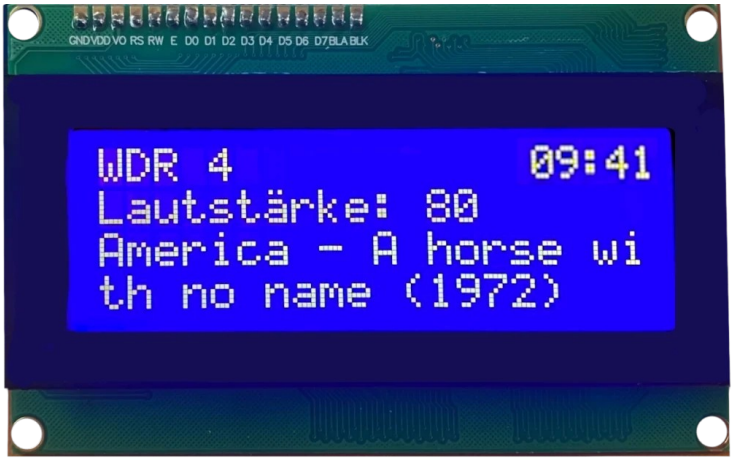
Ziel des Projekts

Ein ESP32 soll zusammen mit einem mp3-Decoder als Internetradio funktionieren. Mit einer Fernbedienung kann der Sender gewechselt und der Ton leiser oder lauter eingestellt werden. Der ESP32 sucht sich selbstständig das WiFi-Netz mit der größten Signalstärke, dadurch kann das Radio in WiFi-Netzen mit Repeatern eingesetzt werden ohne dass WiFi-Daten neu eingegeben werden müssen. Natürlich kann auch ein die Wohnung umspannendes einzelnes WiFi-Netz verwendet werden.

Die Tasten der Fernbedienung:

- Pfeil nach rechts: nächster Sender
- Pfeil nach links: vorheriger Sender
- Pfeil nach unten: leiser
- Pfeil nach oben: lauter
- #: Ton aus, erneute Betätigung: Ton an
- Zifferntasten: Sender 1 – 10 (0 = 10)
- *: Hintergrundbeleuchtung ein-/ausschalten
- OK: Titel ausblenden (wird erst bei der nächsten Info aktiviert)

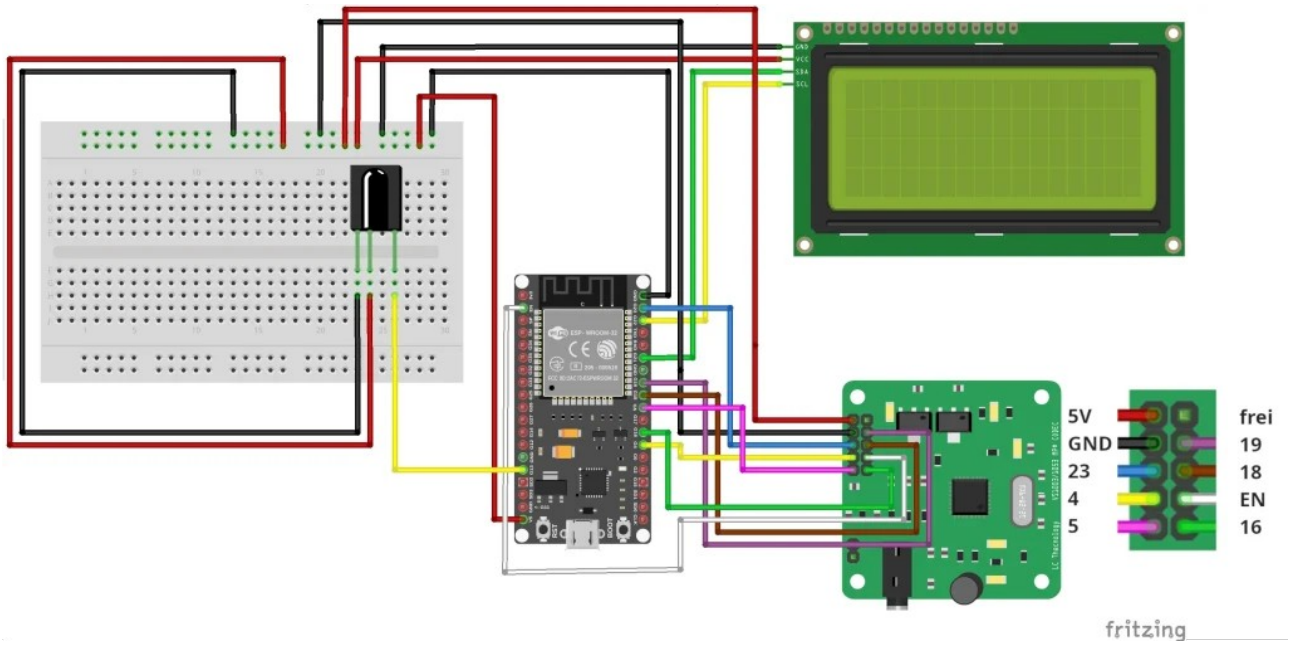
Auf dem LCD werden Sender, Lautstärke, Uhrzeit und ein Teil der vom Sender übermittelten Informationen dargestellt. Das sind der Titel des laufenden Songs, Infos zum Sender oder zu aktuellen Ereignissen und häufig ein kurzer Wetterbericht. Allerdings überschreiten diese Information oft die zur Verfügung stehenden 40 Zeichen. Daher werden sie abgeschnitten, ein korrekter Zeilenumbruch findet auch nicht statt.



