

Dies ist ein Arduino RC-Motoren-Sound- und Lichtcontroller für ESP32.

Rebefreiung

Version 9.13.0

Deigen laden S

0

COntRich Butors

3

Es basiert auf der ATmega 328-Version:https://github.com/TheDIYGuy999/Rc_Engine_Sound und zum Beispiel an Halloween auf Bitlunis:<https://github.com/bitluni/MotionPumpkin>

Die Verdrahtungs- und Softwareinstallationsanweisungen finden Sie weiter unten. Arduino IDE oder Visual Studio Code (mit PlatformIO-Erweiterung) werden unterstützt.

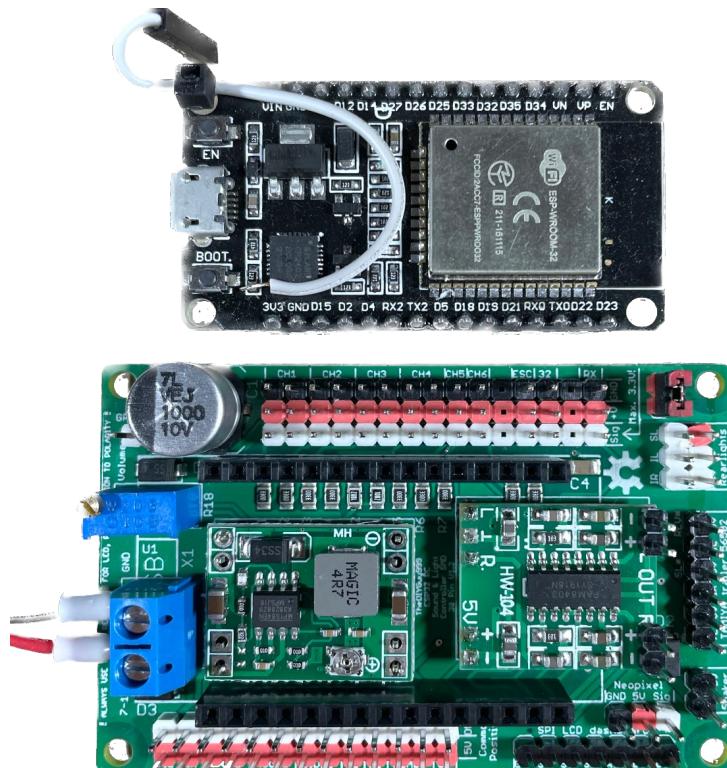
Videoserie:https://www.youtube.com/playlist?list=PLGO5EJClJBCJlvu8frS7LrEU3H2Yz_so

Die Änderungen finden Sie in der [Änderungsprotokoll](#). Die

Diskussions- und Supportthread (auf Deutsch und Englisch): <https://www.rc-modellbau-portal.de/index.php?threads/esp32-arduino-rc-sound-und-lichtcontroller.7183/>

Vollständig montierte, getestete und funktionsfähige 30-Pin-SMD-Version mit Schaltnetzteil und Displayanschluss. Hier bestellen:

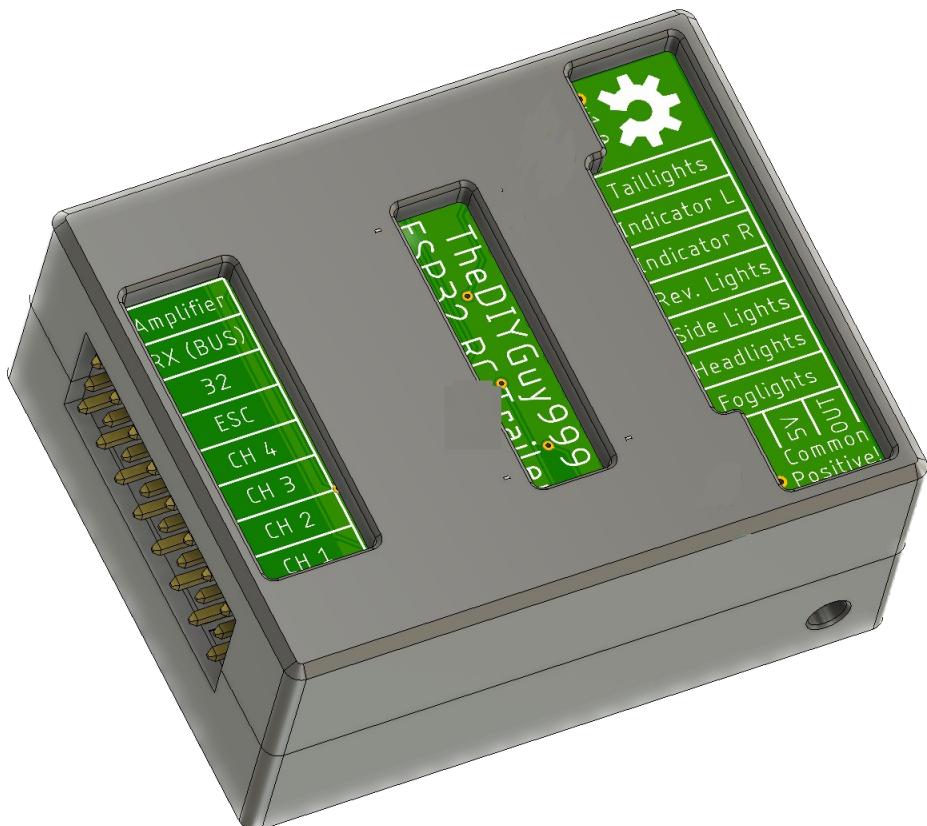
https://www.pcbway.com/project/shareproject/Arduino_RC_engine_sound_light_controller_for_ESP32_a9334731.html



