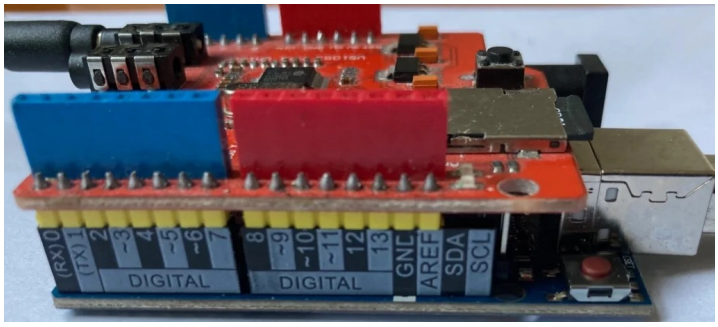
Inhaltsverzeichnis

Ziel des Projekts

mp3-Dateien sollen von einer SD-Karte abgespielt werden. Du benötigst eine Micro-SD-Karte, die mit FAT32 formatiert wurde.
Kopiere mehrere mp3-Dateien auf die SD-Karte.
Du musst die Namen nach folgendem Muster ändern:
track001.mp3, track002.mp3 ...
Der Player soll mit einem Tastenfeld mit vier Tasten oder mit einer Fernbedienung gesteuert werden.

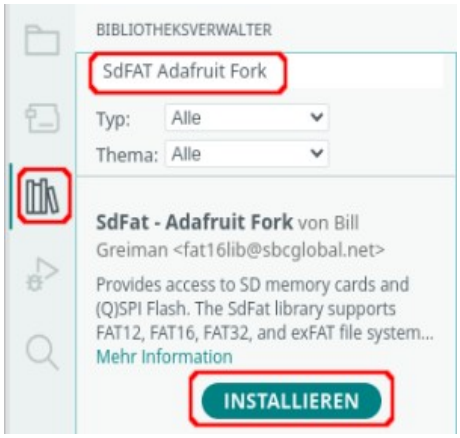
Die Hardware

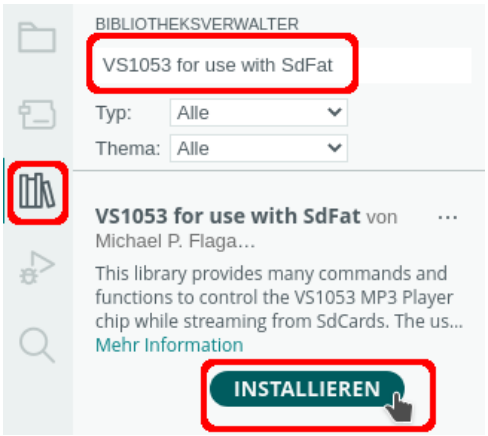


Ein Shield ist ein Bauteil, das auf einen Arduino UNO aufgesteckt wird. Das mp3-Shield VS1053 verfügt über u. a. über einen Decoder für mp3 und einen Steckplatz für Mini-SD-Karten. Es werden zwar alle digitalen und analogen Anschlüsse herausgeführt, sie sind aber nur sehr eingeschränkt nutzbar.

Das Shield verwendet den SPI-Bus und auch noch andere digitale Pins, die für die Ansteuerung des Shields benötigt werden. Daher sind nur die digitalen Pins 5 und 10 uneingeschränkt nutzbar.

Benötigte Bibliotheken:



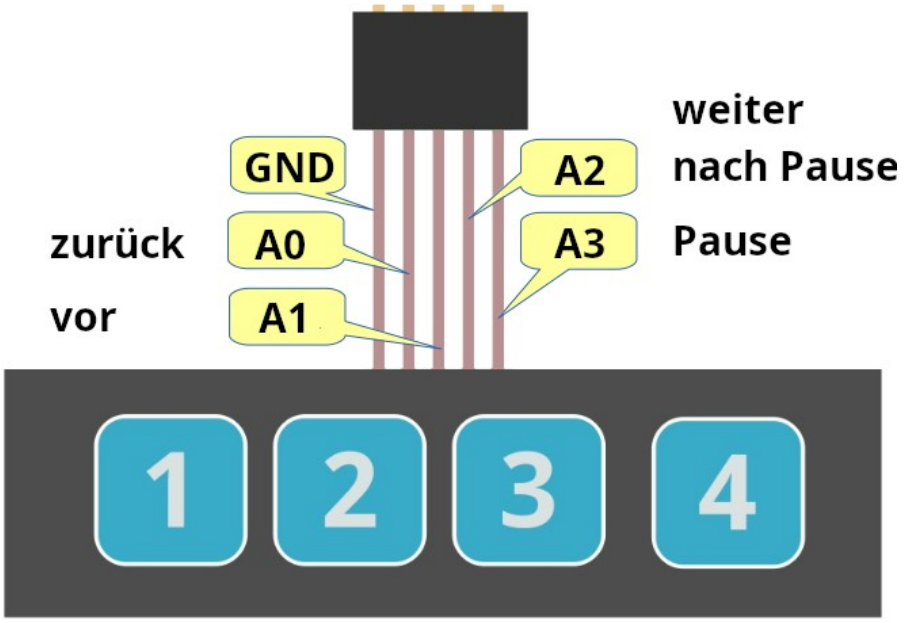


Funktionen der Bibliothek vs1503

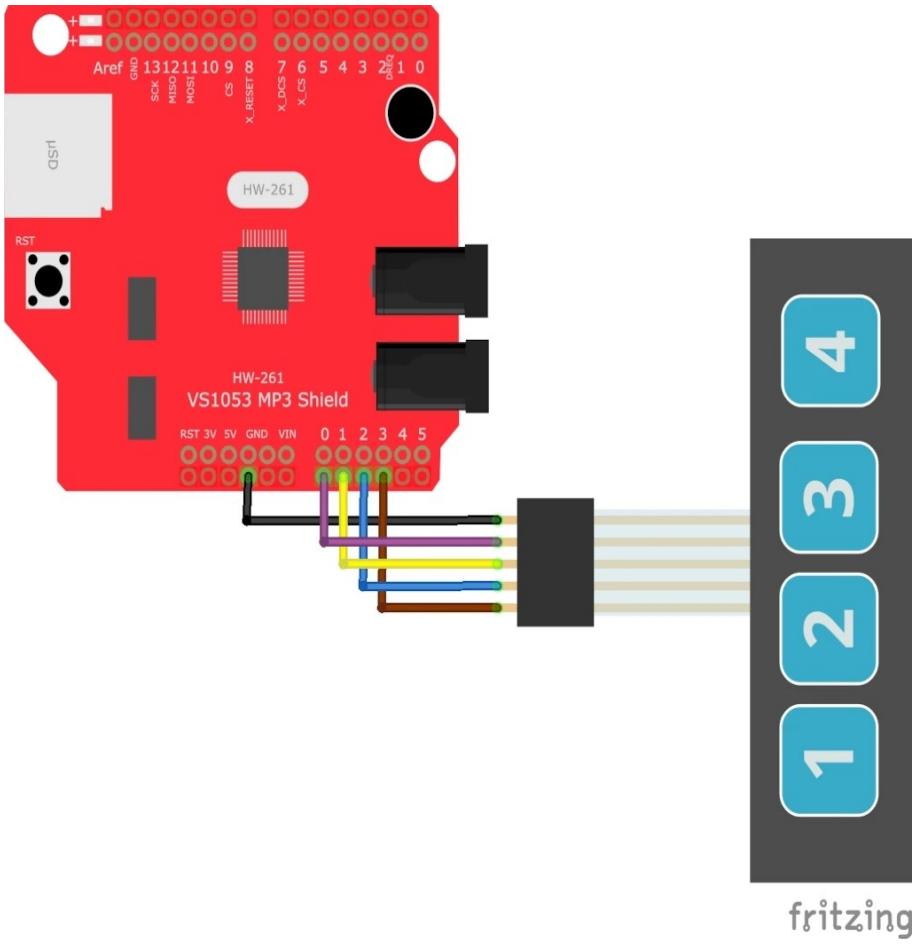
Schlüsselwort	**Aktion**
begin()	Player starten
getState()	Status des Players abfragen 0 = der Player wurde nicht gestartet 1 = der Start des Players war erfolgreich
setVolume(links, rechts)	Lautstärke setzen -> links, rechts 1, 1 sehr laut je größer der Wert, desto leiser
playTrack(Nummer) track001.mp3 → playTrack(1) track002.mp3 → playTrack(2)	spielt den Track (Nummer) Track darf eine wav oder mp3-Datei sein
playMP3(Dateiname) track001.mp3 → playMP3(track001.mp3) track002.mp3 → playMP3(track002.mp3)	im Unterschied zu playTrack() dürfen Dateinamen angegeben werden
stopTrack()	stoppt den gerade laufenden Track
pauseMusic()	pausiert den gerade laufenden Track
resumeMusic()	setzt die Wiedergabe nach der Pause fort
isPlaying()	Stellt fest, ob ein Titel abgespielt wird
setBassAmplitude(Wert);	Bässe einstellen: erlaubte Werte: 0 bis 15
setTrebleAmplitude(Wert)	Höhen einstellen: erlaubte Werte: -8 bis 7
SendSingleMIDInote()	spielt ein „Beep“

