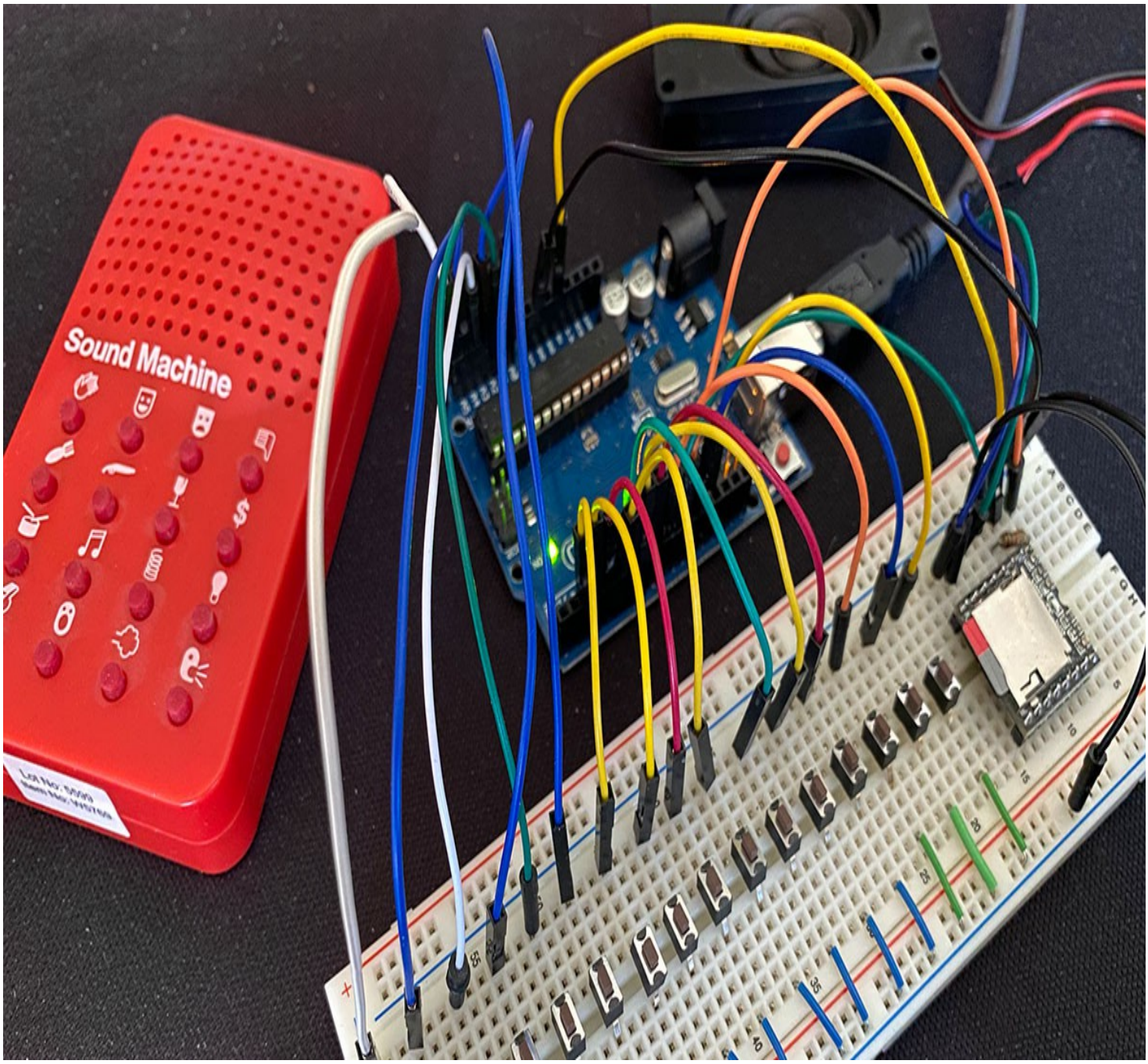
Sound Machine mit Arduino



Sound Machines und Sound Boards sind super. Ich meine die kleinen kompakten Geräte, die aus keinem Startup-Büro wegzudenken sind. 16 Knöpfe lösen die unterschiedlichsten Sounds aus und sorgen sowohl für erheiterte, als auch für genervte Gesichter. Perfekt also für den Büroalltag.

Manchmal wünscht man sich aber, die Sounds austauschen zu können. Kann man so eine Sound Machine nicht einfach mit Arduino bauen? Ja! Das geht. Wie genau, das erfährst du hier.