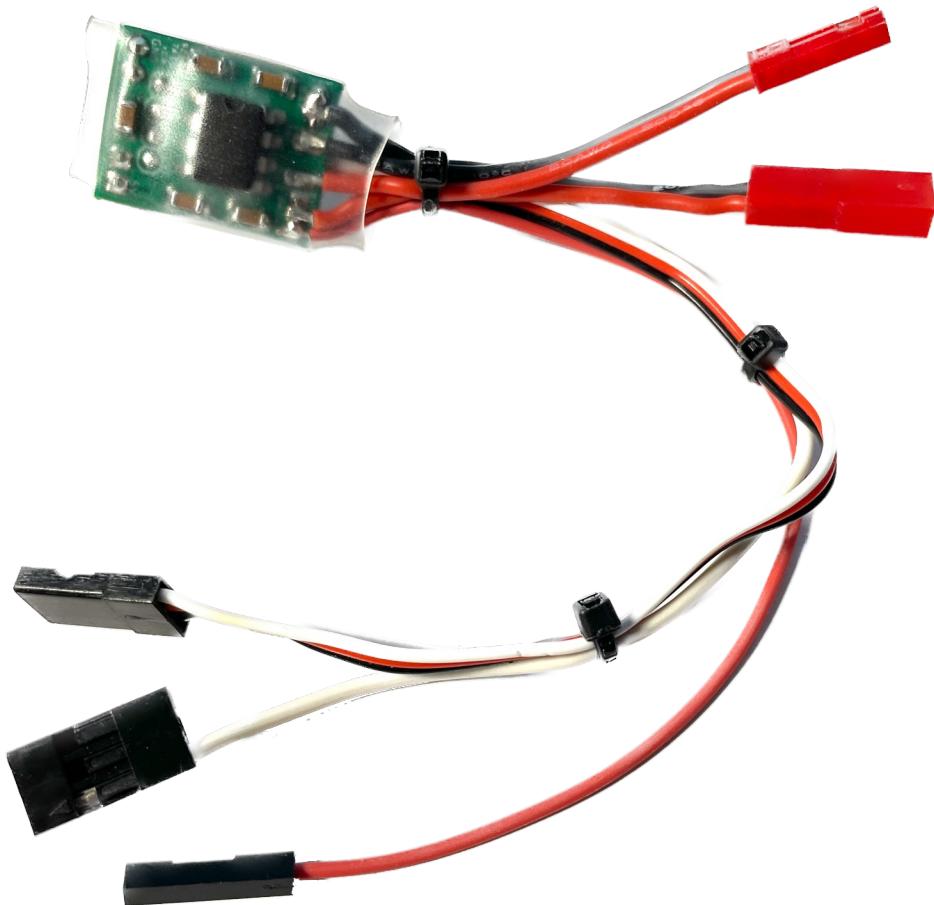
Neu: STL-Daten für ein ansprechendes 3D-gedrucktes Gehäuse verfügbar: https://thediyguy999.github.io/TheDIYGuy999_ESP32_Web_Flasher/products.html



Neu: Kompakte Version für Anhänger oder Traktor: https://thediyguy999.github.io/TheDIYGuy999_ESP32_Web_Flasher/products.html



