

Entwicklung eines Systems zur Steuerung eines
Auto-Radios außerhalb eines Fahrzeugs: Untersuchung und
Ersatz des Canbus-Adapters

PRAXISPROJEKT

ausgearbeitet von

Andreas Schurawlev

vorgelegt an der

TECHNISCHEN HOCHSCHULE KÖLN
CAMPUS GUMMERSBACH
FAKULTÄT FÜR INFORMATIK UND
INGENIEURWISSENSCHAFTEN

im Studiengang
MEDIENINFORMATIK

Erster Prüfer/in: Prof. Matthias Boehmer
Technische Hochschule Köln

Gummersbach, im Mai 2024

Adressen: Andreas Schurawlev
Ackermannstr. 2
92224 Amberg
andreas.schurawlev@smail.th-koeln.de

Prof. Matthias Boehmer
Technische Hochschule Köln
Institut für Informatik
Steinmüllerallee 1
51643 Gummersbach
matthias.boehmer@th-koeln.de

Inhaltsverzeichnis

1 Einleitung	3
1.1 Konzept	3
2 Grundlagen	5
2.1 Auto-Radio und Funktionen	5
2.1.1 Betrieb des Auto Radios	5
2.1.2 Besondere Funktionen und Einschränkungen	5
2.1.3 Signalübertragung über den Canbus-Adapter	5
2.2 Canbus-Adapter	5
3 Analyse und Verständnis der CAN-Kommunikation	7
3.1 MCP2515	7
3.2 Verständnis, Lesen und Senden von CAN-Daten	7
3.3 Test mit zwei Arduinos	9
3.3.1 MCP2515 Library	10
3.3.2 Konfiguration des MCP2515 in Arduino	10
3.3.3 Senden eines CAN-Frames	11
3.3.4 Lesen eines CAN-Frames	12
3.4 Auslesen der Fahrzeug-CAN-Daten	14
4 Entscheidung des Auto-Radios	16
5 Entwicklung des Prototypen	17
5.1 Teil 1 - Ersatz des Canbus-Adapters	17
5.1.1 Ergebnisse	25
5.1.2 Fazit vom Prototypen Teil 1	25
5.2 Teil 2 - Integration der Lenkradsteuerung	25
5.2.1 Fazit	29
6 Gesamtergebnis	30
Abbildungsverzeichnis	33
Quellenverzeichnis	34

Kurzfassung

Die Verbindung von Auto-Radio- und Canbus-Systemen spielt eine entscheidende Rolle im Kontext moderner Fahrzeugtechnik. Angesichts des steigenden Bedarfs an individuellen Fahrzeuganpassungen und der wachsenden Komplexität von Fahrzeugsystemen wird die Entwicklung flexibler Lösungen zur Steuerung und Anpassung von Funktionen immer wichtiger. Diese Arbeit konzentriert sich hauptsächlich darauf, zu erforschen, wie ein Auto-Radio effizient außerhalb eines Fahrzeugs eingesetzt werden kann. Dabei werden die Herausforderungen und Lösungsansätze beleuchtet, die eine Integration und Anpassung von Fahrzeugfunktionen ermöglichen. Besonderes Augenmerk wird auf die Erkundung alternativer Methoden zur Nutzung des Auto-Radios gelegt, ohne dabei direkt auf Canbus-Daten zugreifen zu müssen.

1 Einleitung

Die Integration von Auto-Radio- und Canbus-Systemen spielt eine bedeutende Rolle in der modernen Fahrzeugtechnik. Mit dem zunehmenden Bedarf an individuellen Fahrzeuganpassungen und der steigenden Komplexität von Fahrzeugsystemen gewinnt die Entwicklung von Lösungen zur flexiblen Steuerung und Anpassung von Fahrzeugfunktionen an Bedeutung. Das Hauptaugenmerk dieser Arbeit liegt auf der Untersuchung der Möglichkeiten, wie ein Auto-Radio effektiv außerhalb eines Autos genutzt werden kann. Dabei stehen die Herausforderungen und Lösungen im Mittelpunkt, um eine Integration und Anpassung von Fahrzeugfunktionen zu ermöglichen. Ein besonderer Schwerpunkt liegt dabei auf der Erkundung alternativer Ansätze zur Nutzung des Auto-Radios ohne direkten Zugriff auf Canbus-Daten.

1.1 Konzept

Um die Nutzung eines Auto-Radios außerhalb eines Autos zu ermöglichen und zu testen, soll ein Canbus-Adapter durch einen Mikrocontroller wie Arduino oder ESP32 ersetzt werden. Dieser Mikrocontroller wird programmiert, um die Funktionen des Canbus-Adapters zu simulieren und gleichzeitig die Steuerung des Auto-Radios zu ermöglichen. Der Prototyp soll es ermöglichen, wichtige Funktionen des Auto-Radios per Knopfdruck zu bedienen, ähnlich wie im Fahrzeug selbst.

Für die Entwicklung des Prototyps werden zunächst alle erforderlichen Komponenten besorgt, einschließlich des Auto-Radios mit dem Canbus-Adapter und des entsprechenden Mikrocontrollers. Anschließend erfolgt die Analyse der Canbus-Kommunikation, um ein Verständnis für die übertragenen Daten und deren Bedeutung für die Funktionalität des Auto-Radios zu erhalten.

Der Mikrocontroller wird programmiert, um die Canbus-Funktionalität des Adapters zu simulieren und die entsprechenden Signale an das Auto-Radio zu senden. Dabei werden Funktionen implementiert, die es ermöglichen, das Radio über Schalter oder Taster zu bedienen. Der Prototyp wird ausführlich getestet, um sicherzustellen, dass alle Funktionen korrekt funktionieren und eine zuverlässige Steuerung des Auto-Radios gewährleistet ist.

Visuelle Darstellung des gewünschten Ergebnis

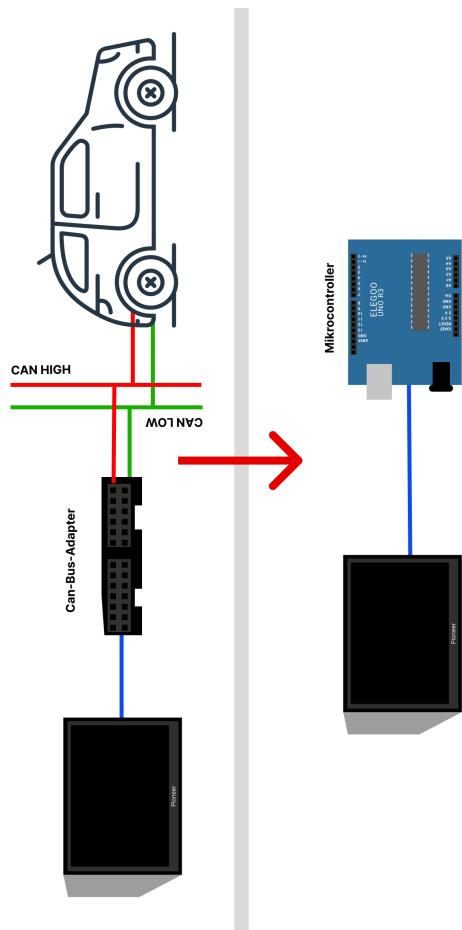


Abbildung 1.1: Ziel zur Entfernung des Canbus-Adapters und der Steuerung des Auto-Radios ohne Auto (Quelle: eigene Darstellung)

2 Grundlagen

2.1 Auto-Radio und Funktionen

Im folgenden Abschnitt werden grundlegende Informationen über das Auto-Radio und seine Funktionen erläutert. Dies beinhaltet:

2.1.1 Betrieb des Auto Radios

Das Auto-Radio benötigt eine Spannungsversorgung von 12V DC, die üblicherweise vom Fahrzeughalterie geliefert wird. Es wird üblicherweise durch die Zündung des Fahrzeugs aktiviert und deaktiviert.