Benötigte Bauteile

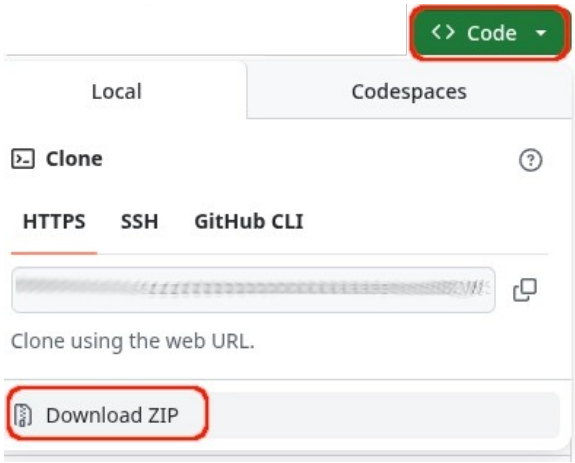
- mp3-Decoder VS1053
- 4-zeiliges LCD mit I²C-Schnittstelle
- Leitungsdrähte
- Infrarot-Modul und beliebige Fernbedienung

Der Schaltplan



Benötigte Bibliotheken

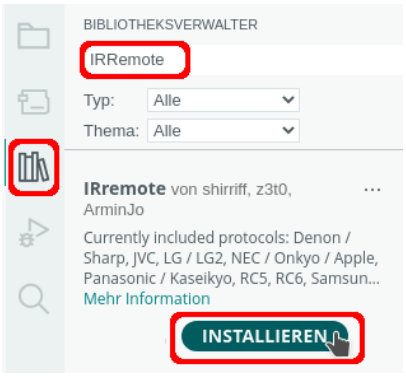
Bibliothek ↳ESP_VS1053_Library
Bibliothek ↳ESP32_VS1053_Stream



Beide Bibliotheken müssen als zip-Datei herunter geladen werden. Klicke auf der Webseite auf den grünen Button und füge die Bibliothek mit **Sketch -> zip-Bibliothek hinzu**..







Minimalistischer Player

```
#include "WiFi.h"
#include "VS1053.h"
#include "ESP32_VS1053_Stream.h"

#define VS1053_CS 5
#define VS1053_DCS 16
#define VS1053_DREQ 4

ESP32_VS1053_Stream stream;

char Router[] = "Router_SSID";
char Passwort[] = "xxxxxxxx";

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

  WiFi.begin(Router, Passwort);

  Serial.println("------------------------");

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(200);
    Serial.print(".");
  }
```

```
  // Schlaf-Modus verhindern
  WiFi.setSleep(false);

  Serial.println("WiFi verbunden...");

  // SPI starten
  SPI.setHwCs(true);
  SPI.begin();

  // Decoder starten
  // bei Misserfolg Programm beenden
  if (!stream.startDecoder(VS1053_CS, VS1053_DCS, VS1053_DREQ) || !stream.isChipConnected())
  {
    Serial.println("Decoder läuft nicht - Programm beendet");
    while (1);
  }
  else Serial.println("Decoder gestartet ...");

  // Lautstärke
  stream.setVolume(80);

  // Sender WDR4 http und https
  // stream.connectToHost("http://wdr-wdr4-live.icecast.wdr.de/wdr/wdr4/live/mp3/128/stream.mp3");
  stream.connectToHost("https://wdr-wdr4-live.icecastssl.wdr.de/wdr/wdr4/live/mp3/128/stream.mp3");

  // Metadaten des Senders
  stream.setInfoCB(infoCallback);
}

void loop()
{
  stream.loop();
  delay(5);
}

// wenn Metadaten vorhanden sind
void infoCallback(const char *info)
{
  Serial.println(info);
}
```

Mit Fernbedienung

Der Programmcode berücksichtigt diese beiden gebräuchlichen Fernbedienungen.





