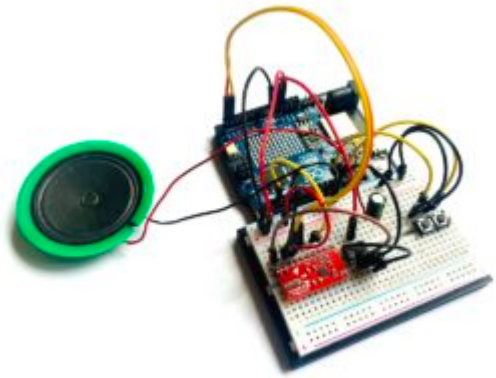
Arduino UKW-Radio – Musik hören ohne Internet



Wer hat nicht schon einmal davon geträumt, sein eigenes Radio zu bauen? Bei Pollux Labs findest du ein Internetradio, aber solange es die klassische UKW noch gibt, spricht auch nichts gegen einen passenden Empfänger. In diesem Projekt lernst du, wie du aus wenigen, günstigen Bauteilen ein **UKW-Radio mit Sendersuchlauf** erstellen kannst. Das Herzstück des Projekts ist ein Arduino Uno, der als Gehirn des Radios dient. Dazu kommt ein spezielles Radio-Modul (Si470x), das den Empfang übernimmt, und einen LM386-Verstärker, um den Sound auf einen Lautsprecher zu bringen.

Dieses Projekt ist perfekt für Einsteiger geeignet, die erste Erfahrungen mit dem Arduino, I²C-Kommunikation und einfachen Audioschaltungen sammeln möchten. Am Ende hast du nicht nur ein cooles, selbstgebautes Gadget, sondern auch ein tieferes Verständnis dafür, wie diese Komponenten zusammenspielen. Los geht’s!

Diese Bauteile brauchst du

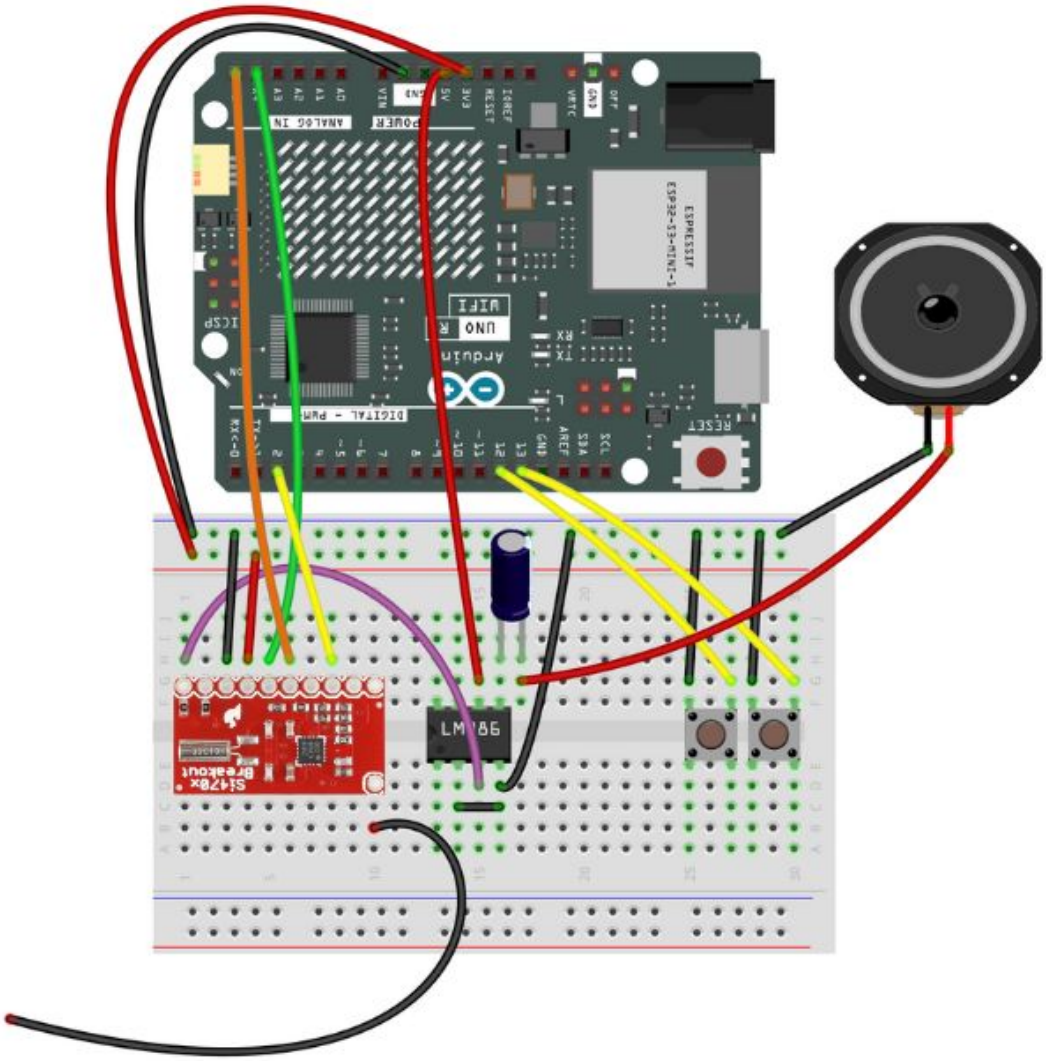
Für dieses Projekt benötigst du die folgenden Bauteile:

- **Mikrocontroller:** Arduino UNO*

- **Radio-Modul:** Si4703*
- **Verstärker:** LM386 Audioverstärker-IC*
- **Audio-Ausgabe:** 1x kleiner Lautsprecher (0,5 Watt mit 8 Ohm)
- **Bedienelemente:** 2x Push Buttons
- **Verkabelung:** Breadboard und diverse Jumper-Kabel
- **Antenne:** Ein ca. 75 cm langes Stück Draht

Der Aufbau auf dem Breadboard

Orientiere dich beim Aufbau des UKW-Radios an der folgenden Skizze.



___STEADY_PAYWALL___

Da auf dem Breadboard ziemlich viel los ist, hier noch einmal alle Verbindungen in einer Tabelle.

Bauteil (Von)	**Pin am Bauteil**	**Verbindung zu**	**Anmerkung**
Si4703 Radio-Modul	3.3V / VCC	3.3V am Arduino	**Wichtig: Nicht an 5V anschliessen!**
	GND	GND am Arduino	Gemeinsame Masse

	SDIO (SDA)	A4 am Arduino	I²C Datenleitung
	SCLK (SCL)	A5 am Arduino	I²C Taktleitung
	RST	D2 am Arduino	Reset-Leitung
	LOUT (Left Out)	**Input / +** am LM386	Audio-Signal zum Verstärker
	ANT	ca. 75cm Draht	Die Antenne, ins Leere hängend
LM386 Verstärker	VS	5V am Arduino	Strom für den Verstärker
	GND	GND am Arduino	Gemeinsame Masse
	+INPUT	LOUT vom Radio	**Das Audiosignal geht hier rein**
	-INPUT	GND	Dient als Referenz für +INPUT
	VOUT	Zum **Lautsprecher** mit 100uF Kondensator	Verstärkter Audio-Ausgang
Taster 1	Ein Pin	D13 am Arduino	Sendersuche aufwärts (Seek Up)
	Anderer Pin	GND am Arduino	
Taster 2	Ein Pin	D12 am Arduino	Sendersuche abwärts (Seek Down)
	Anderer Pin	GND am Arduino	

So funktioniert das Radio

Ohne Antenne kein Radioempfang, klar. Hierfür reicht bereits ein langes, isoliertes Kabel (ca. 75 – 100cm) aus der Werkstatt. Du kannst die Antenne direkt an das Radio-Modul löten oder es in dein Breadboard stecken.

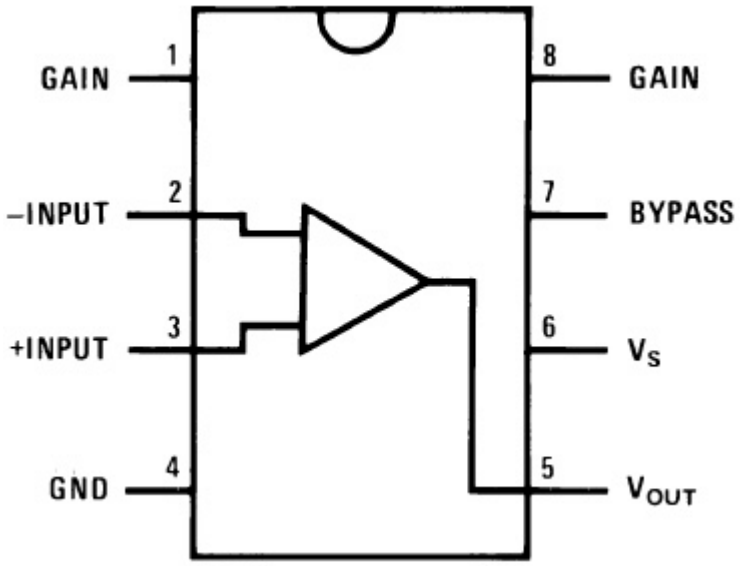
Mit den beiden Buttons startest du die Sendersuche: Der rechte sucht aufwärts nach Frequenzen mit Empfang, der linke Button entsprechend abwärts.