Player mit Tastenfeld

Das Tastenfeld



Der Schaltplan



Das Programm

Bibliotheken und Variable

Im Kopf des Programms werden die benötigten Bibliotheken eingebunden und die Variablen definiert.

```
#include "vs1053_SdFat.h"
#include "Adafruit_Debounce.h"

// Bezeichnung der SD-Karte
SdFat sd;

// Bezeichnung des mp3-Shields
vs1053 MP3player;

// Variable für das Lesen des Verzeichnisses
File Verzeichnis;
File Datei;

// die Taster
int TASTER1 = A0; // zurück
int TASTER2 = A1; // weiter
int TASTER3 = A2; // weiter nach Pause
int TASTER4 = A3; // Pause

// "Prallverhinderer" für die Tasten starten
Adafruit_Debounce Taste_zurueck(TASTER1, LOW);
Adafruit_Debounce Taste_naester(TASTER2, LOW);
Adafruit_Debounce Taste_weiter(TASTER3, LOW);
Adafruit_Debounce Taste_Pause(TASTER4, LOW);

// Tracknummer/Anzahl der Tracks
int Track = 1;
int TrackMax = 0;
```

Der setup-Teil

```
void setup()
{
  // "Prellverhinderer" starten
  Taste_zurueck.begin();
  Taste_naester.begin();
  Taste_weiter.begin();
  Taste_Pause.begin();

  // Seriellen Monitor starten
  Serial.begin(9600);

  // auf Seriellen Monitor warten
  while (!Serial);

  Serial.println();
```

```
  // SD Karte starten
  sd.begin(SD_SEL, SPI_FULL_SPEED);

  // Anzahl der Tracks im Wurzelverzeichnis zählen
  // Char-Array für den Dateinamen
  char Dateiname[13];
  if (!sd.chdir("/")) sd.errorHalt("keine SD-Karte vorhanden");

  Verzeichnis.open("/");
  Serial.println("Dateiname    Größe");
  Serial.println("-----------------------------");

  while (Datei.openNext(&Verzeichnis, O_READ))
  {
    Datei.getName(Dateiname, sizeof(Dateiname));

    // handelt es sich um eine Musikdatei (isFnMusic)
    if (isFnMusic(Dateiname) )
    {
      Serial.print(Dateiname);

      // Dateigröße ermitteln, in MB umwandeln, Punkt durch Komma ersetzen
      float DateiGroesse = Datei.fileSize();
      String Groesse = String(DateiGroesse / 1000000);
      Groesse.replace(".", ",");
      Serial.println("\t" + Groesse + " MB");
      TrackMax ++;
    }
    Datei.close();
  }

  Serial.println();
  Serial.println("Anzahl der Tracks: " + String(TrackMax));
  Serial.println("-----------------------------");

  // Player starten
  MP3player.begin();

  // Höhen: erlaubte Werte: -8 bis 7
  MP3player.setTrebleAmplitude(4);

  // Bässe: erlaubte Werte 0 bis 15
  MP3player.setBassAmplitude(7);

  // Status des Players ermitteln
  if (MP3player.getState() == 1) Serial.println("Player erfolgreich gestartet");

  // Lautstärke setzen -> links, rechts -> 1, 1 sehr laut
  // je größer die Werte desto leiser
  MP3player.setVolume(50, 50);

  // 1. Track spielen
  Serial.println("Spiele Track " + String(Track));
  MP3player.playTrack(Track);
}
```