- ab Zeile 159 werden die verfügbaren WiFi-Netze als Werte-Paar definiert:
  SSID des Routers/Repeaters, Passwort
  WiFiNetzwerke.addAP("Router_SSID", "xxxxxxxx");
  WiFiNetzwerke.addAP("Repeater_1", "xxxxxxxx");
- das Programm versucht eine WiFi-Verbindung mit dem WiFi-Netz mit der größten Signalstärke (RSSI) aufzubauen
  gelingt das nicht, wird das Programm mit einem entsprechenden Hinweis beendet
- die Signalstärke des WiFi-Netzwerks entscheidet über einen störungsfreien Radioempfang bei einer eingeschränkten Signalstärke wird eine Warnung eingeblendet
- Wenn du die Senderliste erweitern willst, musst du die Stream-Adresse des neuen Senders an beliebiger Position im Array Sender hinzufügen. Die Liste mit den Beschreibungen musst du mit dem Namen des neuen Senders an der gleichen Position erweitern.
- kann der Decoder nicht gestartet werden, wird das Programm mit einem entsprechenden Hinweis beendet
- war der Start erfolgreich, werden verschiedene Statusmeldungen eingeblendet





Beim Kompileren wird eine Warnmeldung angezeigt:

```
INFO: No definition for LED_BUILTIN found -> default LED feedback is disabled.
```