Und noch einige Worte zum Lautsprecher: Diesen schließt du nicht direkt an den Ausgang des LM386 an, sondern mit einem dazwischengeschalteten **Elektrolyt-Kondensator (Elko) mit 100uF**. Der Kondensator wirkt wie ein Filter: **Er blockiert Gleichspannung und lässt nur das Audiosignal zum Lautsprecher durch.** Man nennt ihn auch Koppelkondensator. Achte beim Einbau des Elkos unbedingt auf die Polarität – Plus kommt an den LM386 und Minus (gekennzeichnet durch einen Streifen) vor den Anschluss des Lautsprechers.

Ein genauerer Blick: Der LM386 Audioverstärker

Das Audiosignal, das direkt aus dem Radio-Modul kommt, ist viel zu schwach, um einen Lautsprecher anzutreiben. Hier kommt der LM386 ins Spiel. Er ist ein klassischer und sehr robuster Audioverstärker-Chip, perfekt für Hobby-Projekte. Seine Aufgabe ist es, das leise Signal vom LOUT-Pin des Radios aufzunehmen und es so zu verstärken, dass wir es deutlich hören können.

Da der Chip selbst keine Beschriftung hat, ist ein Pinout-Diagramm unerlässlich, um die Anschlüsse zu identifizieren. Achte auf die Kerbe an einer der kurzen Seiten des Chips – damit identifizierst du die linke und rechte Seite des LM386.



Die wichtigsten Anschlüsse für dieses Projekt:

- **Pin 6 (Vs) & Pin 4 (GND):** Hier wird der Verstärker mit Strom versorgt (5V und Masse).
- **Pin 3 (+INPUT):** An diesen „nicht-invertierenden“ Eingang legen wir unser Audiosignal vom Radio an.
- **Pin 2 (-INPUT):** Der „invertierende“ Eingang dient als Referenz und wird einfach mit Masse (GND) verbunden.
- **Pin 5 (Vout):** Hier kommt das verstärkte Signal für den Lautsprecher heraus.

Die übrigen Pins (Gain, Bypass) benötigen wir für eine einfache Schaltung nicht, sie könnten aber für eine höhere Lautstärke oder bessere Rauschunterdrückung genutzt werden.

Noch ein Hinweis: Das Radio-Modul gibt es auch mit einer Kopfhörer-Buchse. Bei dieser Version kannst du dir die Verstärkung über den LM386 und sogar die Antenne sparen. Sowohl der Ton als auch der Empfang kommt dann über die Kopfhörer bzw. deren Kabel.

Die passende Bibliothek für das Si403 Radio-Modul

Damit dein Arduino mit dem Radio-Modul sprechen kann, benötigt er eine passende Bibliothek. Diese findest du im Bibliotheksmanager deiner Arduino IDE. Suche dort einfach nach **SI470x** und installiere die aktuelle Version der Bibliothek **PU2CLR SI470X**. Das **X** steht hier für die verschiedenen Varianten des Modul – neben dem 4703 gibt es auch die Modelle 4701 und 4702.

