ESP32-Audio-Kit – Jukebox

- Hardware & Software -

Spezifikation der Funktionalität

1) Wifi-Manager
   Der Wifi-Manger ermöglicht die Verbindung zu einem vorhandenen Wifi-AP.
   Wenn keine Verbindung hergestellt werden kann wird ein AP (IP: 192.168.4.1) bereitgestellt über den die Verbindung konfiguriert werden kann.
   Die Credentials werden im internen Speicher abgelegt.
2) Web-Server
   Ein einfacher Web-Sever ermöglicht die weitere Konfiguration.
3) Audio-Player
   Der Audio-Player spielt Musik ab. Die Auswahl des abzuspielenden Stücks wird über die Tasten der Joukebox gesteuert.
   Die Lautstärke wird über den Web-Server oder über ein Podi gesetuert.
4)
5)
6)
7)

Ablage der File auf der SD Karte:

Matrix aus Tasten 1-10 und a-k (ACHTUNG: i wird nicht genutzt)

	a	b	c	d	...
1	1a	1b	1c	...	
2	2a	2b	...		
3	3a	...			
4	...				
5					
…					

Das sind maximal 100 Files.
Bei 4 MB pro Datei wird ein Speicher von mind. 400 MB benötigt (z.B. 8GB SD Karte).

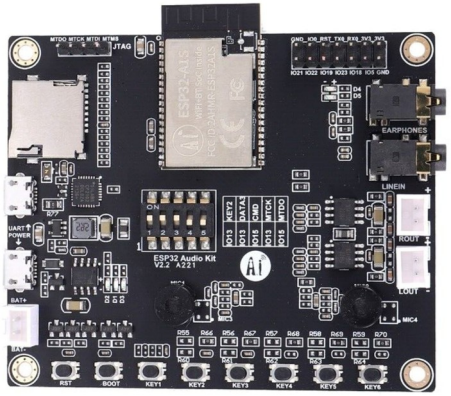
ESP32-Audio-Kit – Jukebox

- Hardware & Software -

Hardware

- ESP32 Audio Kid V2.2 A149 (20€)
- Jukebox M100K Stereo von Harting (ist da)
- Verstärker/Endstufe 2x20W (80-100€)
- Evt. Lautsprecher 2x20X (40-60€)

Das Audio Kit hat eine Leistung von 2x3W. Das dürfte für den Anwendungszweck zu wenig sein. Daher benötigen wir noch eine Endstufe. Wir müssen prüfen ob Komponenten aus der Jukebox noch verwendet werden können (z.B. Lautsprecher, Endstufe, Stromversorgung).
Evtl kann ein kleiner LCD-Display eingebaut werden, um darauf den Zustand anzuzeigen (z.B. Welches Stück gerade gespielt wird, Dauer, etc.).



Entwicklungsumgebung:

- PlattformIO
- Windows 10

Aufwand:

- ca. 3-5 PT Programmentwicklung
- ca. 1 PT Einbau der Elektronik

Software-Module:

Wifi-Manager:

- https://github.com/tzapu/WiFiManager.git
- Check https://dronebotworkshop.com/wifimanager/ how to use

Web-Server:

- framework-arduinoespressif

Audio:

- https://github.com/pschatzmann/arduino-audio-tools.git
- https://github.com/pschatzmann/arduino-audiokit.git
- https://github.com/pschatzmann/arduino-libhelix.git
- https://github.com/pschatzmann/arduino-libmad.git
- https://github.com/greiman/SdFat.git

Weitere Quellen und Anregungen

Espressif

- https://github.com/espressif/arduino-esp32.git

ESP32-Sonos-Jukebox

- https://github.com/Frank-Bemelman/ESP32-Sonos-Jukebox-V3.git

ESP32-Audio-Kit – Jukebox

- Hardware & Software -

Spezifikation der Funktionalität – SD Karte

Ablage der File auf der SD Karte:

Matrix aus Tasten 1-10 und a-j

	a	b	c	d	...
1	1a	1b	1c	...	
2	2a	2b	...		
3	3a	...			
4	...				
5					
…					

Das sind maximal 100 Files.
Bei 4 MB pro Datei wird ein Speicher von mind. 400 MB benötigt.

Struktur auf der SD Karte:
Damit die Files nicht umständlich umbenannt werden müssen biete sich die Ablage in eine geeigneten Struktur an.