2.1.2 Besondere Funktionen und Einschränkungen

Einige Funktionen des Auto-Radios, wie z.B. die Bluetooth-Konnektivität, sind möglicherweise nur verfügbar, wenn bestimmte Bedingungen erfüllt sind, wie z.B. das anziehen der Handbremse oder das Einschalten der Beleuchtung bei Dunkelheit. Zusätzlich können auch Signale von der Lenkradsteuerung an das Radio gesendet werden, um bestimmte Funktionen zu steuern.

2.1.3 Signalübertragung über den Canbus-Adapter

Bestimmte Signale, die zur Steuerung des Auto-Radios benötigt werden, werden über den Canbus-Adapter gesendet. Dazu gehören Informationen wie der Status der Zündung, die Position der Handbremse und andere wichtige Daten, die für die korrekte Funktionsweise des Radios erforderlich sind.

Durch das Verständnis dieser Grundlagen wird ein fundierter Hintergrund für die Integration und Anpassung des Auto-Radios außerhalb eines Autos geschaffen.

2.2 Canbus-Adapter

Der Controller Area Network (CAN-Bus) ist ein standardisiertes Netzwerkprotokoll, das in modernen Fahrzeugen zur Kommunikation zwischen verschiedenen elektronischen Steuergeräten verwendet wird. Es ermöglicht die Übertragung von Daten zwischen verschiedenen Komponenten des Fahrzeugs, wie z.B. Motorsteuerung, Getriebebesteuerung und vieles mehr¹

¹siehe CAN-BUS im Auto erklärt von ARS24

2 Grundlagen

Die Integration eines Auto-Radio-Systems in ein Fahrzeug, das keinen direkten Zugang zum Canbus hat, erfordert einen speziellen Adapter. Es ist von entscheidender Bedeutung, den Canbus-Adapter speziell für das entsprechende Fahrzeugmodell zu wählen. Dies liegt daran, dass jedes Auto unterschiedliche Canbus-Daten aufweist, die spezifisch für seine Bauweise und Ausstattung sind. Daher müssen Canbus-Adapter sorgfältig ausgewählt werden, um die spezifischen Daten eines bestimmten Fahrzeugs korrekt zu lesen und zu verarbeiten.

Ein Canbus-Adapter für ein Auto-Radio dient als Bindeglied zwischen dem Radio und dem Fahrzeug. Durch den Einsatz eines solchen Adapters kann das Radio auf Signale wie Zündung, Rückwärtsgang (für die Rückfahrkamera), Lenkrad-Tasten-Fernbedienung und Speedpulse zugreifen. Die Verarbeitung dieser Daten ermöglicht es dem Radio, bestimmte Funktionen zu aktivieren oder zu deaktivieren, um eine nahtlose Integration in das Fahrzeug zu gewährleisten².

²Siehe Erläuterungen zum CAN-Bus im vorherigen Abschnitt

3 Analyse und Verständnis der CAN-Kommunikation

In dem Prototypen wird ein CAN-Controller verwendet, um CAN-Bus-Daten von einem Arduino zu senden oder zu empfangen. Der gewählte CAN-Controller ist der MCP2515. Das Ziel dieses Prototyps besteht darin, das Verständnis dafür zu vertiefen, wie korrekte Daten an den CAN-Bus-Adapter gesendet werden können. Durch das Erlernen dieses Prozesses können wir sicherstellen, dass die Kommunikation mit dem CAN-Bus effektiv und zuverlässig ist, was für die Integration von Hardware-Komponenten in Fahrzeugsystemen von entscheidender Bedeutung ist.

Der Grund für die Auswahl des MCP2515 als CAN-Controller liegt darin, dass es bereits viele verfügbare Bibliotheken gibt und die Implementierung vergleichsweise unkompliziert ist. Dies erleichtert die Entwicklung und Integration des CAN-Controllers in den Prototypen erheblich.

3.1 MCP2515

Der MCP2515 ist ein integrierter Schaltkreis, der als Stand-Alone-Controller für den CAN-Bus (Controller Area Network) fungiert. Seine Hauptfunktion besteht darin, die serielle Kommunikation zwischen verschiedenen Steuergeräten in einem System zu ermöglichen, wie zum Beispiel in einem Fahrzeug. Der MCP2515 übernimmt die Aufgabe des CAN-Controllers und ermöglicht es, CAN-Nachrichten zu senden und zu empfangen¹.

3.2 Verständnis, Lesen und Senden von CAN-Daten

Um CAN-Daten zu verstehen, zu lesen und zu senden, ist es wichtig, die Struktur der CAN-Nachrichten zu verstehen und wie sie interpretiert werden können. CAN-Nachrichten bestehen aus verschiedenen Komponenten, darunter die CAN-ID, die Data Length Code (DLC) und die Datenbytes selbst.

Um CAN-Nachrichten zu lesen, ist es notwendig, den Inhalt der Nachrichten zu analysieren und zu interpretieren. Die CAN-ID gibt Aufschluss darüber, von welchem Steuergerät die Nachricht stammt und welche Art von Nachricht es ist. Die Datenbytes enthalten die eigentlichen Informationen, die gesendet werden, wie zum Beispiel Sensorwerte, Befehle oder Statusinformationen.

¹Produktbeschreibung des MCP2515

3 Analyse und Verständnis der CAN-Kommunikation

Für das Senden von CAN-Nachrichten ist es wichtig, die CAN-ID sowie die Datenbytes korrekt zu setzen. Je nach Anwendung und den Anforderungen des Systems können verschiedene CAN-Nachrichtenformate verwendet werden, wie zum Beispiel Standard- oder erweiterte Rahmen.

Die Kommunikation über den CAN-Bus erfordert ein genaues Verständnis der CAN-Protokolle und der Funktionsweise des CAN-Busses. Durch das Lesen und Analysieren von CAN-Daten können wichtige Informationen über das Fahrzeug oder das System gewonnen werden, was für die Diagnose, Steuerung und Überwachung entscheidend sein kann.

3.3 Test mit zwei Arduinos

Das Ziel dieser Analyse besteht darin, eine LED mithilfe von CAN-Signalen ein- und auszuschalten. Dazu werden zwei Arduinos und zwei MCP2515-Controller verwendet.

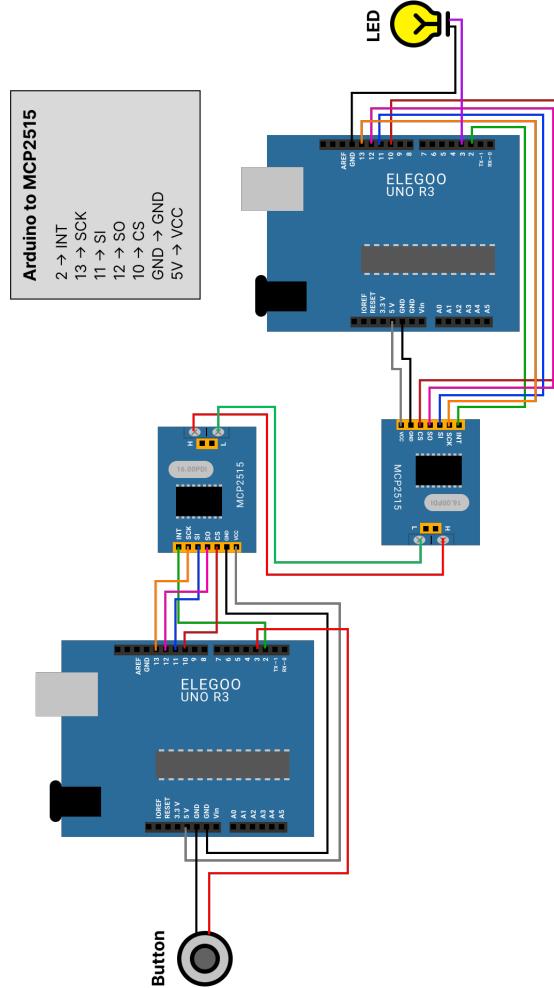


Abbildung 3.1: Schematik der Can-Kommunikation zwischen zwei Arduino's (Quelle: eigene Darstellung)

Im vorliegenden Setup agiert der erste Arduino als Lichtschalter. Sobald dieser ein- oder ausgeschaltet wird, sendet er über den MCP2515 ein entsprechendes Signal an den zweiten Arduino. Letzterer, ausgestattet mit einem weiteren MCP2515, empfängt die CAN-Daten und filtert sie. Je nach den empfangenen CAN-Daten wird die LED entsprechend ein- oder ausgeschaltet.