Neu: RZ7886 7A ESC: https://thediyguy999.github.io/TheDIYGuy999_ESP32_Web_Flasher/products.html



Battery voltage: 7.34 V

Number of cells: 2 (2S battery)

Battery cutoff voltage: 6.60 V

WiFi settings

Wireless trailer settings

ESC settings

Servo settings

Light settings

Flickering while cranking Xenon simulation

Swap L & R indicators Indicators as sidemarkers

LED indicators Separate full beam (roof light connector)

No cab lights No fog lights

Flashing blue lights

Cab light brightness: 100



Side light brightness: 150



Reversing light brightness: 140



Fog light brightness: 200



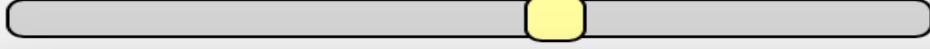
Tail light dimmed brightness (while not braking, use low value in Indicators as sidemarkers mode): 30



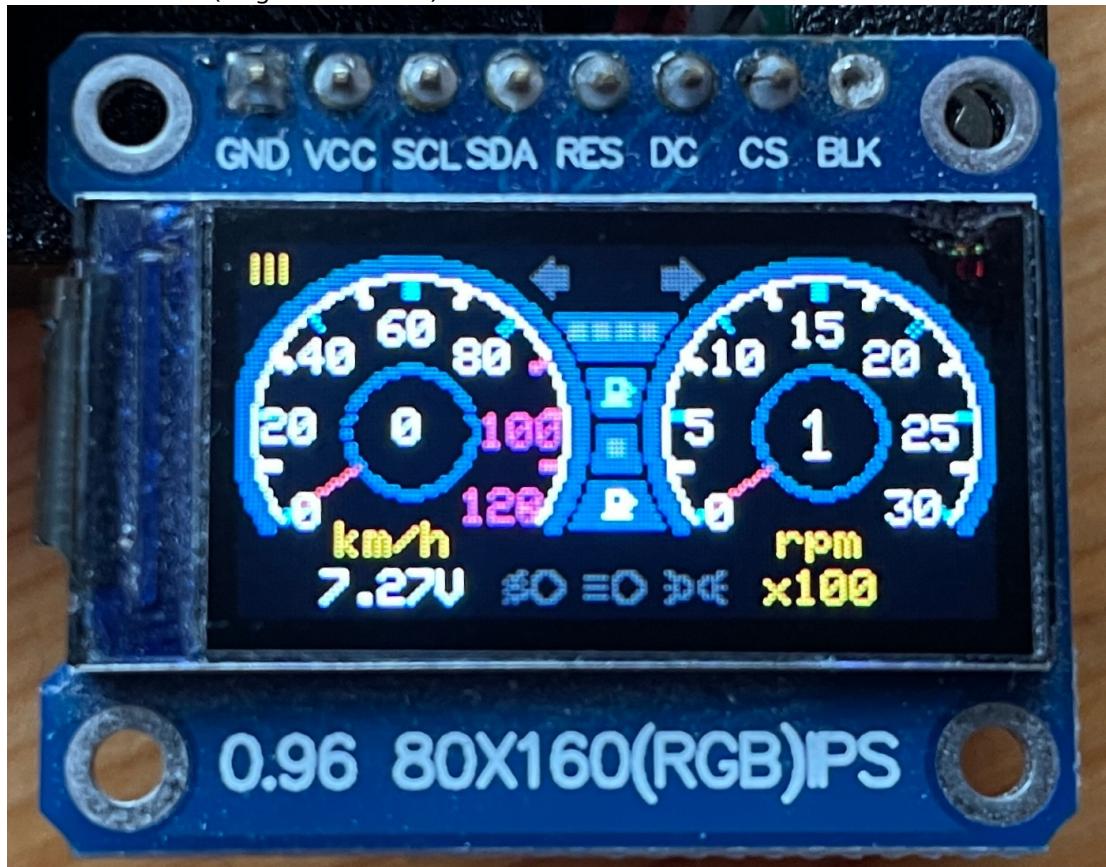
Tail light dimmed brightness (while parking lights only): 3