/a1/<filename>.mp3
/a2/<filename>.mp3
…
/j10/<filename>.mp3

Alternative:

/a/1/<filename>.mp3
  /2/<filename>.mp3
...
/j/9/<filename>.mp3
  /10/<filename>.mp3

Die Files können über einen Computer auf die SD Karte geschrieben werden.

Upload-Funktion über den internen Web-Server.

ESP32-Audio-Kit – Jukebox

- Hardware & Software -

Spezifikation der Funktionalität – Auswahl Musik

Auswahl der Musikstücke über Buttons der Jukebox:

Auf Basis der Arbeit von Frank-Bemelman kann die Auswahl über die Schaltung von Widerständen als Spannungsteiler und das Auslesen der Resultierenden Spannung erfolgen. Er hat eine eine Funktion (int AdcConvert(int value)) genaut, die genau dies macht. Die Schaltung sieht dann wie folgt aus:

```
//
//                     no
//                   resistor
// +3V3 ----[560]---+  here   +--[4K7]--+--[2K2]--+--[1K5]--+--[820]--+--[560]--+--[390]--+--[220]--+--[150]--+---[68]--+
//                  |         |         |         |         |         |         |         |         |         |         |
//     NC CONTACTS  +----=----+----=----+----=----+----=----+----=----+----=----+----=----+----=----+----=----+----=----+
//     KEY    ROW   |   10         9         8         7         6         5         4         3         2         1    |
//                  |                                                                                                  |
//                  +--------> to analog input pin ESP32 D36                                                           +--------> to GND
//

//
//                     no
//                   resistor
// +3V3 ----[560]---+  here   +--[4K7]--+--[2K2]--+--[1K5]--+--[820]--+--[560]--+--[390]--+--[220]--+--[150]--+---[68]--+
//                  |         |         |         |         |         |         |         |         |         |         |
//     NC CONTACTS  +----=----+----=----+----=----+----=----+----=----+----=----+----=----+----=----+----=----+----=----+
//     KEY    ROW   |    K         J         H         G         F         E         D         C         B         A    |
//                  |                                                                                                  |
//                  +--------> to analog input pin ESP32 D39                                                           +--------> to GND
//

if(value>3864)return 10;
if(value>3364)return 9;
if(value>2846)return 8;
if(value>2421)return 7;
if(value>2051)return 6;
if(value>1673)return 5;
if(value>1227)return 4;
if(value>815)return 3;
if(value>459)return 2;
if(value>127)return 1;
return 0; // no key pressed, all closed
```

Gemessen wir die Spannung an einem analog Input PIN (D36). Die Widerstände müssen so gewählt sein, dass sich ein eindeutiger Zustand ergibt (Spannung muss klar abgegrenzt sein).

ESP32-Audio-Kit – Jukebox

- Hardware & Software -

Spezifikation der inneren Architektur

ESP32 als Basis mit 2 Core Prozessor:

Setup:

- ***Start-FS***
- ***Wifi-Manager***
- ***xTaskCreatePinnedToCore()***

Core 0:

- ***Jukebox***
  - AudioPlayer
  - SD (Musik-Speicher)

Core 1:

- ***Admin Interface***
  - WebServer
    - SPIFF (interner Speicher)
    - SD (Musik-Speicher)
  - Main-Page
    - Login/Logout
    - File-Manager
    - Reboot

```
/**
 * @brief htaskAudioPlayerCode()
 *
 * @param pvParameters
 */
void taskAudioPlayerCode( void * pvParameters ){
    dbSerialPrint("hTaskWebServer running on core "); dbSerialPrintln(xPortGetCoreID());
    /* ------------------------------------------------- */
    /* start AudioPlayer                                 */
    AudioPlayer()->begin();
    /* ------------------------------------------------- */
    /* catch upload server events                        */
    while(true)
    {
        AudioPlayer()->handleInput();
        vTaskDelay(10/portTICK_PERIOD_MS);
    }
}
```

```
/**
 * @brief htaskWebServerCode()
 *
 * @param pvParameters
 */
void taskWebServerCode( void * pvParameters ){
    dbSerialPrint("hTaskWebServer running on core "); dbSerialPrintln(xPortGetCoreID());
    /* ------------------------------------------------- */
    /* switch webserver on                               */
    WebServer()->on( "/", HTTP_POST, handleSuccess, handleFileUpload);
    /* ------------------------------------------------- */
    /* receive fileSize once a file is selected          */
    WebServer()->on( "/fs", HTTP_POST, [](){fileSize = WebServer()->arg(F("fileSize")).toInt();
                                            WebServer()->send(200, F("text/plain"), "");});
    /* ------------------------------------------------- */
    /* called when the url is not defined here use it to  */
    /* load content from SPIFFS                          */
    WebServer()→onNotFound([](){ if( !handleFileRead( WebServer()->uri() ) )
                                {WebServer()->send(404, F("text/plain"), F("FileNotFound"));}});
    /* ------------------------------------------------- */
    /* start HTTP server                                 */
    WebServer()->begin();
    /* ------------------------------------------------- */
    /* catch upload server events                        */
    while(true)
    {
        WebServer()->handleClient();
        vTaskDelay(10/portTICK_PERIOD_MS);
    }
}
```