Der ESP32-Wroom besitzt keine eingebaute LED, du kannst die Warnung ignorieren.

```
// Quellen der Bibliotheken
// ESP32_VS1053_Stream:
// https://github.com/CelliesProjects/ESP32_VS1053_Stream
// VS1053
// https://github.com/baldram/ESP_VS1053_Library
#include "WiFi.h"
#include "VS1053.h"
#include "ESP32_VS1053_Stream.h"
#include "LCDIC2.h"
#include "IRremote.hpp"
#include "time.h"
#include "WiFiMulti.h"

WiFiMulti WiFiNetzwerke;

int EmpfaengerPin = 13;
```

```
// 4-zeiliges LCD
LCDIC2 lcd(0x27, 20, 4);

// Pins des VS1053
#define VS1053_CS 5
#define VS1053_DCS 16
#define VS1053_DREQ 4

// NTP-Server aus dem Pool
#define Zeitserver "de.pool.ntp.org"

#define Zeitzone "CET-1CEST,M3.5.0/02,M10.5.0/03"

// time_t enthält die Anzahl der Sekunden seit dem 1.1.1970 0 Uhr
time_t aktuelleZeit;

tm Zeit;

// Variablen Sender
int SenderIndex = 0;
int AnzahlSender = 0;

int Lautstaerke = 80;
int GesicherteLautstaerke = Lautstaerke;
bool Beleuchtung = true;
bool Titelanzeigen = true;
int AnzeigeZeit = true;

ESP32_VS1053_Stream stream;

/*
  rtone[4] = {toneha, tonehf, tonela, tonelf}
  toneha ( 0 - 15) -> Einstellung der Höhenverstärkung (0 = aus, in Schritten von 1,5 dB)
  tonehf ( 0 - 15) -> Einstellung der unteren Grenze der Höhenfrequenz x 1000 Hz
  tonela ( 0 - 15) -> Einstellung der Bassverstärkung (0 = aus, in Schritten von 1 dB)
  tonelf ( 0 - 15) -> Einstellung der unteren Grenze der Bassfrequenz x 10 Hz
*/
uint8_t rtone[4]  = {15, 15, 5, 5};

const char* Sender[] =
{
  "http://wdr-1live-live.icecast.wdr.de/wdr/1live/live/mp3/128/stream.mp3",  // 1Live
  "http://wdr-wdr2-bergischesland.icecast.wdr.de/wdr/wdr2/bergischesland/mp3/128/stream.mp3",// WDR2
  "http://wdr-wdr4-live.icecast.wdr.de/wdr/wdr4/live/mp3/128/stream.mp3", // WDR4
  "http://wdr-wdr5-live.icecast.wdr.de/wdr/wdr5/live/mp3/128/stream.mp3", // WDR 5
  "http://wdr-cosmo-trap.icecast.wdr.de/wdr/cosmo/trap/mp3/128/stream.mp3", // WDR COSMO
  "http://dispatcher.rndfnk.com/swr/swr1/bw/mp3/128/stream.mp3", // SWR 1
  "http://liveradio.swr.de/sw282p3/swr3/play.mp3", // SWR3
  "http://dispatcher.rndfnk.com/swr/swr4/ko/mp3/128/stream.mp3", // SWR4
  "http://avw.mdr.de/streams/284321-1_mp3_high.m3u", // MDR JUMP Rock channel
  "http://dispatcher.rndfnk.com/hr/hr1/rheinmain/high", // HR1
  "http://icecast.ndr.de/ndr/ndr1niedersachsen/braunschweig/mp3/128/stream.mp3", // NDR 1
  "http://icecast.ndr.de/ndr/ndr2/schleswigholstein/mp3/128/stream.mp3", // NDR 2
  "http://dispatcher.rndfnk.com/rbb/rbb888/live/mp3/mid", // RBB 88,8
  "http://liveradio.sr.de/sr/sr3/mp3/128/stream.mp3", // SR 3
  "http://icecast.ndr.de/ndr/njoy/live/mp3/128/stream.mp3", // N-Joy
  "http://st01.sslstream.dlf.de/dlf/01/128/mp3/stream.mp3?aggregator=web", // Deutschlandfunk
  "http://antennebrandenburg.de/livemp3", // Antenne Brandenburg
  "http://stream.antenne.de/antenne/stream/mp3", // Antenne Bayern live
  "http://stream.antenne.de/antenne-bayern-70er-rock/stream/mp3", // Antenne Bayern 70er Rock
```

```
  "http://stream.antenne.de/classic-rock-live/stream/mp3", // Antenne Bayern Classic Rock
  "http://stream.antenne.de/antenne-bayern-80er-rock/stream/mp3", // Antenne Bayern 80er Rock
  "http://stream.antenne.de/90er-hits/stream/mp3", // Antenne Bayern 90er
  "http://www.charivari.de/webradio/r8082.m3u", // Radio Charivari München
  "http://stream.rpr1.de/webradio/mp3-128/" // RPR1
};

const char *Beschreibung[]
{
  "1Live", "WDR 2", "WDR 4", "WDR 5", "WDR COSMO", "SWR 1", "SWR 3", "SWR 4", "MDR JUMP",
  "HR 1", "NDR 1", "NDR 2", "RBB 88,8",
  "SR 3", "N-Joy", "Deutschlandfunk", "Antenne Brandenburg", "Antenne Bayern live",
  "Antenme Bayern 70er", "Antenne Bayern Rock", "Antenne Bayern 80er",
  "Antenne Bayern 90er",
  "Radio ChariVari", "RPR1"
};

// Tasten Fernbedienung
/*
#define Lautstaerke_plus 70
#define Lautstaerke_minus 21
#define Sender_vor 67
#define Sender_zurueck 68
#define Sender_1 22
#define Sender_2 25
#define Sender_3 13
#define Sender_4 12
#define Sender_5 24
#define Sender_6 94
#define Sender_7 8
#define Sender_8 28
#define Sender_9 90
#define Sender_0 82
#define Ton_aus 74
#define Beleuchtung_ein_aus 66
#define TitelAnzeigeUmschalten 64
*/

// alternative Fernbedienung
#define Lautstaerke_plus 24
#define Lautstaerke_minus 82
#define Sender_vor 90
#define Sender_zurueck 8
#define Sender_1 69
#define Sender_2 70
#define Sender_3 71
#define Sender_4 68
#define Sender_5 64
#define Sender_6 67
#define Sender_7 7
#define Sender_8 21
#define Sender_9 9
#define Sender_0 25
#define Ton_aus 13
#define Beleuchtung_ein_aus 22
#define TitelAnzeigeUmschalten 28
```

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

  // Bluetooth ausschalten
  btStop();

  // LCD starten
  lcd.begin();

  // Cursor "verstecken"
  lcd.setCursor(false);