3.3.1 MCP2515 Library

Für die Implementierung der CAN-Kommunikation wurde die MCP2515-Bibliothek verwendet. Diese Bibliothek, zusammen mit einer Erweiterung durch eine neue Klasse `Canbus_Communicator`, bietet eine benutzerfreundliche Schnittstelle für die Kommunikation mit dem MCP2515 CAN-Controller. Besonders hervorzuheben ist die neue Klasse, die eine Vielzahl von Funktionen bereitstellt, um die CAN-Kommunikation zu vereinfachen².

Die Bibliothek unterstützt eine breite Palette von Geschwindigkeiten, was eine flexible Anpassung an verschiedene CAN-Bus-Systeme ermöglicht. Darüber hinaus wurden einige Funktionen entfernt, die für dieses Projekt nicht relevant sind, um die Bibliothek schlanker und einfacher zu handhaben.

3.3.2 Konfiguration des MCP2515 in Arduino

Die Integration des MCP2515 in Arduino-Programme erfordert eine genaue Konfiguration sowie die korrekte Implementierung entsprechender Funktionen. In diesem Abschnitt werden die notwendigen Schritte zur Initialisierung des MCP2515 und zur Kommunikation über den CAN-Bus mithilfe der `CANBUS_COMMUNICATOR`-Bibliothek erläutert.

```
1 #include "can_communicator.h"
2 #include "SPI.h"
3 void setup() {
4     Serial.begin(9600);
5
6     pinMode(BUTTON_PIN, INPUT_PULLUP);
7
8     SPI.begin();
9     can = new CANBUS_COMMUNICATOR(10, CAN_83K3BPS,
10                                MCP_16MHZ);
11    delay(100);
12 }
```

In Arduino-Programmen wird die `SPI.begin()`-Anweisung verwendet, um die Serial Peripheral Interface (SPI)-Kommunikation zu initialisieren. SPI ist ein serielles Kommunikationsprotokoll, das es ermöglicht, Daten zwischen Mikrocontrollern und anderen Peripheriegeräten in Echtzeit zu übertragen. Dies benötigen wir für die Kommunikation mit dem MCP2515.

Der `CANBUS_COMMUNICATOR` ermöglicht die Kommunikation über den CAN-Bus. Es werden verschiedene Parameter benötigt, um den CAN-Bus korrekt zu konfigurieren.

²Github-Repository von rnd-ash

Der erste Parameter (10) stellt den CS-Pin des MCP2515 dar. Der zweite Parameter (CAN_83k3BPS) gibt die CAN-Geschwindigkeit an (83.3 kBit/s). Es ist wichtig zu beachten, dass alle verwendeten MCP2515 denselben CAN-Speed verwenden müssen. Für Fahrzeuganwendungen muss der CAN-Speed entsprechend den Anforderungen des Fahrzeugs festgelegt werden, um eine korrekte Kommunikation zu gewährleisten. Eine falsche Geschwindigkeitseinstellung kann zu fehlerhaften CAN-Daten oder sogar zum Ausbleiben von CAN-Nachrichten führen. Der dritte Parameter (MCP_16MHZ) legt die Taktfrequenz des MCP2515 fest (16 MHz). Die Standardtaktfrequenz des MCP2515 beträgt 8 MHz und kann durch Entfernen und Neusetzen eines 16 MHz fähigen Quarzkristall erhöht werden.

3.3.3 Senden eines CAN-Frames

Der folgende Codeausschnitt zeigt eine Funktion zum Senden von CAN-Frames über den CAN-Bus. Es werden verschiedene Konstanten definiert, die die CAN-ID und die zu sendenden Daten repräsentieren. Die Funktion `sendToCanbus` akzeptiert ein Byte-Datenargument und erstellt einen CAN-Frame mit der definierten CAN-ID und den Daten. Anschließend wird der erstellte CAN-Frame über den `CANBUS_COMMUNICATOR` an den CAN-Bus gesendet.

```
1 #define CAN_ID_LIGHTS 0x0230
2 #define CAN_DATA_LIGHTS_TURN_ON 0x40
3 #define CAN_DATA_LIGHTS_TURN_OFF 0x00
4 #define CAN_DLC_LIGHTS 1
5
6 void sendToCanbus(uint8_t data){
7     can_frame send;
8     send.can_id = CAN_ID_LIGHTS;
9     send.can_dlc = CAN_DLC_LIGHTS;
10    send.data[0] = data;
11    can->sendToBus(&send);
12 }
```

3.3.4 Lesen eines CAN-Frames

Die folgende Funktion dient zum Lesen von CAN-Frames vom CAN-Bus. Zunächst werden einige Konstanten definiert, die die CAN-ID und die zugehörigen Daten für die Steuerung der LED repräsentieren. Die Funktion `handleMessage` ruft die Methode `read_frame` des `CANBUS_COMMUNICATOR` auf, um einen empfangenen CAN-Frame abzurufen.

```

1 #define CAN_ID_LIGHTS 0x0230
2 #define CAN_DATA_LIGHTS_TURN_ON 0x40
3 #define CAN_DATA_LIGHTS_TURN_OFF 0x00
4     void handleMessage(){
5         can_frame *read = can->read_frame();
6
7         if(read->can_dlc == 0)
8         {
9             return;
10        }
11
12        if(read->can_id == CAN_ID_LIGHTS)
13        {
14            if(read->data[0] == CAN_DATA_LIGHTS_TURN_ON)
15            {
16                digitalWrite(LED_PIN, HIGH);
17            }
18            if(read->data[0] == CAN_DATA_LIGHTS_TURN_OFF)
19            {
20                digitalWrite(LED_PIN, LOW);
21            }
22            can->printFrame(read);
23        }
24    }

```

Wenn die CAN-ID übereinstimmt, wird überprüft, ob die empfangenen Daten den Befehl zum Ein- oder Ausschalten der Scheinwerfer repräsentieren. Abhängig von den empfangenen Daten wird das LED-Pin entweder auf HIGH oder LOW gesetzt, um die LED entsprechend ein- oder auszuschalten. Schließlich wird der empfangene CAN-Frame mit der Methode `printFrame` des `CANBUS_COMMUNICATOR` ausgegeben.