Bauteile

- Arduino Set, Breadboard, Kabel, etc.*

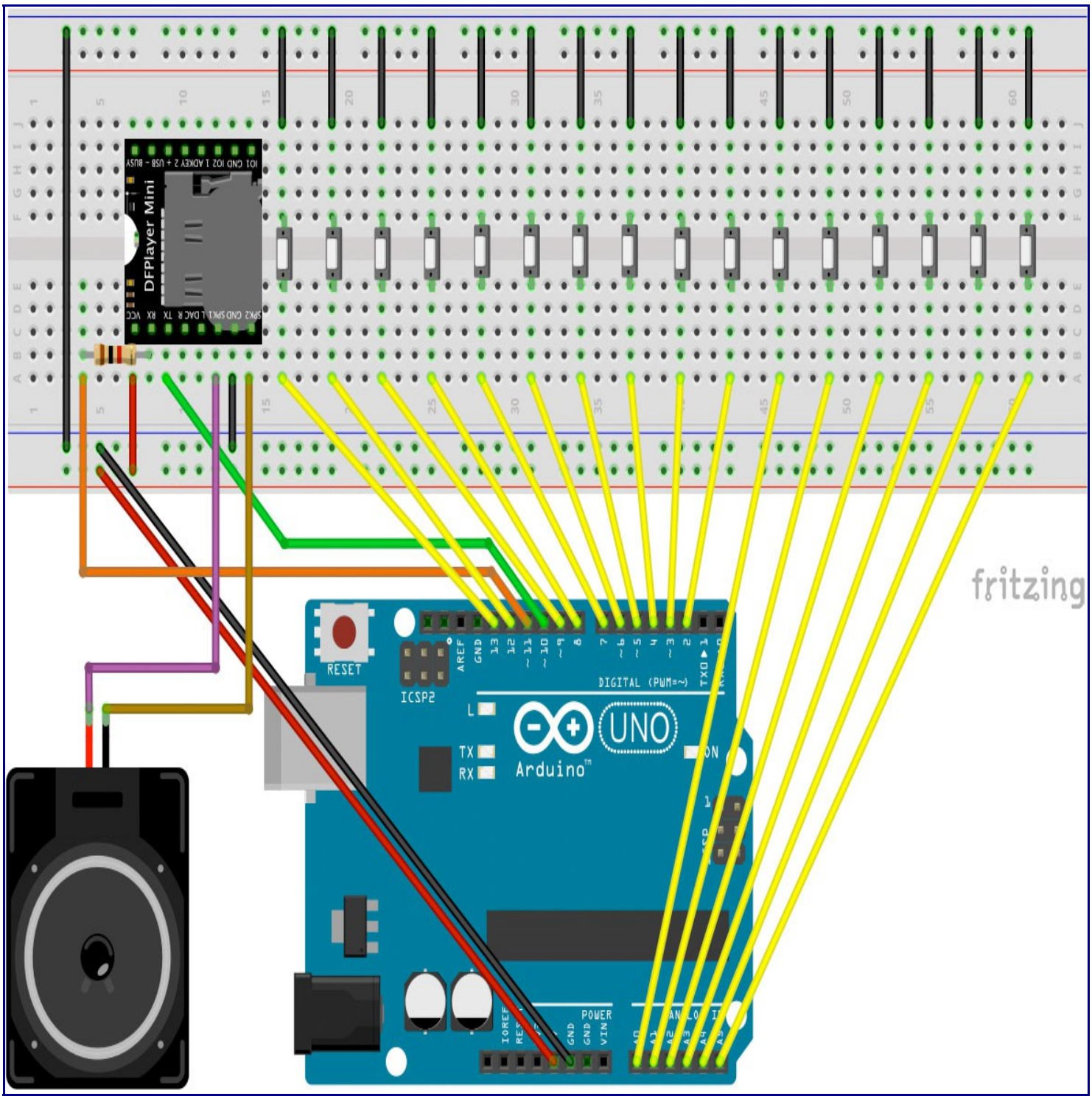
https://starthardware.org/sound-machine-mit-arduino/

- 16 Taster
- Widerstand 1kOhm
- DFPlayer Mini*
- MicroSD-Karte*
- Lautsprecher*

Schaltplan

Die Schaltung besteht aus dem DFPlayer Mini, einen Lautsprecher und 16 Taster. Der DFPlayer ist mit GND und 5V versorgt und an die SoftwareSerial-Pins 10 und 11 verbunden. Die Verbindung mit dem Pin 11 bekommt noch einen 1kOhm Widerstand. Der Lautsprecher ist direkt an den DFPlayer angeschlossen.

Wir verwenden für die Taster, die später die Sounds auslösen sollen, die internen Pullup-Widerstände des Arduino-Boards. Alle Taster sind mit dem GND und jeweils einem digitalen oder analogen Pin des Arduinos verbunden. Die analogen Input-Pins lassen sich nämlich auch einfach als digitale Pins zum Auslesen von Tastern verwenden.



Mehr Infos zum DFPlayer Mini findest du übrigens hier: DFPlayer Mini mit Arduino.

SD-Karte

Kopiere die folgenden Sounds auf eine SD-Karte. Achte darauf, dass sie im FAT16 oder FAT32 Format formatiert ist. Die einzelnen MP3 Dateien müssen im Ordner »MP3« im Hauptverzeichnis (root) der SD-Karte zu finden sein.

Mac-User sollten beim Formatieren darauf achten, dass sie auch Master Boot Record im Festplattendienstprogramm auswählen.