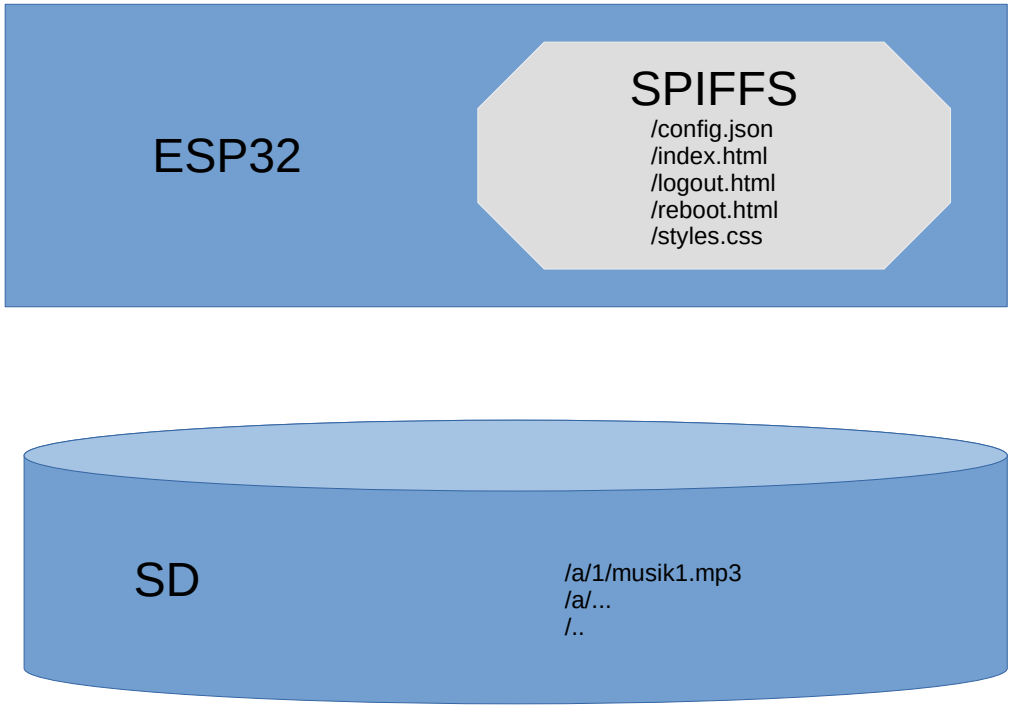
ESP32-Audio-Kit – Jukebox

- Hardware & Software -

Spezifikation der inneren Architektur

Admin Interface

- WebServer
  - SPIFFS (interner Speicher)
  - SD (Musik-Speicher)
- Login mit User, Passwort
  - Logout
- File-Manager
  - list
  - mkdir, rmdir
  - upload
  - delete
- Reboot



ESP32-Audio-Kit – Jukebox

- Hardware & Software -

Spezifikation der Admin-UI

Admin UI

- single site
- Verwaltung Speicher
  - Ordner anlegen/löschen
  - Files hochladen/löschen/abspielen

Musikspeicher, auf der SD-Karte (SD)
Programmspeicher, im internen Speicher (SPIFFS)

Anzeige:

- Releaseinfo
- Speicher
  - Art, frei, belegt, total