Ein Beispiel für einen gedruckten CAN-Frame ist:

- 2411,560,1,64 - für "Ein" (Licht anschalten)
- 2565,560,1,0 - für "Aus" (Licht ausschalten)

Erklärung des gedruckten CAN-Frame:

- 2411 - Millisekunden
- 560 - ID - 0x0230 (hexadecimal zu decimal)
- 1 - DLC
- 64 - DATA - 0x40
- 0 - DATA - 0x00

Dieses Beispiel zeigt einen CAN-Frame mit vier Datenfeldern. Die ersten beiden Zahlen geben die Millisekunden und die CAN-ID des Frames an. Die dritte Zahl ist die Data Length Code (DLC), die angibt, wie viele Datenbytes im Frame enthalten sind. Die vierte Zahl sind die eigentlichen Daten, die im Frame übertragen werden. In diesem Fall bedeutet "Ein" (Licht anschalten), dass das Licht eingeschaltet wird, was durch den Wert "64" im Datenfeld angezeigt wird. "Aus" (Licht ausschalten) wird durch den Wert "0" im Datenfeld repräsentiert.

3.4 Auslesen der Fahrzeug-CAN-Daten

Das Ziel des Auslesens der CAN-Daten eines Mercedes-Benz W211 besteht darin, die Zündungs-ID zu finden und zu verstehen. Dazu wurde der MCP2515 mit dem Canbus-Interface des W211 verbunden. Im W211 gibt es zwei Schnittstellen: den Can-B und den Can-C Interface. Der Can-B ist unter anderem für das Infotainment-System des W211 zuständig, daher wird er für diesen Zweck genutzt.

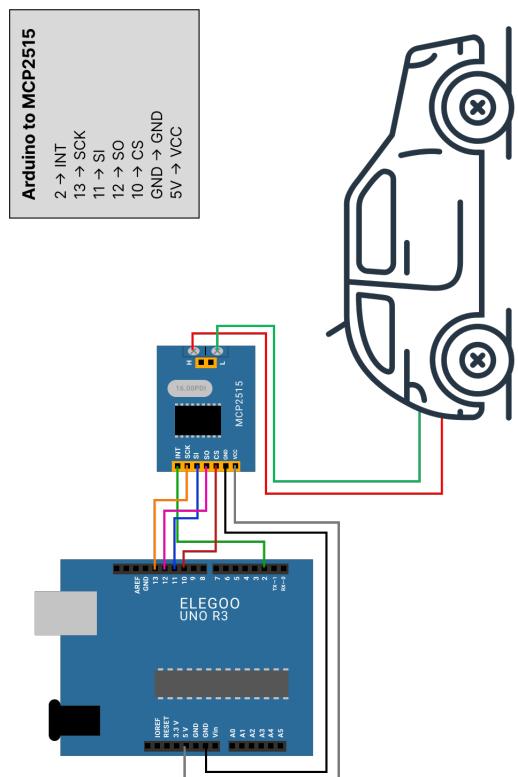


Abbildung 3.2: Schematik der Can-Kommunikation zwischen Auto und Arduino (Quelle: eigene Darstellung)

3 Analyse und Verständnis der CAN-Kommunikation

Es ist zu beachten, dass die CAN-Geschwindigkeit von Fahrzeug zu Fahrzeug variieren kann. Diese Information wurde aus verschiedenen Forenquellen bezogen³. Die Anpassung der CAN-Geschwindigkeit erfolgte erfolgreich und ermöglichte eine Kommunikation zwischen dem Arduino und dem Fahrzeug-CAN-Bus.

Während des Auslesevorgangs der CAN-Daten wurden sehr viele Daten gesammelt, jedoch wurde nur ein Bruchteil davon gespeichert. Während des Auslesens der CAN-Daten wurde die Zündung mehrmals an- und ausgeschaltet. Anschließend wurden die gesammelten Daten mit einem Python-Skript sortiert. Das Skript liest die erstellte TXT-Datei mit den CAN-Daten aus und sortiert sie nach ihrer ID. Dabei fiel auf, dass ein Wert mit der ID 0x00 häufig vorkommt. Wie im GitHub-Repository von rnd-ash⁴ zu sehen ist, entspricht dieser Wert der Klemme mit dem Namen "KL_50_EIN" - Klemme 50 ist eingeschaltet. Die Klemme 50 dient dazu, den Motor zu starten⁵.

Erkannte Zündungsdaten:

- Millis: 4322, ID: 0x0, DLC: 6, Data: [0x1, 0x0, 0x2, 0x0, 0x0, 0x0]
- Millis: 15119, ID: 0x0, DLC: 6, Data: [0x3, 0x0, 0x2, 0x0, 0x0, 0x0]

Hierbei steht der Wert "0x1"/"0x3" im ersten Byte für die Position des Schlüssels, während der dritte Byte "0x2" unverändert bleibt.

³Quelle der Foreninformationen

⁴siehe Github-Repository über die gesammelten CAN-Daten

⁵Informationen zur Klemme 50

4 Entscheidung des Auto-Radios

Nach eingehender Analyse fiel die Entscheidung auf das Auto-Radio Pioneer SPH-DA250DAB in Verbindung mit einem spezifischen Canbus-Adapter für das Modell W211 Mercedes Benz Baujahr 2004. Diese Wahl basiert auf mehreren Kriterien:

- Kompatibilität: Das Pioneer SPH-DA250DAB ist bekannt für seine breite Kompatibilität mit verschiedenen Fahrzeugmodellen, einschließlich des W211 Mercedes Benz.
- Funktionalität: Das Radio bietet eine Vielzahl von Funktionen, die den Anforderungen an modernes Infotainment und Fahrzeugsteuerung gerecht werden.
- Canbus-Integration: Durch die Verwendung eines dedizierten Canbus-Adapters, der speziell für das W211-Modell entwickelt wurde, wird eine nahtlose Integration mit den Fahrzeugsystemen gewährleistet. Dies ermöglicht eine effiziente Kommunikation und Steuerung über den Fahrzeug-CAN-Bus.
- Datenquelle: Die CAN-Bus-Daten können entweder direkt aus dem echten Fahrzeug oder aus vertrauenswürdigen Quellen wie einer GitHub-Seite bezogen werden. Die Richtigkeit und Zuverlässigkeit der Daten wurden sorgfältig überprüft und validiert.

Durch diese sorgfältige Auswahl des Auto-Radios und des entsprechenden Canbus-Adapters wird eine solide Grundlage für die Implementierung des Systems geschaffen, die sowohl den funktionalen Anforderungen als auch den Fahrzeug-spezifischen Eigenschaften gerecht wird.

5 Entwicklung des Prototypen

5.1 Teil 1 - Ersatz des Canbus-Adapters

Für die Umsetzung des Prototyps werden folgende Hardwarekomponenten benötigt:

- Arduino oder ESP32 (Mikrocontroller)
- Android Auto-Radio mit Canbus-Adapter
- MCP2515 (CAN-Bus-Controller)
- 12V-Netzteil (Stromversorgung)

Diese Komponenten bilden die Grundlage für die Entwicklung und Integration des Auto-Radio-Prototyps.

Verkabelung des Auto-Radios

Die Verkabelung des Auto-Radios erfolgte gemäß den Anweisungen im Installationsanleitung¹ des Pioneer SPH-DA250DAB. Dabei wurden die Kabel korrekt identifiziert und angeschlossen. Der gelbe Draht dient der 12V DC-Versorgung, während der schwarze Draht für die Erdung steht. Diese Informationen sind im Handbuch des Auto-Radios, SPH-DA250DAB zu finden.

Die Skizze beinhaltet nur für das Projekt benötigten Pins. Die anderen verbleibenen Pins sind für die Lautsprächanlage.

Trotz der sorgfältigen Verkabelung und des erfolgreichen anschliessen des Auto-Radios konnte dieses nicht eingeschaltet werden, da noch ein Zündungssignal erwartet wird.

Canbus-Adapter

Durch das prüfen der Kabelstrang wurde festgestellt welche Pins am Canbus-Adapter für welchen kabel zuständig sind.