Head light dimmed brightness (while parking lights only): 3

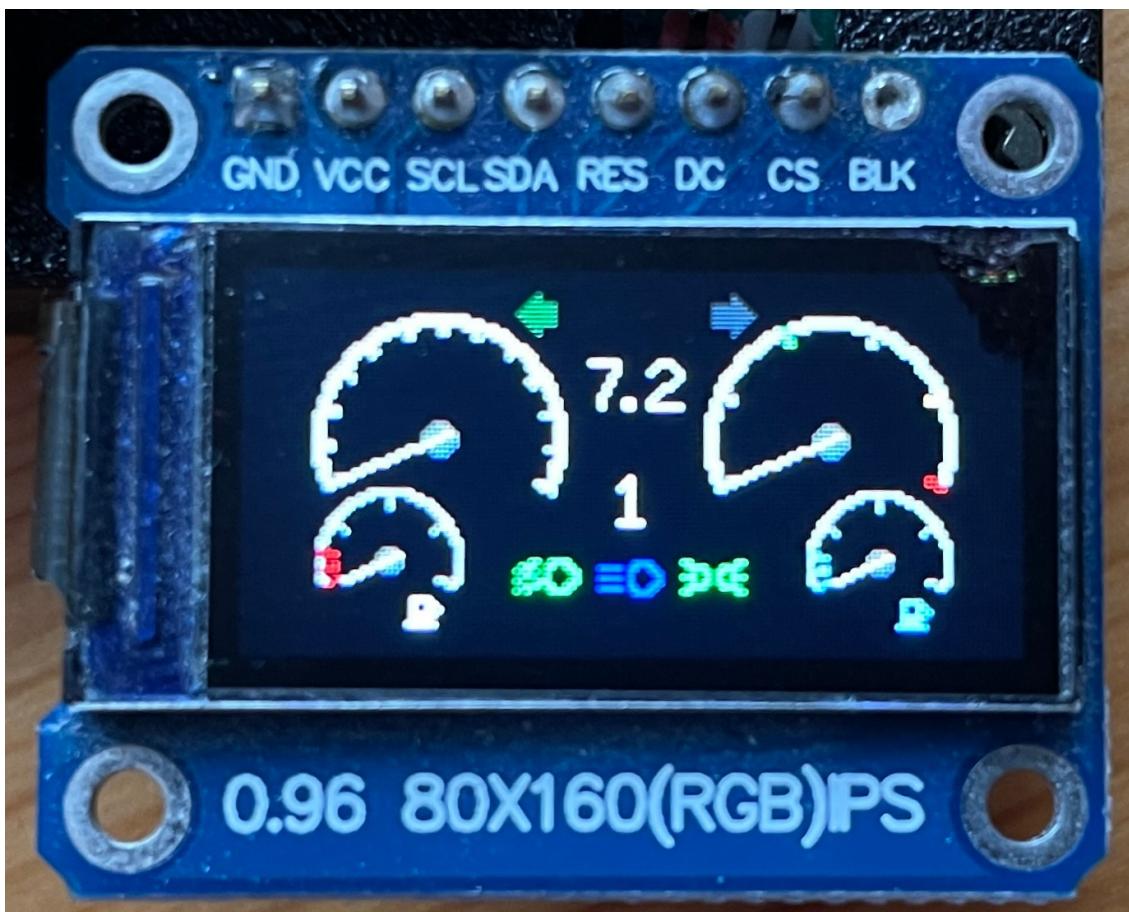


Neu: LCD-Armaturenbrett (Original von Frevic)

