nicht vergessen!!!

So nun geht es an die Programmierung des Pattern 2 für den Sound

Hierfür machen wir wieder ein neues Blatt (Einstellungen übernehmen? Nein)
Dann tragen wir die Werte in den gelben Kasten ein:

Nun geben wir dem Registerblatt wieder einen Namen, wie zum Beispiel: „Baumfaeller Sound“
Jetzt gehen wir wieder in die Tabelle und füllen diese wie folgt aus:



Für die Soundansteuerung benötigen wir zwei Zeilen, nämlich „ADKEY1“ und „ADKEY2“, diese schreiben wir neben LED-Nr. 1 und LED-Nr. 2.
Wir erinnern uns, das sind die Ansteuerwerte aus der Editor-Geschichte von oben.

Nun geben wir in Spalte 1 unter Dauer 200 ms ein (den Wert hatten wir oben als Zeitwert rausgesucht) und in Spalte 2 eine Dauer von 100 ms.
Das machen wir, damit die Zeiten beim Einschalten des gesamten Ablaufs synchron laufen (siehe Pattern 1, Spalte 1 und Spalte 2)

In der Goto-Tabelle geben wir wieder in Spalte 2 ein E für Ende ein und in Spalte 3 PS für Start.
Nun kommen wir zur Spalte 3, hier starten wir im ADKEY2 mit 200 ms um den Sound mit der Helligkeit 134 zu starten.
(diese Werte haben wir oben der 1. Tabelle entnommen)

ADKEY2 füllen wir mit Nullen, da wir hiermit keinen Sound starten. Wir brauchen die Zeile aber, damit ADKEY2 abgefragt wird.

Nachdem unsere Sounddatei 26 Sekunden lang ist, müssen wir in Spalte 4 die Restdauer des Sounds definieren, also 25800 ms.
Somit ergeben sich insgesamt aus Spalte 3 und 4 die 26 Sekunden Gesamtdauer.

Spalte 5 gibt eine Wartezeit von 19 Sekunden vor, in der nichts passiert.
Irgendwann muss sich ja der Baum wieder aufrichten und der Holzfäller wieder in seine Ausgangsposition zurückkehren.

Wichtig ist in Spalte 5 wieder G1 einzugeben, damit nach dem Ablauf wieder mit Spalte 3 begonnen werden kann.

Speichern nicht vergessen!!!

jetzt noch die Programmierung im Prog_Generator

Im Programm-Generator machen wir uns, indem wir auf dem Reiter DCC einen Rechtsklick machen unter
„Verschieben und Kopieren“ das Unterfenster auf und setzen einen Haken bei „Kopie erstellen“, dann Klick auf „OK“.

Zeile eins sollte die Heartbeat RGB beinhalten und in der Spalte B angehakt sein.

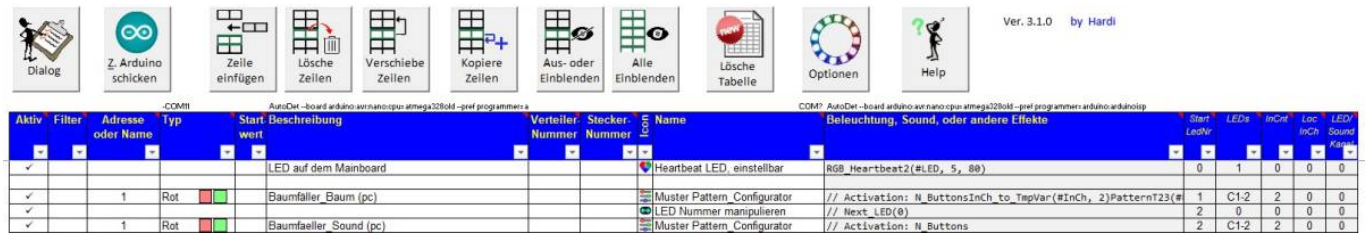
Nun gehen wir nochmal in den Pattern-Generator und übertragen die Datei „Baumfaeller_Baum“.
Diese Datei fügen wir z.B. in Zeile 5 ein.
Dann geben wir unter „Adresse oder Name“ eine DCC-Adresse (z.B. 1) ein und wählen auf Nachfrage den Schalter „Rot“ (es werden automatisch rot und grün angelegt)

Dann machen wir in der Zeile 6 in der Spalte „Name“ einen Doppelklick, es sollte sich das Kontextmenue mit
Auswahlmöglichkeiten öffnen.
Unter dem Punkt „Manipulation“ wählen wir „LED manipulieren“ aus (dazu muss der Expertenmodus aktiviert sein)
Parameter 0 als Änderung von StartLedNr. eingeben und Kanal 0 lassen.

Nun gehen wir nochmal in den Pattern-Generator und übertragen die Datei „Baumfaller_Sound“, wählen die Zeile 7 aus und fügen diese ein.
Dann geben wir wieder, wie zuvor, die gleiche DCC-Adresse (z.B. 1) ein und wählen auf Nachfragen den Schalter „Rot“ aus.
Es werden wieder Rot und Grün angelegt.

Speichern nicht vergessen!!!

So sollte das Programm im Prog_Generator aussehen:



Nun können wir das Programm an den Arduino schicken.

Endlich sind wir

fertig !!

Jetzt müsste es, nachdem man in der Excel den „Grünen Punkt“ bei „Baumfaeller_Baum“ oder „Baumfaeller_Sound“ anklickt losgehen.

Stoppen kann man das Ganze dann wieder mit dem „Roten Punkt“

Bei „Stopp“ (Roter Punkt) läuft die Sounddatei bis zum Ende durch, die Servos stoppen in der Position in der sie gerade sind.
So lange sollte auch gewartet werden, bevor man das Projekt neu startet.