Nachdem das Auto-Radio erfolgreich verkabelt wurde, wurde der Canbus-Adapter angeschlossen, um die CAN-Kommunikation zwischen dem Canbus-Adapter und dem Arduino zu ermöglichen.

¹Die Anweisung zum anschließen der Kabeln findet man auf der zweiten Seite unter Power Cord

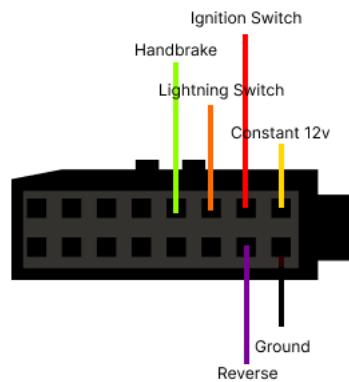


Abbildung 5.1: Skizze des Steckers vom Auto-Radio mit Pin-Bezeichnungen (Quelle: eigene Darstellung)

Senden des Zündungssignals

Um das Auto-Radio einzuschalten, wurde ein Zündungssignal an den Canbus-Adapter gesendet. Dieser Schritt war entscheidend, da das Radio eine Zündungsquelle benötigt, um ordnungsgemäß zu funktionieren. Das Zündungssignal wurde mithilfe eines Arduinos und des MCP2515-Moduls an den Canbus-Adapter gesendet.

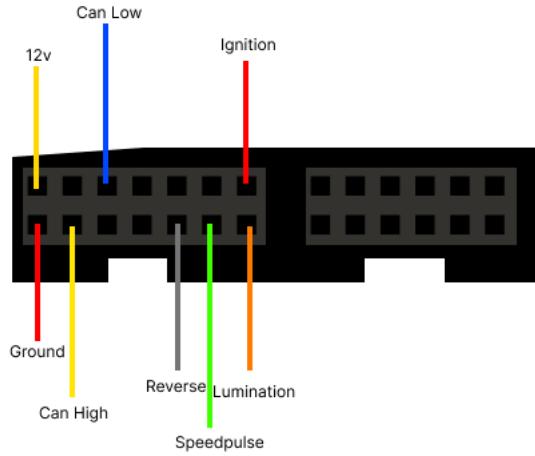


Abbildung 5.2: Skizze der benötigten Anschlüsse des Canbus-Adapters mit Pin-Bezeichnungen (Quelle: eigene Darstellung)

```

1 #include "can_communicator.h"
2 #include "SPI.h"
3
4 #define IGNITION_ID 0x0000
5 #define IGNITION_DLC 6
6
7 void KL_15C_TWO(){
8
9     can_frame sendFrame;
10    sendFrame.can_id = IGNITION_ID;
11    sendFrame.can_dlc = IGNITION_DLC;
12
13    for(int i = 0; i < sendFrame.can_dlc; i++){
14        sendFrame.data[i] = 0x00;
15    }
16
17    sendFrame.data[0] = 0x03;
18    sendFrame.data[2] = 0x02;
19    can->sendToBus(&sendFrame);
20 }
```

Der Wert für das Zündungssignal wurde gemäß der Echtzeitdaten des Autos festgelegt.

5 Entwicklung des Prototypen

Dabei entspricht die Ignition_id "0x0000" und die dlc "6". In Byte 0 wird der Wert "0x03" gesendet, um anzugeben, dass der Schlüssel in der dritten Position steht, während Byte 2 den Wert "0x02" behält.

Visuelle Darstellung der Verbindungen des Versuches

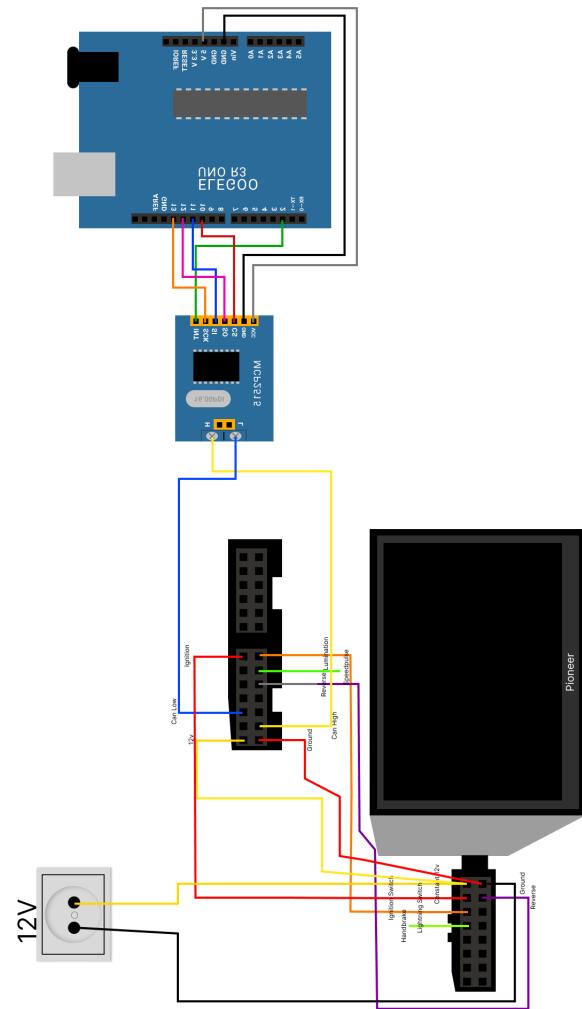


Abbildung 5.3: Skizze zur Verbindung vom Arduino zum Canbus-Adapter (Quelle: eigene Darstellung)

Erwartungen und Ergebnisse

Es wurde erwartet, dass das Senden des Zündungssignals über den Canbus-Adapter das Radio einschaltet. Allerdings wurden keine sofortigen Ergebnisse festgestellt, was auf mögliche Probleme hinweisen könnte.

Fehleranalyse

Selbst nach einer umfassenden Fehleranalyse blieb das Problem mit dem Zündungssignal bestehen. Es wurden verschiedene Konfigurationen und Einstellungen überprüft, um sicherzustellen, dass das Zündungssignal korrekt erkannt und verarbeitet wird. Trotzdem konnte das Zündungssignal nicht erfolgreich an den Canbus-Adapter gesendet werden. Um die Ursache genauer zu bestimmen, wurde der Canbus-Adapter in einem realen Fahrzeug (Modell W211) getestet, wo er ebenfalls nicht funktionierte. Diese Ergebnisse legen nahe, dass das Problem möglicherweise nicht auf eine fehlerhafte Konfiguration oder Einstellung zurückzuführen ist, sondern auf einen Defekt des Canbus-Adapters selbst oder auf eine Inkompatibilität mit dem Fahrzeugmodell. Es ist daher anzunehmen, dass entweder ein falsches Produkt geliefert wurde oder dass der Canbus-Adapter nicht für das spezifische Fahrzeugmodell geeignet ist.

Lösung

Da der Canbus-Adapter weiterhin nicht funktioniert hat und eine detaillierte Analyse nicht möglich war, wurde entschieden, ihn aus dem System zu entfernen, ohne weitere Untersuchungen durchzuführen. Über den Canbus-Adapters wurden Hypothesen darüber angestellt, wie er funktionieren könnte. Durch Tests im Arduino wurde anschließend eine Nachbildung seiner Funktionalität erstellt.

Funktionsweise eines CAN-Bus-Adapters

Durch den Austausch des Canbus-Adapters durch den Arduino war es möglich, die hypothetischen Annahmen zur Funktionsweise eines Canbus-Adapters zu verifizieren. Dies ermöglichte eine genauere Einsicht in die Arbeitsweise eines Canbus-Adapters sowie seine Interaktion mit dem Auto-Radio.