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

  // auf Seriellen Monitor warten
  while (!Serial)
  {
    delay(10);
  }

  lcd.setCursor(0, 0);
  lcd.print("Wifi starten ...");

  // verfügbare WiFi-Netzwerke
  WiFiNetzwerke.addAP("Repeater-Buero", "xxxxxxxx");
  WiFiNetzwerke.addAP("Repeater-oben", "xxxxxxxx");
  WiFiNetzwerke.addAP("Repeater-Wohnzimmer", "xxxxxxx");

  WiFi.mode(WIFI_STA);

  if (WiFiNetzwerke.run() == WL_CONNECTED)
  {
    Serial.println("WiFi verbunden ...");
  }

  delay(2000);

  Serial.println(WiFi.SSID());
  Serial.println("IP Adresse: ");
  Serial.println(WiFi.localIP());
  Serial.println("RSSI: " + String(WiFi.RSSI()));

  WiFi.setSleep(false);
  WiFi.setAutoReconnect(true);

  if(WiFi.RSSI() == 0)
  {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Keine");
    lcd.setCursor(0, 1);
    lcd.print("WiFi-Verbindung...");
    while(1);
  }
```

```
if(WiFi.RSSI() <-70)
{
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("RSSI: " + String(WiFi.RSSI()));
  lcd.setCursor(0, 1);
  lcd.print("schwaches WiFi");
  lcd.setCursor(0, 2);
  lcd.print("eingeschr\xe1nkter");
  lcd.setCursor(0, 3);
  lcd.print("Radioempfang!");
  delay(3000);
  lcd.clear();
}
lcd.setCursor(0, 1);
lcd.print("Wifi gestartet ... ");

// SPI starten
SPI.setHwCs(true);
SPI.begin();

// Decoder starten
if (!stream.startDecoder(VS1053_CS, VS1053_DCS, VS1053_DREQ) || !stream.isChipConnected())
{
  Serial.println("Decoder konnte nicht gestartet werden ..");
  Serial.println("Programm angehalten!");
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Programm");
  lcd.setCursor(0, 1);
  lcd.print("angehalten");
  while (1);
}

stream.setTone(rtone);

Serial.println("MP3-Decoder gestartet ...");
lcd.setCursor(0, 2);
lcd.print("Decoder gestartet ..");
delay(3000);
time(&aktuelleZeit);

// localtime_r -> Zeit in die lokale Zeitzone setzen
localtime_r(&aktuelleZeit, &Zeit);

// Zeitzone: Parameter für die zu ermittelnde Zeit
configTzTime(Zeitzone, Zeitserver);

lcd.setCursor(0, 3);
lcd.print("aktuelle Zeit holen");
int Zaehler = 0;

// beim Start entspricht das Datum der Unixtime: 1.1.1970
// das Datum soll erst angezeigt werden, wenn es korrekt ist
String Jahr = String(Zeit.tm_year + 1900);

// String Jahr nach "1970" durchsuchen
int Suche = Jahr.indexOf("1970");
```

```
// solange die Suche nicht erfolgreich ist
// maximal 40 Sekunden
while (Suche != -1  && Zaehler < 40)
{
  // aktuelle Zeit holen
  time(&aktuelleZeit);

  // localtime_r -> Zeit in die lokale Zeitzone setzen
  localtime_r(&aktuelleZeit, &Zeit);
  Jahr = String(Zeit.tm_year + 1900);

  // String Jahr nach "1970" durchsuchen
  Suche = Jahr.indexOf("1970");
  delay(1000);
  Zaehler ++;
}

// Datum/Zeit konnte nicht synchronisiert werden
if (Suche != -1)
{
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Zeit synchronisieren");
  lcd.setCursor(0,1);
  lcd.print("fehlgeschlagen ");
  lcd.setCursor(0,2);
  lcd.print("Keine ");
  lcd.setCursor(0,3);
  lcd.print("Anzeige der Zeit ...");
  AnzeigeZeit = false;
}

else
{
  // Datum/Zeit erfolgreich synchronisiert
  lcd.setCursor(0, 3);
  lcd.print("Zeit synchronisiert ");
  delay(3000);
}

// Sender verbinden
stream.connectToHost(Sender[SenderIndex]);

// wenn der Stream nicht mehr läuft ...
if (!stream.isRunning())
{
  Serial.println("Radio wird neu gestartet ...");
  stream.stopSong();
  stream.connectToHost(Sender[SenderIndex]);
}