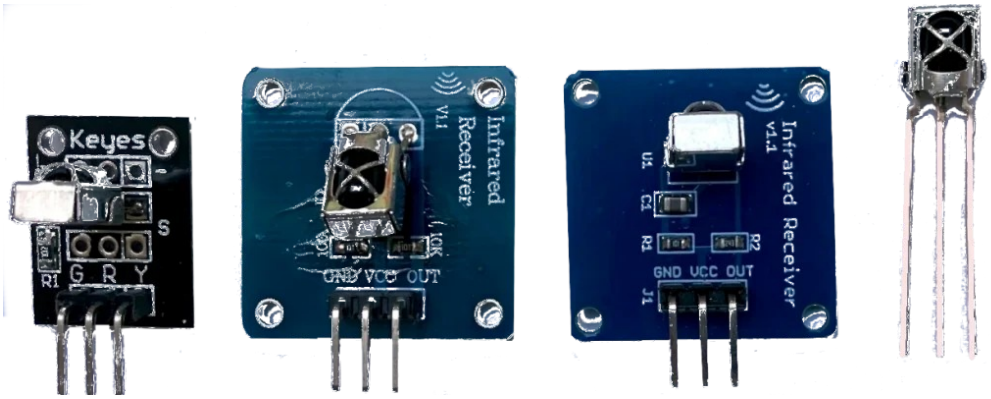
Der loop-Teil

```
void loop()
{
  // testen, ob eine Taste gedrückt wurde
  Taste_zurueck.update();
  Taste_naester.update();
  Taste_weiter.update();
  Taste_Pause.update();

  // vorherigen Track abspielen
  if (Taste_zurueck.justPressed())
  {
    vorherigerTitel();
  }

  // nächster Track
  if (Taste_naester.justPressed())
  {
    naechsterTitel();
  }

  // Pause
  if (Taste_weiter.justPressed())
  {
    Pause();
  }

  // weiter abspielen
  if (Taste_Pause.justPressed())
  {
    Weiter();
  }

  // wenn der aktuelle Titel beendet ist
  if (!MP3player.isPlaying())
  {
    if (Track < TrackMax) Track ++;
    else Track = 1;
    Serial.println("Spiele Track " + String(Track));
    MP3player.playTrack(Track);
  }
}
```

Funktionen

```
void naechsterTitel()
{
  // kurzes Beep spielen
  MP3player.SendSingleMIDInote();
  MP3player.SendSingleMIDInote();

  // aktuellen Track stoppen
  MP3player.stopTrack();

  // wenn der letzte Track gespielt wurde
  // -> Neustart mit 1
  if (Track < TrackMax) Track ++;
  else Track = 1;
  Serial.println("Spiele Track " + String(Track));
  MP3player.playTrack(Track);
}

void vorherigerTitel()
{
  // kurzes "Beep" spielen
  MP3player.SendSingleMIDInote();
  MP3player.SendSingleMIDInote();

  // laufenden Track stoppen/aktuellen Track abspielen
  MP3player.stopTrack();
  if (Track > 1) Track --;
  Serial.println("Spiele Track " + String(Track));
  MP3player.playTrack(Track);
}

void Pause()
{
  MP3player.SendSingleMIDInote();
  MP3player.SendSingleMIDInote();

  // Track pausieren
  Serial.println("Pause Track " + String(Track));
  MP3player.pauseMusic();
}

void Weiter()
{
  MP3player.SendSingleMIDInote();
  MP3player.SendSingleMIDInote();

  // Wiedergabe fortsetzen
  Serial.println("Weiter Track " + String(Track));
  MP3player.resumeMusic();
}
```