Ein CAN-Bus-Adapter liest Daten vom CAN-Bus und filtert sie entsprechend. Er sucht nach bestimmten Can-ID's, die bestimmte Signale oder Ereignisse im Fahrzeug repräsentieren.

Beispielsweise erkennt der Adapter das Zündungssignal, indem er nach der ID sucht, die mit dem Zündungsvorgang verknüpft ist. Wenn die entsprechende ID gefunden wird und die Daten darauf hinweisen, dass die Zündung eingeschaltet ist (durch das Vorhandensein einer bestimmten Bitposition) sendet der Canbus-Adapter ein 12V-Signal zum Zündungspin des Auto-Radios, wodurch das Radio eingeschaltet wird. Dieses Prinzip gilt auch für andere Signale, wie beispielsweise das Signal für die Rückfahrkamera oder das Ein- und Ausschalten der Scheinwerfer. Bei der Handbremse hingegen wird

5 Entwicklung des Prototypen

das Signal über die Ground-Verbindung gesendet. Sobald ein Ground-Signal am entsprechenden Pin im Radio anliegt, werden bestimmte Funktionen freigeschaltet, die während der Fahrt normalerweise nicht zugänglich sind.

Arduino ersetzt den Canbus-Adapter

Nachdem ein besseres Verständnis für die Funktionsweise des Canbus-Adapters entwickelt wurde, erfolgte die Ersetzung des Canbus-Adapters durch den Arduino. Dadurch konnte versucht werden, Signale für Zündung, Rückfahrkamera, Handbremse und Beleuchtung über den Arduino zu senden.

Um dies zu realisieren, wurden 4 Relais und 4 Taster verwendet. Da der Arduino nur 5V senden kann, wurde eine Lösung benötigt, um 12V-Signale zu senden. Hierfür wurde ein Stromverteiler verwendet, der an eine 12V-Stromquelle angeschlossen ist und sowohl die Relais als auch das Auto-Radio mit Strom versorgt.

Per Tastendruck ist es möglich die Relais ein- oder auszuschalten, um das passende Signal senden zu können.

Visuelle Darstellung der Verbindungen des Prototypen

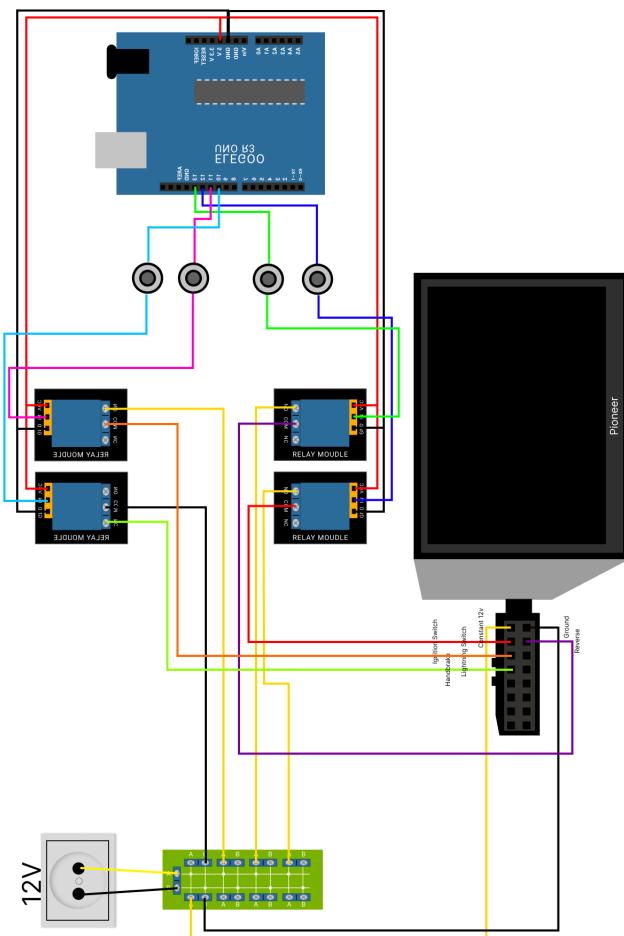


Abbildung 5.4: Skizze zum anschliessen des Auto-Radios mit dem Arduino (Quelle: eigene Darstellung)

5.1.1 Ergebnisse

Nach der Implementierung dieser Lösung wurde festgestellt, dass das Auto-Radio nun über das Zündungssignal eingeschaltet wird. Ebenso wird die Rückfahrkamera aktiviert, obwohl keine Kamera angeschlossen ist. Der Bildschirm wird außerdem dunkler, wenn das Illumination-Signal gesendet wird.

Bei der Handbremse ist zu beachten, dass sie ein Ground-Signal sucht, anstatt ein 12V-Signal. Daher sendet ein Relais bei Bedarf ein Ground-Signal, das per Knopfdruck aktiviert wird.

5.1.2 Fazit vom Prototypen Teil 1

Dieser Prototype hat sich als eine lehrreiche Erfahrung erwiesen, die uns ein tieferes Verständnis für die Integration von Auto-Radios und elektronischen Steuersystemen in Fahrzeugen vermittelt hat. Erfolgreich wurde der Canbus-Adapter durch den Arduino ersetzt und alle Funktionen des Auto-Radios sind steuerbar.

Obwohl einige Herausforderungen auftraten, wie das Ausbleiben sofortiger Ergebnisse beim Senden des Zündungssignals über den Canbus-Adapter, wurde durch gründliche Fehleranalyse und Hypothesenbildung eine alternative Lösung gefunden. Der Einsatz von Relais und Tastern ermöglichte es, die erforderlichen 12V-Signale zu senden und das Auto-Radio sowie weitere Funktionen wie die Rückfahrkamera und die Beleuchtung zu steuern.

Durch die Visualisierung der Verbindungen und die detaillierte Beschreibung der Implementierungsschritte wurde ein umfassendes Protokoll erstellt, das nicht nur unser Vorgehen dokumentiert, sondern auch anderen Forschern und Technikern als Leitfaden dienen kann.

5.2 Teil 2 - Integration der Lenkradsteuerung

In diesem Abschnitt wird die Integration der Lenkradsteuerung in das Auto-Radio behandelt. Nach Abschluss der Implementierung des Canbus-Adapters wird die Aufmerksamkeit auf weitere Funktionen des Radios gerichtet.

Funktionsprinzip der Lenkradsteuerung

Die Lenkradsteuerung erfolgt über die K-Line, was zunächst eine eingehende Recherche erforderte, um das genaue Funktionsprinzip zu verstehen. Durch das Anschließen eines Klinkensteckers an das Auto-Radio konnte mittels eines Multimeters festgestellt werden, dass 3,2V aus der Klinkenbuchse kommen. Dies deutet darauf hin, dass das Radio 3,2V durch die K-Line sendet.