// Codec
stream.setCodecCB(codecCallBack);

// Bitrate
stream.setBitrateCB(bitrateCallback);
```

```
  // Name des Senders
  // stream.setStationCB(stationCallback);

  // Metadaten des Senders
  stream.setInfoCB(SenderInfo);

  stream.setVolume(Lautstaerke);

  // Anzahl der Sender im Array feststellen
  AnzahlSender = sizeof(Sender) / sizeof(Sender[0]);
  Serial.println("Anzahl der Sender:" + String(AnzahlSender));
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(WiFi.SSID());
  lcd.setCursor(0, 1);
  lcd.print(String(WiFi.localIP().toString()));
  lcd.setCursor(0, 2);
  lcd.print("Signalst\xe1rke: " + String(WiFi.RSSI()));
  lcd.setCursor(0, 3);
  lcd.print(String(AnzahlSender) + " Sender");
  delay(5000);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(Beschreibung[SenderIndex]);
  lcd.setCursor(0, 1);
  lcd.print("Lautst\xe1rke: " + String(Lautstaerke) + "  ");
}

void loop()
{
  // decode() -> Daten lesen
  if (IrReceiver.decode())
  {
    // kurzes delay, damit nur ein Tastendruck gelesen wird
    delay(200);

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

    if (IrReceiver.decodedIRData.command > 0 && IrReceiver.decodedIRData.command < 95)
    {
      // Werte abfragen Zählung der Sender beginnt mit 0
      if (IrReceiver.decodedIRData.command == Sender_0)
      {
        SenderUmschalten(10);
        SenderIndex = 10;
      }
      if (IrReceiver.decodedIRData.command == Sender_1)
      {
        SenderUmschalten(0);
        SenderIndex = 0;
      }
      if (IrReceiver.decodedIRData.command == Sender_2)
      {
        SenderUmschalten(1);
        SenderIndex = 1;
      }
```

```
    if (IrReceiver.decodedIRData.command == Sender_3)
    {
      SenderUmschalten(2);
      SenderIndex = 2;
    }
    if (IrReceiver.decodedIRData.command == Sender_4)
    {
      SenderUmschalten(3);
      SenderIndex = 3;
    }
    if (IrReceiver.decodedIRData.command == Sender_5)
    {
      SenderUmschalten(4);
      SenderIndex = 4;
    }
    if (IrReceiver.decodedIRData.command == Sender_6)
    {
      SenderUmschalten(5);
      SenderIndex = 5;
    }
    if (IrReceiver.decodedIRData.command == Sender_7)
    {
      SenderUmschalten(6);
      SenderIndex = 6;
    }
    if (IrReceiver.decodedIRData.command == Sender_8)
    {
      SenderUmschalten(7);
      SenderIndex = 7;
    }
    if (IrReceiver.decodedIRData.command == Sender_9)
    {
      SenderUmschalten(8);
      SenderIndex = 8;
    }
    if (IrReceiver.decodedIRData.command == Ton_aus) TonAus();
    if (IrReceiver.decodedIRData.command == Sender_zurueck) SenderZurueck();
    if (IrReceiver.decodedIRData.command == Sender_vor) SenderVor();
    if (IrReceiver.decodedIRData.command == Lautstaerke_plus) Lauter();
    if (IrReceiver.decodedIRData.command == Lautstaerke_minus) Leiser();
    if (IrReceiver.decodedIRData.command == Beleuchtung_ein_aus) BeleuchtungUmschalten();
    if (IrReceiver.decodedIRData.command == TitelAnzeigeUmschalten)
    {
      Titelanzeigen = !Titelanzeigen;

      // vorhandene Info löschen
      lcd.setCursor(0, 2);
      lcd.print("                    ");
      lcd.setCursor(0, 3);
      lcd.print("                    ");
    }
  }
}
```

```
  // wenn Stream gestoppt -> neu verbinden
  if (!stream.isRunning())
  {
    stream.connectToHost(Sender[SenderIndex]);
  }