Player mit Fernbedienung



Verschiedene Infrarot-Empfänger mit VS1838B-Modul

Die Verwendung einer Fernbedienung bietet mehr Möglichkeiten, den Player zu steuern:

- Pfeil links: vorheriger Titel
- Pfeil rechts: nächster Titel
- Pfeil unten: Lautstärke um 5 verringern
- Pfeil oben: Lautstärke um 5 erhöhen
- OK: Zufallsgenerator ausschalten

Testprogramm Keyes-Fernbedienung

```
// benötigte Bibliothek einbinden
#include "IRremote.hpp"

// der Pin, an dem der Infrarot-Empfänger angeschlossen ist
int EmpfaengerPin = 5;

void setup()
{
  // Seriellen Monitor starten
  Serial.begin(9600);

  // Infrarot-Empfänger starten
  IrReceiver.begin(EmpfaengerPin);
}

void loop()
{
  // decode() -> Daten lesen
  if (IrReceiver.decode())
  {
    // kurzes delay, damit nur ein Tastendruck gelesen wird

    delay(200);
    // resume -> nächsten Wert lesen
    IrReceiver.resume();
```

```
    /*
      der Empfänger empfängt zwischendurch Signale,
      die nicht ausgewertet werden können
      es sollen dehalb nur die korrekt erkannten Tasten ausgewertet werden
      die Dezimalwerte der korrekten erkannten Tasten liegen zwischen > 0 und < 95
      es wird abgefragt, ob das empfangene Kommando decodedIRData.command
      zwischen 0 und (&&) 95 liegt
    */
    if (IrReceiver.decodedIRData.command > 0 && IrReceiver.decodedIRData.command < 95)
    {
      Serial.print("Dezimalwert: ");

      // IrReceiver.decodedIRData.command = Wert der gedrückten Taste
      Serial.print(IrReceiver.decodedIRData.command);
      Serial.print(" -> ");

      // Werte abfragen und anzeigen
      if (IrReceiver.decodedIRData.command == 22) Serial.println("Taste 1");
      if (IrReceiver.decodedIRData.command == 25) Serial.println("Taste 2");
      if (IrReceiver.decodedIRData.command == 13) Serial.println("Taste 3");
      if (IrReceiver.decodedIRData.command == 12) Serial.println("Taste 4");
      if (IrReceiver.decodedIRData.command == 24) Serial.println("Taste 5");
      if (IrReceiver.decodedIRData.command == 94) Serial.println("Taste 6");
      if (IrReceiver.decodedIRData.command == 8) Serial.println("Taste 7");
      if (IrReceiver.decodedIRData.command == 28) Serial.println("Taste 8");
      if (IrReceiver.decodedIRData.command == 90) Serial.println("Taste 9");
      if (IrReceiver.decodedIRData.command == 82) Serial.println("Taste 0");
      if (IrReceiver.decodedIRData.command == 66) Serial.println("Taste *");
      if (IrReceiver.decodedIRData.command == 74) Serial.println("Taste #");
      if (IrReceiver.decodedIRData.command == 68) Serial.println("Pfeil links");
      if (IrReceiver.decodedIRData.command == 67) Serial.println("Pfeil rechts");
      if (IrReceiver.decodedIRData.command == 70) Serial.println("Pfeil oben");
      if (IrReceiver.decodedIRData.command == 21) Serial.println("Pfeil unten");
      if (IrReceiver.decodedIRData.command == 64) Serial.println("OK");
    }
  }
}
```