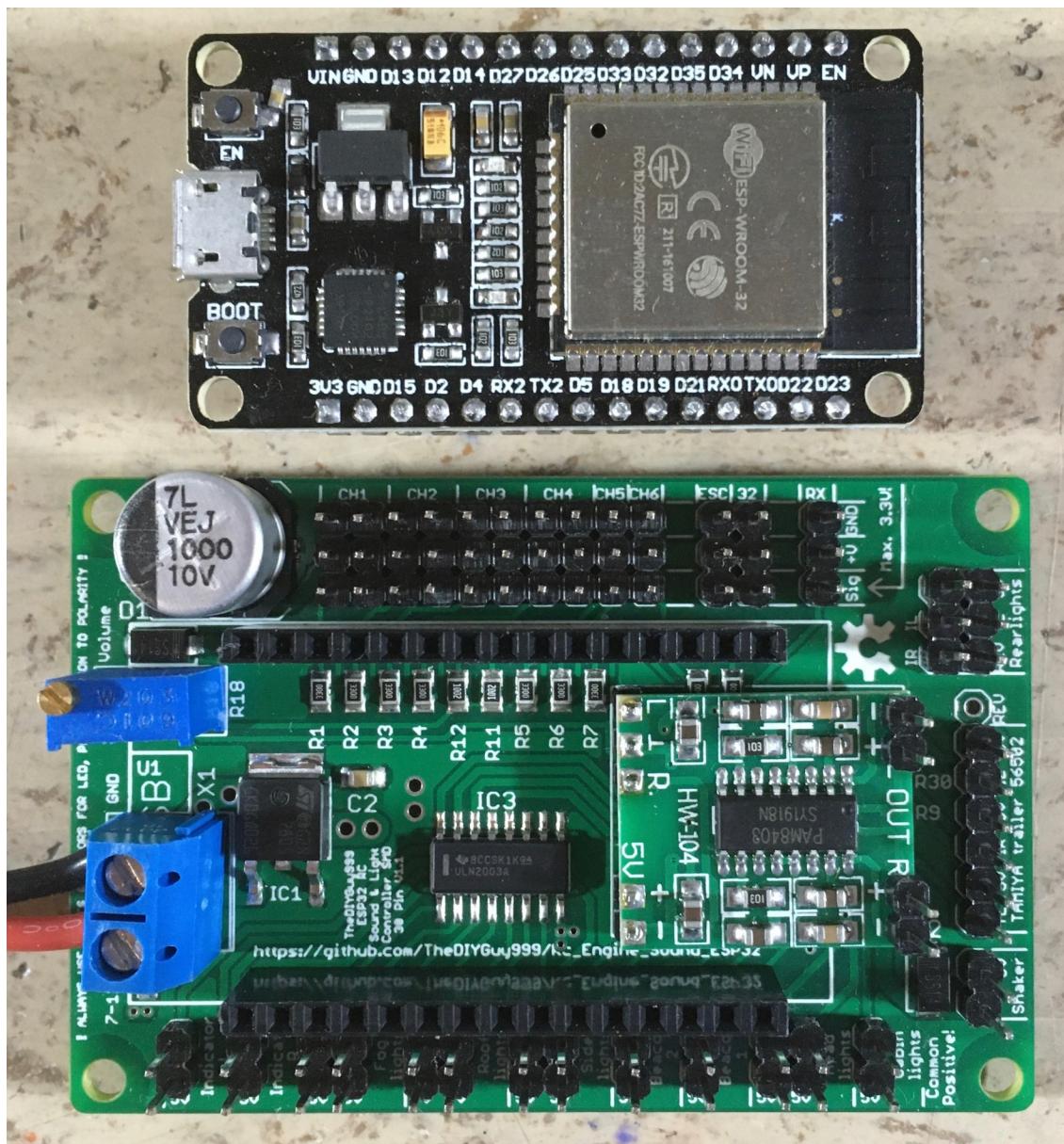


<https://www.facebook.com/profile.php?id=100066616574355>

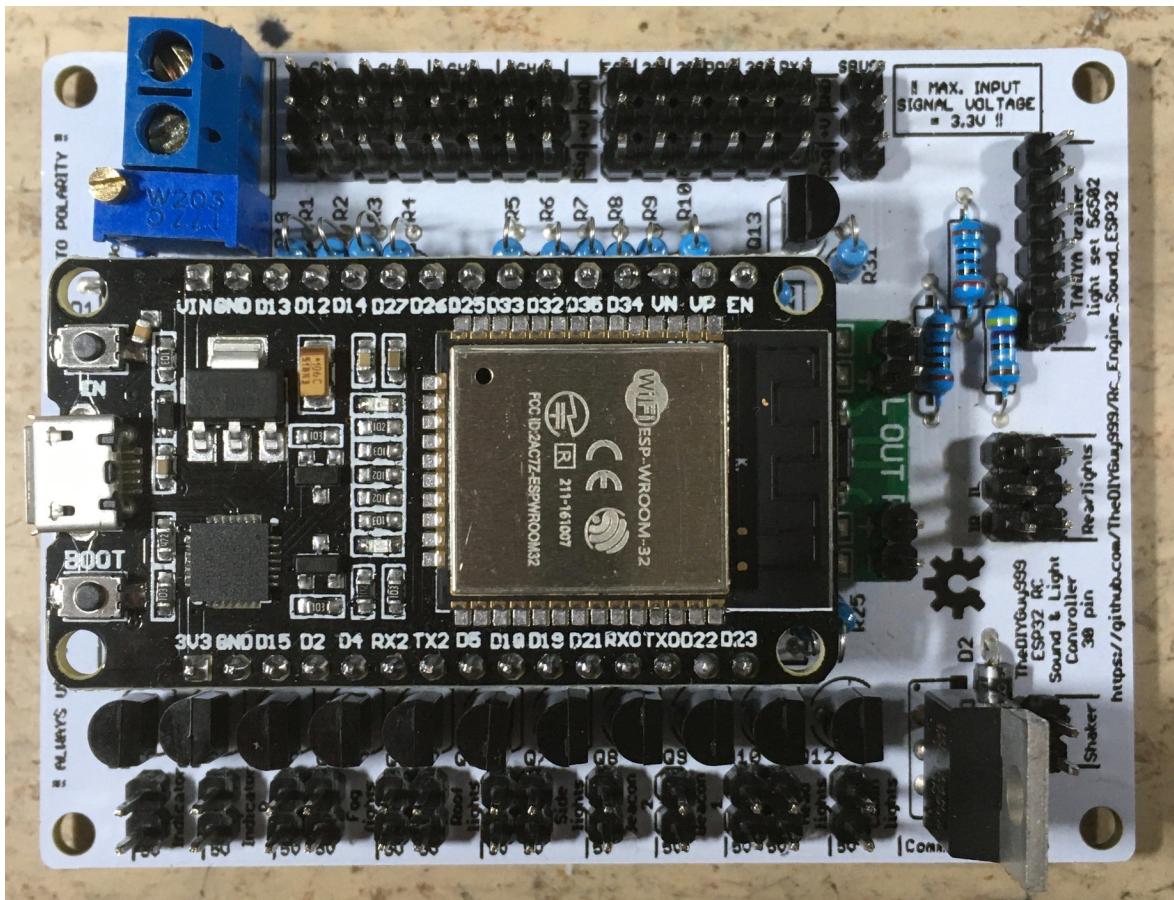
LCD-Armaturenbrett (Original von Gamadril)