  // Stream abspielen
  stream.loop();
  delay(5);
  if(AnzeigeZeit) ZeitAnzeigen();
}

void SenderVor()
{
  // wenn der letzte Sender -> zurück auf 0
  if (SenderIndex >= AnzahlSender - 1) SenderIndex = 0;
  else SenderIndex++;
  Serial.println(SenderIndex);
  stream.stopSong();
  stream.connectToHost(Sender[SenderIndex]);
  InfoAnzeigen(SenderIndex);
}

void SenderZurueck()
{
  // wenn der letzte Sender -> zurück auf 0
  if (SenderIndex == 0) SenderIndex = AnzahlSender - 1;
  else SenderIndex--;
  Serial.println(SenderIndex);
  stream.stopSong();
  stream.connectToHost(Sender[SenderIndex]);
  InfoAnzeigen(SenderIndex);
}

void Leiser()
{
  if (Lautstaerke > 5) Lautstaerke -= 5;
  // if (Lautstaerke == 0) Lautstaerke += 5;
  GesicherteLautstaerke = Lautstaerke;
  stream.setVolume(Lautstaerke);
  Serial.println("Lautstärke: " + String(Lautstaerke));
  lcd.setCursor(0, 1);
  lcd.print("Lautst\xe1rke: " + String(Lautstaerke) + "  ");
}

void Lauter()
{
  if (Lautstaerke < 95) Lautstaerke += 5;
  GesicherteLautstaerke = Lautstaerke;
  stream.setVolume(Lautstaerke);
  Serial.println("Lautstärke: " + String(Lautstaerke));
  lcd.setCursor(0, 1);
  lcd.print("Lautst\xe1rke: " + String(Lautstaerke) + "  ");
}
```

```
void TonAus()
{
  if (Lautstaerke == 0)
  {
    stream.setVolume(GesicherteLautstaerke);
    lcd.setCursor(0, 1);
    Lautstaerke = GesicherteLautstaerke;
    lcd.print("Lautst\xe1rke: " + String(Lautstaerke) + "  ");
  }
  else
  {
    Lautstaerke = 0;
    stream.setVolume(Lautstaerke);
    Serial.println("Lautstärke: " + String(Lautstaerke));
    lcd.setCursor(0, 1);
    lcd.print("Ton aus         ");
  }
}

void SenderUmschalten(int SenderIndex)
{
  Serial.println(SenderIndex);
  stream.stopSong();
  stream.connectToHost(Sender[SenderIndex]);
  InfoAnzeigen(SenderIndex);
}

void BeleuchtungUmschalten()
{
  Beleuchtung = !Beleuchtung;
  if(Beleuchtung) lcd.setBacklight(true);
  else lcd.setBacklight(false);
}

void InfoAnzeigen(int SenderIndex)
{
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(Beschreibung[SenderIndex]);
  lcd.setCursor(0, 1);
  lcd.print("Lautst\xe1rke: " + String(Lautstaerke) + "  ");
}

// Codec
void codecCallBack(const char *codec)
{
  Serial.println("Codec: " + String(codec));
}

// Bitrate
void bitrateCallback(uint32_t bitrate)
{
  Serial.println("Bitrate: " + String(bitrate));
}
```

```
void stationCallback(const char *name)
{
  Serial.println(name);
}

// wenn Metadaten vorhanden sind
void SenderInfo(const char *info)
{
  Serial.println(info);

  if(Titelanzeigen)
  {
    String Titel = String(info) + " ";

    // vorhandene Info löschen
    lcd.setCursor(0, 2);
    lcd.print("                    ");
    lcd.setCursor(0, 3);
    lcd.print("                    ");

    // Sonderzeichen umwandeln
    Titel.replace("°", "\337");
    Titel.replace("ä", "\341");
    Titel.replace("ö", "\357");
    Titel.replace("ü", "\365");
    Titel.replace("ß", "\342");

    // Titel nach 20 Zeichen umbrechen
    lcd.setCursor(0, 2);
    lcd.print(Titel.substring(0, 20));
    lcd.setCursor(0, 3);
    lcd.print(Titel.substring(20, 40));
  }
}
void ZeitAnzeigen()
{
  // aktuelle Zeit holen
  time(&aktuelleZeit);

  // localtime_r -> Zeit in die lokale Zeitzone setzen
  localtime_r(&aktuelleZeit, &Zeit);

  lcd.setCursor(15, 1);
  // Stunde: wenn Stunde < 10 -> 0 davor setzen
  if (Zeit.tm_hour < 10) lcd.print("0");

  lcd.print(String(Zeit.tm_hour));
  lcd.print(":");

  // Minuten
  if (Zeit.tm_min < 10) lcd.print("0");
  lcd.print(String(Zeit.tm_min));
}
```