Download: Sounds

Die Dateien sind von folgenden Urhebern und stehen unter CC-BY Lizenz:

1. Lachen (CC-BY LittleRainySeasons)
2. Airhorn (CC-BY SPANAC)
3. Rimshot (CC-BY SPANAC)
4. Cartoon Clock (CC-BY SPANAC)
5. Sad Trumpet(CC-BY SPANAC)
6. Boo(CC-BY SPANAC)
7. Wah-Wah-Wah (CC-BY ALEXANDAR)
8. Badum-Tss (CC-BY ALEXANDAR)
9. Car Lock Sound (CC-BY SPANAC)
10. Burp (CC-BY SPANAC)
11. Coo-Coo-Clock (CC-BY SPANAC)
12. Bell-Rings-Short (CC-BY SPANAC)
13. Crash-Sound (CC-BY SPANAC)
14. Clapping (CC-BY SPANAC)
15. Hammer (CC-BY SPANAC)
16. Extreme Alarm Sound (CC-BY SPANAC)

Code für die Sound Machine mit Arduino

Das Programm deklariert 16 Arduino-Pins als INPUT_PULLUP und fragt diese im Loop ständig ab. Wird einer der Button gedrückt, wird ein Sound ausgelöst.

Die Debugging-Funktionen habe ich drin gelassen, da es die Fehlersuche enorm erleichtert.

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

SoftwareSerial mySoftwareSerial(10, 11); // RX, TX
DFRobotDFPlayerMini myDFPlayer;

int buttonPins[16] = {19,18,17,16,15,14,2,3,4,5,6,7,8,9,12,13};   // Pins, an denen die Taster
angeschlossen sind

void setup() {
  mySoftwareSerial.begin(9600);           // Start der SoftwareSerial Kommunikation
  Serial.begin(115200);                   // Start der "normalen" seriellen Kommunikation für
den Serial Monitor

  if (!myDFPlayer.begin(mySoftwareSerial)) {  // Verbindung der Software Serial Kommunikation
mit dem DFPlayer
    Serial.println(F("Unable to begin:"));
    Serial.println(F("1.Please recheck the connection!"));
    Serial.println(F("2.Please insert the SD card!"));
    while(true){                          // Programmabbruch -> while(true) ist Endlosschleife
      delay(0);                           // Code für den Watchdog des ESP8266, nicht relevant
für Arduino UNO
    }
  }

  myDFPlayer.volume(20);                      // Lautstärke auf 10 ( Werte von 0 - 30 ist
möglich)
  for (int i=0; i<16; i++){                   // wiederhole 16 Mal (für jeden Taster)
    pinMode(buttonPins[i], INPUT_PULLUP);     // Taster werden mit internen Pullup-
```

https://starthardware.org/sound-machine-mit-arduino/

```
Widerständen verwendet
  }

}

void loop() {
  for (int i=0; i<16; i++){                    // wiederhole 16 Mal (für jeden Taster)
    if (digitalRead(buttonPins[i]) == LOW){    // wenn ein Taster gedrückt wird
     Serial.println(i);
     myDFPlayer.playMp3Folder(i+1);                        // spiele das MP3 der Nummer des
Tasters nach von der SD-Karte
     delay(200);
    }
  }
  delay(20);

  /* Alles ab hier dient dem Debugging und kommt aus den Beispiel-Dateien der Library */
  if (myDFPlayer.available()) {
    printDetail(myDFPlayer.readType(), myDFPlayer.read()); // Print the detail message from
DFPlayer to handle different errors and states.
  }
}

void printDetail(uint8_t type, int value){

  switch (type) {
    case TimeOut:
      Serial.println(F("Time Out!"));
      break;
    case WrongStack:
      Serial.println(F("Stack Wrong!"));
      break;
    case DFPlayerCardInserted:
      Serial.println(F("Card Inserted!"));
      break;
    case DFPlayerCardRemoved:
      Serial.println(F("Card Removed!"));
      break;
    case DFPlayerCardOnline:
      Serial.println(F("Card Online!"));
      break;
    case DFPlayerUSBInserted:
      Serial.println("USB Inserted!");
      break;
    case DFPlayerUSBRemoved:
      Serial.println("USB Removed!");
      break;
    case DFPlayerPlayFinished:
      Serial.print(F("Number:"));
      Serial.print(value);
      Serial.println(F(" Play Finished!"));
      break;
    case DFPlayerError:
      Serial.print(F("DFPlayerError:"));
      switch (value) {
        case Busy:
          Serial.println(F("Card not found"));
          break;
        case Sleeping:
          Serial.println(F("Sleeping"));
          break;
```

```
        case SerialWrongStack:
          Serial.println(F("Get Wrong Stack"));
          break;
        case CheckSumNotMatch:
          Serial.println(F("Check Sum Not Match"));
          break;
        case FileIndexOut:
          Serial.println(F("File Index Out of Bound"));
          break;
        case FileMismatch:
          Serial.println(F("Cannot Find File"));
          break;
        case Advertise:
          Serial.println(F("In Advertise"));
          break;
        default:
          break;
      }
      break;
    default:
      break;
  }

}
```