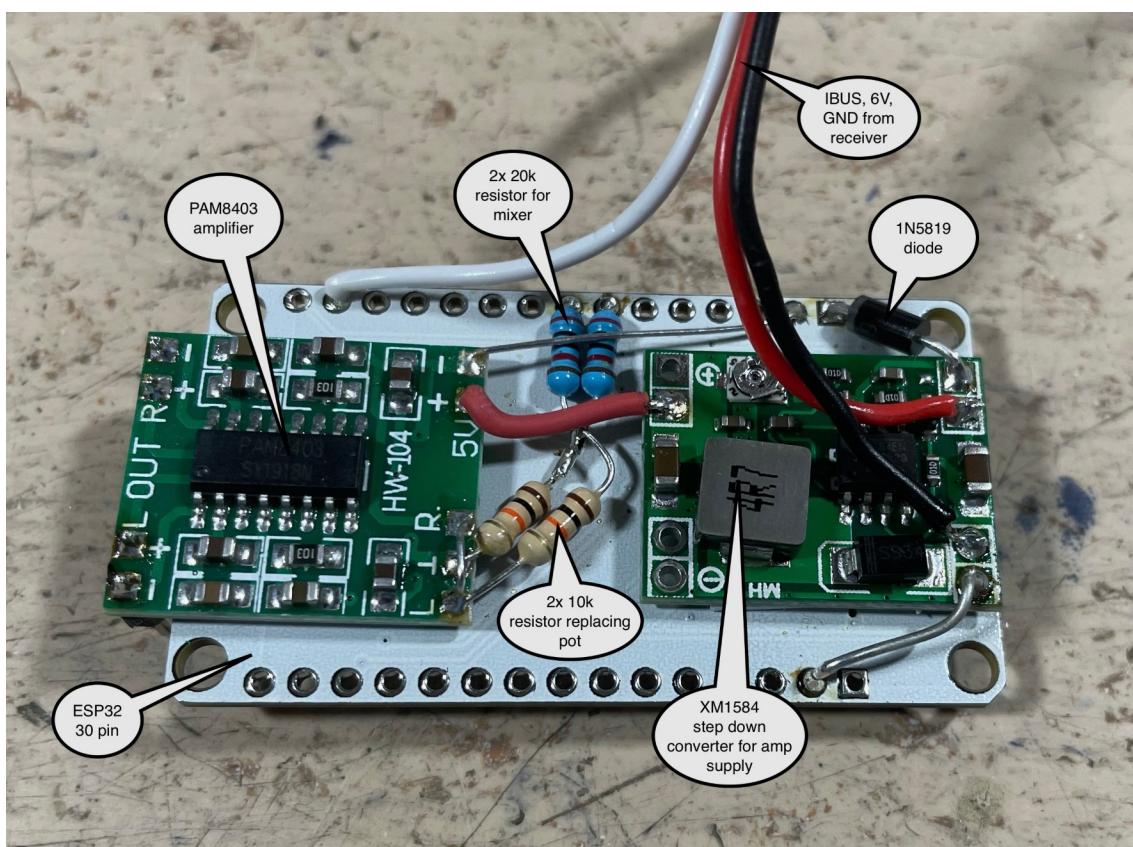
Vollständig montierte, getestete und funktionsfähige 30-Pin-SMD-Version, hergestellt und vormontiert von
<https://www.pcbway.com>