Sobald neu gestartet wird, fahren die Servos in die Ausgangsposition und das Programm startet von vorn.

Da der Sound sehr laut ist, habe ich im Prog_Generator noch weitere Zeilen eingefügt um die Lautstärke zu regulieren.
Hierzu benutze ich die Taster mit den dazugehörigen LEDs der Hauptplatine.

- Switch D1 (der linke Taster) macht in diesem Beispiel „leiser“
- Switch D2 (der mittlere Taster) macht in diesem Beispiel „lauter“

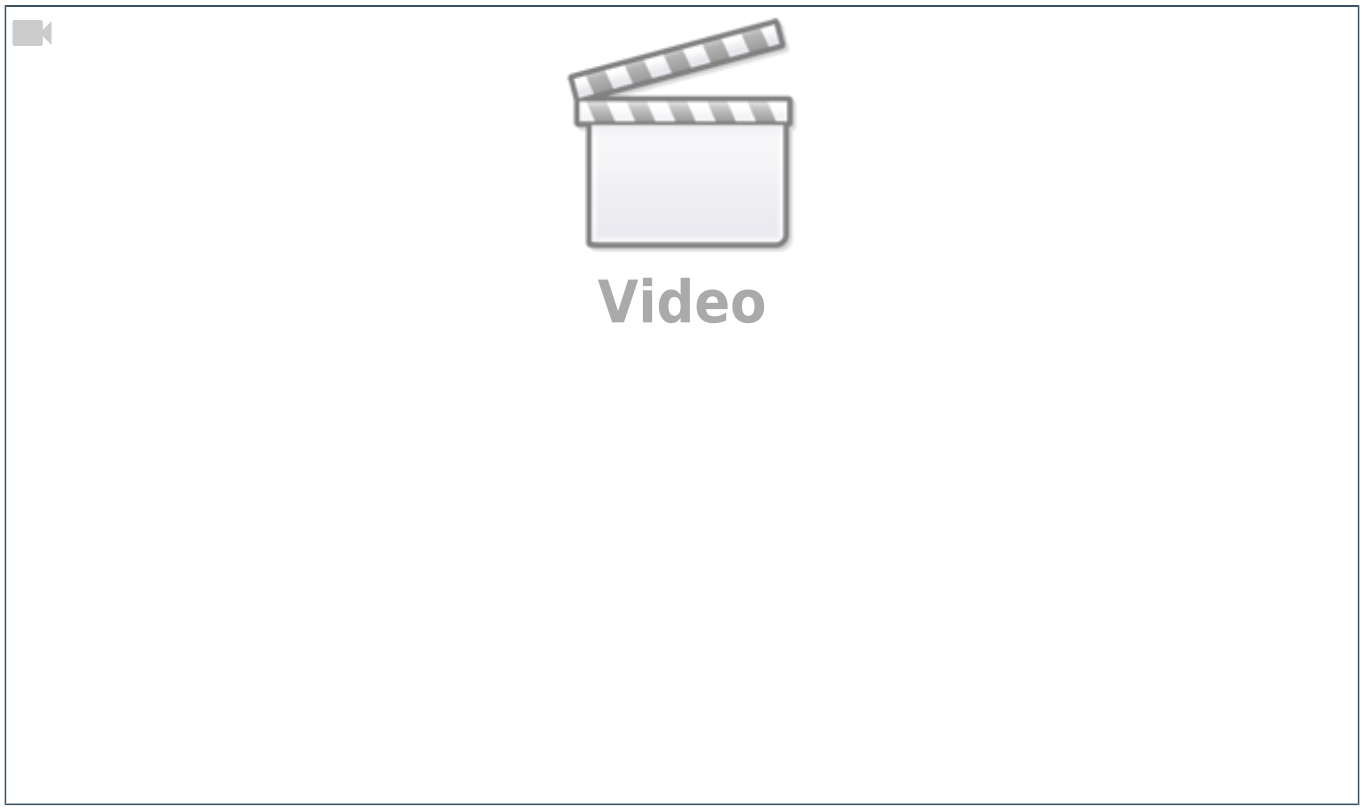
Hier die entsprechenden Zeilen für die Programmierung:

8	✓		SwitchD1								Leiser	Sound_New_DecVol(#LED, #InCh, 5)	2	^ C1-2	1	0	0
9	✓		SwitchD2								Lauter	Sound_New_IncVol(#LED, #InCh, 3)	2	^ C1-2	1	0	0
10	✓		SwitchD1								LEDs der Hauptplatine steuern	Mainboard_LED(1, #InCh)			1	0	
11	✓		SwitchD2								LEDs der Hauptplatine steuern	Mainboard_LED(2, #InCh)			1	0	
12	✓		SwitchD3								LEDs der Hauptplatine steuern	Mainboard_LED(3, #InCh)			1	0	

Nun wünsche ich Euch viel Spaß beim Nachbau und Baumfällen

Hier habe ich noch ein Video des „provisorischen“ Aufbaus für Euch.

Leider habe ich derzeit noch keine Holzfäller-Figur, daher arbeitet hier eine andere Person.
Aber den Ablauf kann man sehr gut nachvollziehen, denke ich.



© Dieses Projekt wurde durch Jürgen (fromue) zur Verfügung gestellt.

From:
https://wiki.mobaledlib.de/ - **MobaLedLib Wiki**

Permanent link:
https://wiki.mobaledlib.de/anleitungen/spezial/codevorlagen/holzfaeller?rev=1639166263

Last update: **2021/12/10 20:57**