Widerstandsbasierte Steuerung

In der Forschungsliteratur² wird auf Bildmaterial (Abbildung 9.1) hingewiesen, das darauf hinweist, dass die Steuerung über Widerstände erfolgt. Dies legt nahe, dass es erforderlich sein könnte, eine 3,2V-Spannung mit einem geeigneten Widerstand zu belegen, der für die jeweilige Funktion der Lenkradsteuerung verantwortlich ist

²Recherche zur Steuerung des Lenkrads

5 Entwicklung des Prototypen

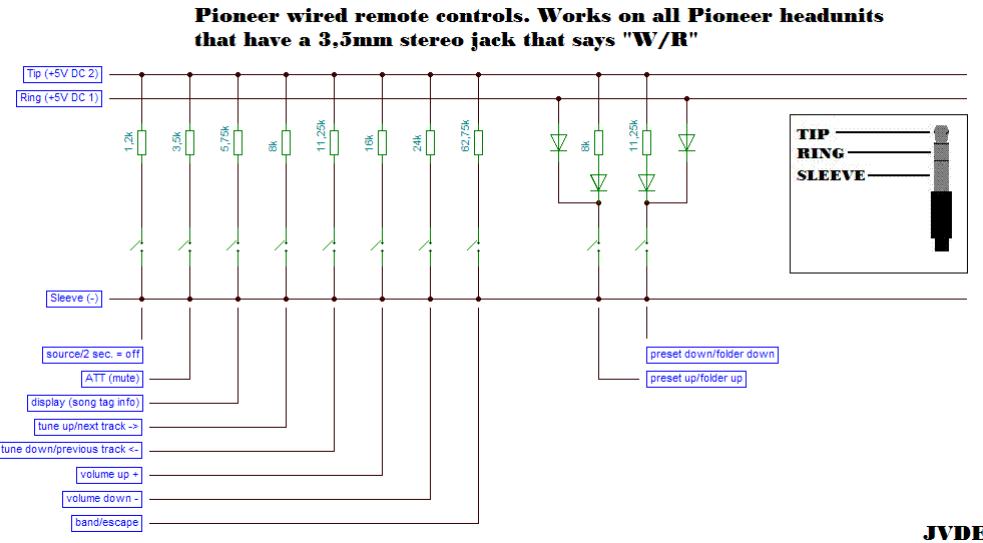


Abbildung 5.5: Schematik für die Funktionen des Lenkrads
(Quelle:<https://forum.arduino.cc/t/steering-wheel-remote-audio-control/223878>)

Kabelbelegung

Die Abbildung 9.1 zeigt, dass die Spitze für die Lautstärkeanpassung und die Auswahl des nächsten bzw. vorherigen Titels zuständig ist. Diese Funktionen wurde genauer untersucht. Der Ring ist nur in Kombination mit der Spitze wichtig für das Durchblättern von Ordnern, was jedoch in diesem Projekt nicht relevant ist.

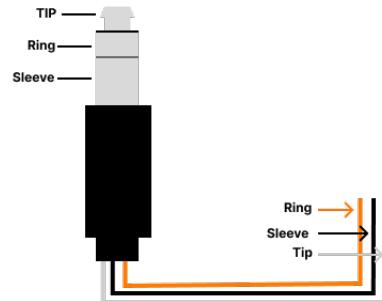


Abbildung 5.6: 3.5mm Klinke für die Lenkradsteuerung (Quelle: eigene Darstellung)

Anschluss

Für den Anschluss der Lenkradsteuerung wurde ein Breadboard und vier Taster verwendet.

Das orangefarbene Kabel (Ring) wird mit dem Pluspol des Breadboards verbunden, während das schwarze (Ground) mit dem Minuspol des Breadboards verbunden wird.

Von dort führen vier Kabel mit jeweiligen Widerständen vom Pluspol zu den Tastern. Jeder Taster hat einen eigenen Widerstand: ein 16k-Widerstand für Volume-Up, ein 24k-Widerstand für Volume-Down, ein 8k-Widerstand für Next Track und ein 11,25k-Widerstand für Previous Track. Wenn ein Taster betätigt wird, schließt sich der Stromkreis, wodurch die Klinke ein Signal mit einer Spannungsänderung von 3,2V durch den Widerstand erhält.

Dieses Setup ermöglicht, die Lenkradsteuerung durch manuelle Tastendrücke zu simulieren und die entsprechenden Funktionen des Auto-Radios zu steuern.

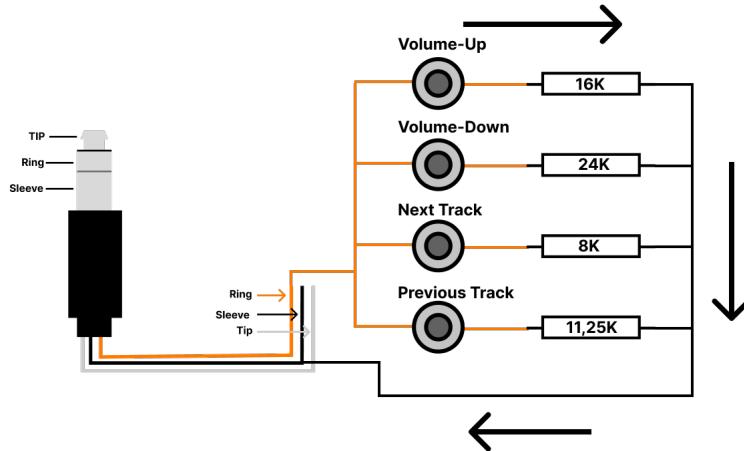


Abbildung 5.7: Verbindung der Klinke zum Taster mit dem passenden Widerstände

5.2.1 Fazit

Die Integration der Lenkradsteuerung in das Auto-Radio stellt eine wichtige Ergänzung dar, um die Benutzerfreundlichkeit und Funktionalität zu verbessern. Durch die manuelle Steuerung mittels Tastendrücken können Benutzer bequem auf verschiedene Funktionen des Radios zugreifen, ohne dabei auf die Bedienelemente am Lenkrad angewiesen zu sein.

Die Verwendung eines Breadboards und von Tastern ermöglicht es uns, die Lenkradsteuerung in einem prototypischen Umfeld zu simulieren und die entsprechenden Funktionen des Auto-Radios zu testen.

In Zukunft könnten weitere Verbesserungen durch den Einsatz digitaler Potentiometer über den Arduino realisiert werden, was eine effizientere Steuerung und eine einfachere Anpassung der Widerstandswerte ermöglichen würde.

6 Gesamtergebnis

In diesem Projekt wurde erfolgreich ein Auto-Radio außerhalb eines Fahrzeugs integriert und funktionsfähig gemacht. Durch die Integration von verschiedenen Komponenten wie einem Arduino, Relais und Tastern sowie der Analyse und Nachbildung der Funktionalität eines CAN-Bus-Adapters konnten wichtige Funktionen des Auto-Radios simuliert und gesteuert werden.

Die Implementierung erfolgte schrittweise, beginnend mit der Verkabelung des Auto-Radios gemäß den Herstelleranweisungen und der Untersuchung des CAN-Bus-Adapters zur Steuerung des Radios. Obwohl der Canbus-Adapter letztendlich nicht getestet werden konnte, konnten dennoch wichtige Erkenntnisse über seine Funktionsweise gewonnen werden.

Die Integration der Lenkradsteuerung über manuelle Tastendrücke ermöglichte eine intuitive Bedienung des Radios und eröffnete weitere Möglichkeiten zur Erweiterung der Funktionalität.

Die Dokumentation des Projekts bietet nicht nur einen detaillierten Einblick in den Entwicklungsprozess, sondern auch eine Grundlage für zukünftige Forschung und Entwicklung im Bereich der Auto-Radio-Integration.

Alle Skizzen, Codes und Materialien kann auf folgenden Plattformen gefunden werden:

- Skizzen und Diagramme sind in Figma unter folgendem Link verfügbar:
https://www.figma.com/file/HCI1JXIQsWgqXGb6VGVfra/PraxisProjekt_2024_Andreas_Schurawlev?type=design&node-id=0%3A1&mode=design&t=18j0DaEHEDXd031G-1
- Der Quellcode und weitere relevante Dateien sind im GitHub-Repository des Projektes zu finden:
https://github.com/cruv3/PraxisProjekt2024_AutoRadio_AndreasSchurawlev

Insgesamt zeigt dieses Projekt die Machbarkeit und Möglichkeiten, ein Auto-Radio außerhalb eines Fahrzeugs funktionsfähig zu machen und bietet einen Ausgangspunkt für weitere Untersuchungen und Entwicklungen auf diesem Gebiet.