Vollständig montiert, getestet und funktionsfähig, 30-polige Durchgangslochversion



Kompakte Version für Bagger (nur IBUS & Sound, Stromversorgung über 6V BEC)





Merkmale:

- Einzigartige Simulation der Fahrzeugmasse (schließen Sie Ihren Crawler-Regler an Pin 33 an). Die Gasannahme wird beim Schalten eines mechanischen 3-Gang-Getriebes angepasst, um sanfte Schaltvorgänge, Getriebeschutz und realistische Geräusche zu gewährleisten. Funktioniert einwandfrei mit TAMIYA 3-Gang-Getrieben. Sollte auch mit Crawler-2-Gang-Getrieben funktionieren. Der Regler wird über einen Zustandsautomaten mit folgenden Zuständen gesteuert: Vorwärts- und Rückwärtsgang (variable Beschleunigung abhängig von der Gashebelstellung), Neutralstellung, Bremsen vorwärts und rückwärts (variable Verzögerung mit feiner Abstufung, abhängig von der Gashebelstellung). Er ermöglicht außerdem die präzise Steuerung von Bremslichtern, Bremsgeräusch, Rückfahrscheinwerfer und Rückfahrtsignal. Beschleunigung und Verzögerung (Ausrollen und Bremsen) sind für jeden Gang separat einstellbar, um maximalen Realismus zu gewährleisten.
- Die einzigartige „virtuelle Kupplung“ ermöglicht es, den Motor unterhalb einer einstellbaren ESC-Ausgangsdrehzahl hochzudrehen. Darüber hinaus greift die Kupplung ein und sorgt dafür, dass der Motorklang mit der Raddrehzahl synchronisiert ist. Klingt und fährt sich in Kombination mit einem Schaltgetriebe einfach hervorragend!
- Simuliertes Automatikgetriebe mit Drehmomentwandler (falls Ihr Fahrzeug kein echtes Schaltgetriebe besitzt)
- Simuliertes Doppelkupplungsgetriebe
- simuliertes, manuell geschaltetes 3-Gang-Getriebe (neu in Version 5.5).
- **Virtueller, schaltbarer Leerlauf ermöglicht das Gasgeben im Stand**
- Jake-Bremse (simulierte pneumatische Motorbremse, hauptsächlich in US-amerikanischen Lkw verwendet)
- Kettenfahrmodus (Doppelgaseingänge an Kanal 2 und 3, für Panzer, Bagger usw.). In diesem Modus wird keine Reglersteuerung unterstützt. (Neu in Version 4.5)
- Panzerkanonengeräusch & Blitz (Neu in Version 4.6)
- Auslösen mehrerer verriegelnder und nicht verriegelnder Aktionen (Töne, Lichter) pro analogem Kanal mithilfe der rcTrigger-Bibliothek (Neu in Version 4.7, noch experimentell)
- Zahlreiche wählbare Geräusche: Anlassen des Motors, Leerlauf, Aufheulen des Motors, Turbopfeifen, Dieselzündungsklopfen, Wastegate-Ventil, Hupen, Sirenen, Rückfahrpiepen, Druckluftbremse, Feststellbremse, Gangschaltung usw.
- Realistischer Motorsound wird dynamisch aus bis zu vier verschiedenen Geräuschen gemischt: Leerlauf, Turbo, Wastegate (alle mit variabler Abtastrate), Dieselzündungsklopfen (feste Abtastrate, daher keine Tonhöhenänderung).
- Lastabhängige Geräusche (Drosselklappenstellung): Leerlauf, Drehzahl, Dieselklopfen
- **Drehzahlabhängige Geräusche: Turbo, Wastegate**
- Dutzende Motoren- und andere Geräusche sind enthalten; Sie können auch eigene Geräusche mit Audacity und dem Bitlunis-Konvertierungstool (Link oben) erstellen.
- Motordrehzahlbereich und Trägheitsmoment einstellbar, Lautstärke aller Geräusche einstellbar, Motorgeräusche separat für Last und Leerlauf.
- Viele weitere Parameter können angepasst werden. Alle Einstellungen sind in der Datei „adjustmentsXyz.h“ leicht zugänglich.
- Es können Audiodateien bis zu 22.050 Hz, 8 Bit, Mono verwendet werden.
- Kompatible Eingangssignale: PWM, PPM, SBUS (invertierte und nicht invertierte Signale), IBUS
- Funktioniert am besten mit einem PAM8403-Verstärkermodul, das über 10-kOhm-Widerstände und ein 10-kOhm-Potentiometer an die Pins 25 und 26 angeschlossen ist (siehe Schaltplan unten).
- Die Motordrehzahl wird anhand des RC-Signals an Pin 13 berechnet. *** VORSICHT, max. 3,3 V an allen Pins! *** 330-Ohm-Widerstände an allen I/O-Pins empfohlen!