Testprogramm beliebige Fernbedienung

```
#include "IRremote.hpp"

int EmpfaengerPin = 5;

void setup()
{
  Serial.begin(9600);

  // Empfänger starten
  IrReceiver.begin(EmpfaengerPin);
}
```

```
void loop()
{
  // Daten lesen
  if (IrReceiver.decode())
  {
    delay(200);

    // resume -> nächsten Wert lesen
    IrReceiver.resume();
    if (IrReceiver.decodedIRData.command > 0)
    {
      /*
        printIRResultMinimal zeigt die verwendete Taste
        P = Protokoll
        C = Kommando in Hex
      */

      Serial.print(F("P = Protokoll C = Taste hexadezimal: "));
      IrReceiver.printIRResultMinimal(&Serial);
      Serial.print(F(" Dezimal: "));
      Serial.println(IrReceiver.decodedIRData.command);
    }
  }
}
```

Das Programm

Bibliotheken und Variable

```
#include "vs1053_SdFat.h"
#include "IRremote.hpp"

// der Pin, an dem der Infrarot-Empfänger angeschlossen ist
int EmpfaengerPin = 4;
// Bezeichnung der SD-Karte
SdFat sd;

// Bezeichnung des mp3-Shields
vs1053 MP3player;

// Variable für das Lesen des Verzeichnisses
File Verzeichnis;
File Datei;

// Tracknummer/Anzahl der Tracks
int Track = 1;
int TrackMax = 0;

int LautstaekeLinks = 50;
int LautstaerkeRechts = 50;

bool zufaelligeWiedergabe = true;
```

Der setup-Teil

```
void setup()
{
  // Infrarot-Empfänger starten
  IrReceiver.begin(EmpfaengerPin);

  // Seriellen Monitor starten
  Serial.begin(9600);

  // auf Seriellen Monitor warten
  while (!Serial);

  Serial.println();

  // SD Karte starten
  sd.begin(SD_SEL, SPI_FULL_SPEED);

  // Anzahl der Tracks im Wurzelverzeichnis zählen
  // Char-Array für den Dateinamen
  char Dateiname[13];
  if (!sd.chdir("/")) sd.errorHalt("keine SD-Karte vorhanden");

  Verzeichnis.open("/");
  Serial.println("Dateiname    Größe");
  Serial.println("-----------------------------");

  while (Datei.openNext(&Verzeichnis, O_READ))
  {
    Datei.getName(Dateiname, sizeof(Dateiname));

    // handelt es sich um eine Musikdatei (isFnMusic)
    if (isFnMusic(Dateiname) )
    {
      Serial.print(Dateiname);

      // Dateigröße ermitteln, in MB umwandeln, Punkt durch Komma ersetzen
      float DateiGroesse = Datei.fileSize();
      String Groesse = String(DateiGroesse / 1000000);
      Groesse.replace(".", ",");
      Serial.println("\t" + Groesse + " MB");
      TrackMax ++;
    }
    Datei.close();
  }

  Serial.println();
  Serial.println("Anzahl der Tracks: " + String(TrackMax));
  Serial.println("-----------------------------");

  // Player starten
  MP3player.begin();

  // Höhen: erlaubte Werte: -8 bis 7
  MP3player.setTrebleAmplitude(4);
```

```
  // Bässe: erlaubte Werte 0 bis 15
  MP3player.setBassAmplitude(7);

  // Status des Players ermitteln
  if (MP3player.getState() == 1) Serial.println("Player erfolgreich gestartet");

  // Lautstärke setzen -> links, rechts -> 1, 1 sehr laut
  // je größer die Werte desto leiser
  MP3player.setVolume(LautstaekeLinks, LautstaerkeRechts);

  randomSeed(millis());
  if (zufaelligeWiedergabe) Track = random(1, TrackMax);
  else Track = 1;

  // 1. Track spielen
  Serial.println("Spiele Track " + String(Track));
  MP3player.playTrack(Track);
}
```