Der Sketch für das Arduino UKW-Radio

Nun zur Software für deinen Radioempfänger. Kopiere den folgenden Sketch und lade ihn auf deinen Arduino:

```
/*
  UKW-Radio mit Arduino Uno, Si470x und LM386
*/

#include <Arduino.h>
#include <Wire.h>      // Für die I2C-Kommunikation
#include <SI470X.h>    // Die Bibliothek für das Radio-Modul

// Pin-Definitionen
const int RESET_PIN = 2;       // Pin für den Reset des Radio-Moduls
const int SEEK_UP_PIN = 13;     // Taster für Sendersuche aufwärts
const int SEEK_DOWN_PIN = 12;  // Taster für Sendersuche abwärts

// Radio-Objekt erstellen
SI470X radio;

// --- Funktion zur Ausgabe der aktuellen Sender-Informationen ---
```

```
void printRadioStatus() {
  // Aktuelle Daten vom Radio-Chip holen
   float currentFrequency = radio.getFrequency() / 100.0; // Frequenz von kHz in MHz umrechnen
  int currentRSSI = radio.getRssi();

  // --- AUSGABE IM SERIELLEN MONITOR ---
  Serial.println("--------------------");
  Serial.print("Aktueller Sender: ");
    Serial.print(currentFrequency, 2); // Zeigt zwei Nachkommastellen an
  Serial.println(" MHz");
  Serial.print("Signalstaerke: ");
  Serial.print(currentRSSI);
  Serial.println(" dBuV");
  Serial.println("--------------------");
}

void setup() {
   // Starte die serielle Kommunikation für die Ausgabe am Computer
  Serial.begin(9600);
  Serial.println("\n\nArduino UKW-Radio wird gestartet...");

   // Konfiguriere die Taster-Pins als Eingänge mit internem Pull-up-Widerstand.
  pinMode(SEEK_UP_PIN, INPUT_PULLUP);
  pinMode(SEEK_DOWN_PIN, INPUT_PULLUP);

  // Initialisiere die I2C-Kommunikation und das Radio-Modul
  Wire.begin();
   radio.setup(RESET_PIN, A4); // A4 ist der SDA-Pin beim Arduino Uno
  Serial.println("Si470x Radio initialisiert.");
  // Setze einen Schwellenwert für den Sendersuchlauf.
  radio.setSeekThreshold(45);
    Serial.println("Sendersuchlauf-Schwelle auf 45 dBuV gesetzt.");
  // Stelle eine mittlere Lautstärke ein (0-15)
  radio.setVolume(5);
  Serial.println("Lautstaerke auf 5 gesetzt.");
```

```
  // Suche automatisch den ersten verfügbaren Sender nach dem
Start
  Serial.println("Suche ersten Sender...");
   radio.seek(0, 1); // Mode 0 = am Bandende weitersuchen,
Direction 1 = aufwärts
  // Gib den Status des ersten Senders aus
  printRadioStatus();
  Serial.println("Setup abgeschlossen. Radio ist bereit.");
}

void loop() {
  // Prüfe, ob der "Suchen Aufwärts"-Taster gedrückt wurde
  if (digitalRead(SEEK_UP_PIN) == LOW) {
     Serial.println("Taster 'Seek Up' gedrueckt -> Suche
naechsten Sender...");
    radio.seek(0, 1);
    delay(200); // Entprellung des Tasters
    printRadioStatus();
  }

  // Prüfe, ob der "Suchen Abwärts"-Taster gedrückt wurde
  if (digitalRead(SEEK_DOWN_PIN) == LOW) {
     Serial.println("Taster 'Seek Down' gedrueckt -> Suche
vorigen Sender...");
    radio.seek(0, 0); // Direction 0 = abwärts
    delay(200);
    printRadioStatus();
  }

  // Kurze Pause, damit der Loop nicht überlastet wird
  delay(50);
}
```

So funktioniert der Sketch

Auch wenn der Code auf den ersten Blick komplex aussieht, lässt er sich in drei logische Blöcke unterteilen. Schauen wir uns die wichtigsten Befehle genauer an.

- **Der setup()-Block: Die Vorbereitung** Dieser Teil wird nur einmal beim Start des Arduinos ausgeführt.
    - Serial.begin(9600);: Startet die Kommunikation mit dem Computer, damit wir uns Nachrichten im Seriellen Monitor ansehen können.
    - pinMode(..., INPUT_PULLUP);: Konfiguriert die Pins für die Buttons zur Sendersuche. Der Arduino aktiviert einen internen Widerstand, der dafür sorgt, dass der Pin ein klares HIGH-Signal hat, solange der Taster nicht gedrückt wird. Das macht die Schaltung stabiler.
    - radio.setup(RESET_PIN, A4);: Weckt das Radio-Modul auf und stellt die I²C-Kommunikation her.
    - radio.setSeekThreshold(45);: Setzt den „Qualitätsfilter“ für den Suchlauf. Nur Sender mit einer Signalstärke von mindestens 45 dBuV werden berücksichtigt. Falls du zu wenige Sender empfängst, wähle einen niedrigeren Wert.
    - radio.setVolume(5);: Stellt die Grundlautstärke ein. Auch diese kannst du bei Bedarf anpassen.
    - radio.seek(0, 1);: Startet den ersten automatischen Suchlauf, um direkt nach dem Einschalten einen Sender zu finden.

- **Der loop()-Block: Die Endlosschleife** Dieser Code wird immer wieder ausgeführt, solange das Radio eingeschaltet ist.
    - Der Arduino prüft ständig, ob die Buttons für die Sendersuche gedrückt wurden.
    - Wenn einer der Buttons gedrückt wird, wird der Befehl zum Suchen an das Radio gesendet.

- **Die printRadioStatus()-Funktion:** Jedes Mal, wenn sie aufgerufen wird, holt sie sich die aktuelle Frequenz (radio.getFrequency()) und Signalstärke (radio.getRssi()) vom Modul und gibt sie sauber

formatiert im Seriellen Monitor aus.

So erweiterst du das Arduino Radio

Der bisherige Aufbau ist vielleicht nur der Anfang: Du kannst ein OLED-Display anschließen und darauf die aktuelle Frequenz anzeigen. Eine sehr sinnvolle Erweiterung ist ein Potentiometer, mit dem du die Lautstärke regeln kannst. Und zuletzt bietet das Si403 Radio-Modul auch RDS (Radio Data System) – damit kannst du weitere Informationen empfangen, sofern sie die Sender zur Verfügung stellen bzw. über ihre Frequenz mitsenden.