6 Gesamtergebnis

Bild des Prototypen

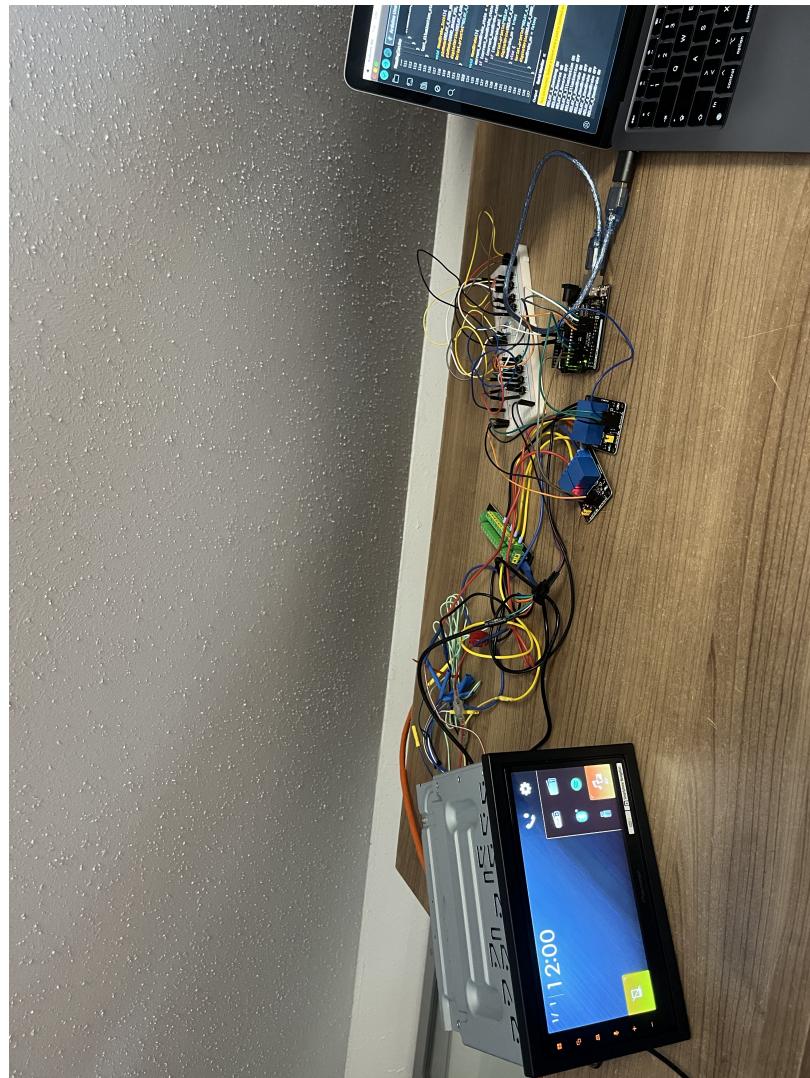


Abbildung 6.1: Eingeschalteter Zustand des Radios (Zündung - an)

Weitere Projektideen

Als nächste Schritte könnten folgende Projektideen in Betracht gezogen werden:

Integration von DREA (Driver's Reaction Evaluation Application)

Die Integration von DREA, dass die Steuerung der Lenkradtasten ermöglicht, könnte eine interessante Erweiterung sein. Durch die Integration von DREA in das Auto-Radio könnte eine verbesserte Benutzererfahrung geschaffen werden, die es ermöglicht, verschiedene Funktionen des Radios intuitiv und sicher zu steuern.

Integration eines Tachometers

Die Integration eines Tachos, der entweder über den CAN-Bus oder direkt mit dem Auto-Radio kommuniziert, könnte eine weitere interessante Projektidee sein. Ein Tacho könnte dem Fahrer zusätzliche Informationen und Funktionen bieten, wie z.B. die Anzeige von Fahrzeuggeschwindigkeit, Drehzahl, Kraftstoffverbrauch und vieles mehr.

Integration eines echten Lenkrads

Die Integration eines echten Lenkrads, das über den CAN-Bus oder direkt mit dem Auto-Radio kommuniziert. Ein echtes Lenkrad könnte dem Entwickler ein realistischeres und intuitiveres erlebnis bieten, indem es die Bedienung verschiedener Funktionen des Radios über Lenkradtasten ermöglicht.

Diese Projektideen bieten interessante Möglichkeiten zur Weiterentwicklung der Integration von Auto-Radios und könnten dazu beitragen, die Funktionalität und Benutzererfahrung weiter zu verbessern.

Abbildungsverzeichnis

1.1	Gewünschtes Ergebnis	4
3.1	Can-Kommunikation zwischen zwei Arduino's	9
3.2	Can-Kommunikation zwischen Auto und Arduino	14
5.1	Auto-Radio Stecker (Vorderseite)	18
5.2	Canbus-Adapter Anschluss (Vorderseite)	19
5.3	Verbindung des Canbus-Adapters zum Arduino	21
5.4	Verbindung vom Auto-Radio zum Arduino	24
5.5	Funktionen des Lenkrads	27
5.6	Klinke für die Lenkradsteuerung	27
5.7	Schematic der Tasten mit Widerstand	28
6.1	Ergebnis des Prototypen	31

Quellenverzeichnis

[ars24.com] (2018, 23. Februar). CAN-BUS im Auto erklärt. Abgerufen von <https://ars24.video/video/can-bus-im-auto-erklaert-can-bus-adapter-autoradios-richtig-anschliessen> (zuletzt aufgerufen am 07.05.2024)

[reichelt.de] Produktbeschreibung des MCP2515. Abgerufen von <https://www.reichelt.de/de/de/can-controller-mit-spi-schnittstelle-dil-18-mcp-2515-i-p-p54514.html?r=1#:~:text=Der%20MCP2515%20von%20Microchip%20Technology,zu%20senden%20und%20zu%20empfangen>. (zuletzt aufgerufen am 07.05.2024)

(Github rnd-ash) MCP2515 Library mit der Erweiterung der CANBUS_COMMUNICATOR Klasse. Abgerufen von https://github.com/rnd-ash/W203-canbus/blob/master/ARDUINO_CODE/can_comm.h (zuletzt aufgerufen am 07.05.2024)

[xdaforums.com] Forumspost zum Thema "Mercedes-Benz CAN-BUS Speed" von Benutzer iwl. Abgerufen von <https://xdaforums.com/t/mercedes-benz-w211-can-bus-speed.3680320/> (zuletzt aufgerufen am 07.05.2024)

[kfz.josefscholz.de] Klemme 50. Abgerufen von <http://www.kfz.josefscholz.de/Anlasser.html#:~:text=Anlasser&text=sind%20in%20ihrem%20Aufbau%20nahezu,Befehl%20zum%20Anlassen%20des%20Motors> (zuletzt aufgerufen am 07.05.2024)

(Github rnd-ash) Mercedes-Benz gesammelte CAN-Daten. https://github.com/rnd-ash/W203-canbus/blob/master/DAT_TRANSLATOR/DECODED/w211_w219%20CAN%20GERMAN.txt (zuletzt aufgerufen am 07.05.2024)

[forum.arduino.cc] Erklärung der Klinkenfunktion für das Auto-Radio von Pioneer <https://forum.arduino.cc/t/steering-wheel-remote-audio-control/223878> (zuletzt aufgerufen am 07.05.2024)

[calatogs.pioneer-car.eu] Installationshandbuch von SPH_DA250DAB https://catalogs.pioneer-car.eu/Manuals/SPH_DA250DAB_CRD5173_installation_manual/?page=1 (zuletzt aufgerufen am 07.05.2024)