Der loop-Teil

```
void loop()
{
  if (IrReceiver.decode())
  {
    // kurzes delay, damit nur ein Tastendruck gelesen wird

    delay(200);

    // resume -> nächsten Wert lesen
    IrReceiver.resume();

    /*
      der Empfänger empfängt zwischendurch Signale,
      die nicht ausgewertet werden können
      es sollen dehalb nur die korrekt erkannten Tasten ausgewertet werden
      die Dezimalwerte der korrekten erkannten Tasten liegen zwischen > 0 und < 95
      es wird abgefragt, ob das empfangene Kommando decodedIRData.command
      zwischen 0 und (&&) 95 liegt
    */
    if (IrReceiver.decodedIRData.command > 0 && IrReceiver.decodedIRData.command < 95)
    {
      // IrReceiver.decodedIRData.command = Wert der gedrückten Taste
      // Werte abfragen und anzeigen
      if (IrReceiver.decodedIRData.command == 64) ZufallSchalten();
      if (IrReceiver.decodedIRData.command == 66) Weiter();
      if (IrReceiver.decodedIRData.command == 74) Pause();
      if (IrReceiver.decodedIRData.command == 68) vorherigerTitel();
      if (IrReceiver.decodedIRData.command == 67) naechsterTitel();
      if (IrReceiver.decodedIRData.command == 70) Lauter();
      if (IrReceiver.decodedIRData.command == 21) Leiser();
    }
  }
```

```
  // wenn der aktuelle Titel beendet ist
  if (!MP3player.isPlaying())
  {
    if (zufaelligeWiedergabe)
    {
      randomSeed(millis());
      Track = random(1, TrackMax);
    }
    else
    {
      if (Track < TrackMax) Track ++;
      else Track = 1;
    }
    Serial.println("Spiele Track " + String(Track));
    MP3player.playTrack(Track);
  }
}
```

Funktionen

```
void naechsterTitel()
{
  // kurzes Beep spielen
  MP3player.SendSingleMIDInote();
  MP3player.SendSingleMIDInote();

  // aktuellen Track stoppen
  MP3player.stopTrack();

  if (zufaelligeWiedergabe)
  {
    randomSeed(millis());
    Track = random(1, TrackMax);
  }
  else
  {
    // wenn der letzte Track gespielt wurde
    // -> Neustart mit 1
    if (Track < TrackMax) Track ++;
    else Track = 1;
    // Serial.println("Spiele Track " + String(Track));
  }
  Serial.println("Spiele Track " + String(Track));
  MP3player.playTrack(Track);
}

void vorherigerTitel()
{
  // kurzes "Beep" spielen
  MP3player.SendSingleMIDInote();
  MP3player.SendSingleMIDInote();

  MP3player.stopTrack();
```

```
  if (zufaelligeWiedergabe)
  {
    randomSeed(millis());
    Track = random(1, TrackMax);
  }
  else
  {
    // laufenden Track stoppen/aktuellen Track abspielen
    if (Track > 1) Track --;
    else Track = TrackMax;
  }

  Serial.println("Spiele Track " + String(Track));
  MP3player.playTrack(Track);
}

void Pause()
{
  MP3player.SendSingleMIDInote();
  MP3player.SendSingleMIDInote();

  // Track pausieren
  Serial.println("Pause Track " + String(Track));
  MP3player.pauseMusic();
}

void Weiter()
{
  MP3player.SendSingleMIDInote();
  MP3player.SendSingleMIDInote();

  // Wiedergabe fortsetzen
  Serial.println("Weiter Track " + String(Track));
  MP3player.resumeMusic();
}

void Lauter()
{
  Serial.println(String(LautstaekeLinks) + "," + String(LautstaerkeRechts));
  if (LautstaekeLinks >= 0) LautstaekeLinks -= 5;
  if (LautstaerkeRechts >=0 ) LautstaerkeRechts -= 5;
  MP3player.setVolume(LautstaekeLinks, LautstaerkeRechts);
}

void Leiser()
{
  Serial.println(String(LautstaekeLinks) + "," + String(LautstaerkeRechts));
  if (LautstaekeLinks < 100) LautstaekeLinks += 5;
  if (LautstaerkeRechts < 100) LautstaerkeRechts += 5;
  MP3player.setVolume(LautstaekeLinks, LautstaerkeRechts);
}
```

```
void ZufallSchalten()
{
  zufaelligeWiedergabe = !zufaelligeWiedergabe;
  if (zufaelligeWiedergabe) Serial.println("Zufall ein");
  else Serial.println("Zufall aus");
}
```

Hartmut Waller letzte Änderung: 30.11.25