- Nichtlineare Drosselklappenkennlinien können in "curves.h" generiert werden.
- **Lichteffekte:** Scheinwerfer (Fern- und Abblendlicht), Rücklichter, Bremslichter, Nebelscheinwerfer, Dachleuchten, Kabinenleuchten, Rückfahrscheinwerfer, Blinker, Warnblinker, Blaulicht usw. (max. 12 Ausgänge)
- Motorschwingungssimulation mittels eines Schwingmotors mit exzentrischem Gewicht: Starke Vibration beim Anlassen, mittlere im Leerlauf, geringe beim Gasgeben.
- Lautstärke einstellbar (per Fernbedienung)
- Verwenden Sie einen ESP32, die CPU-Frequenz muss auf 240 MHz eingestellt sein.
- Eagle-Schaltplan und Platinendatei enthalten. Vorgefertigte Gerber-Dateien ermöglichen Ihnen eine einfache Bestellung Ihrer Platine.

- Inklusive, einfach zu bedienender .wav-zu-.h-Audiodateikonverter
- Kanäle können einfach über "remoteSetup.h" zugewiesen werden.
- **Vorkonfigurierte Profile für Flysky FS-i6X und Arduino Micro RC-Fernbedienung (neu in v.5.5)**
- Variable Länge für Horn und Sirene, Verwendung des Schleifenbereichs in den Audiodateien (neu in Version 5.6).
- **BUS-Decoder für Lenkservo und Schalt servo (Servos an CH1 & CH2 anschließen)**
- Der Servo für die Sattelkupplung (5th Wheel) kann an die CH4-Pins angeschlossen werden (nicht im PWM-Kommunikationsmodus).
- Der TAMIYA-Anhängerpräsenzs schalter kann an Pin 32 angeschlossen werden (abhängig von der Einstellung "#define THIRD_BRAKELIGHT" im Tab "6_adjustmentsLights.h").
- Unterstützung für nichtlineare Gas- und Lenkkurven (für eine präzisere Steuerung um die Mittelstellung). Verwenden Sie „EXPONENTIAL_THROTTLE“ und „EXPONENTIAL_STEERING“ in „2_adjustmentsRemote.h“.
- Unterstützung für die HOBBYWING Quicrun Fusion Motor-/Regler-Kombination. Verwenden Sie "#define QUICRUN_FUSION" in "3_adjustmentsESC.h"
- Unterstützung für eine Seilwinde, angeschlossen an Kanal 3 (nur im BUS-Kommunikationsmodus). Verwenden Sie "#define MODE2_WINCH" in "7_adjustmentsServo.h". Mit der Taste "Modus 2" kann dann zwischen Hupen-/Sirenesteuerung und Seilwindensteuerung über Kanal 4 umgeschaltet werden. Die Seilwinde wird von einer älteren RC-Servotreiberplatine angesteuert. Geschwindigkeit und Neutralstellung erfolgen über die Positionen "CH3L", "CH3C" und "CH3R".

- Unterstützung für LCD-Dashboard
- Unterstützung für 2812 Neopixel LED (GPIO0)
- Unterstützung für Hydraulikbagger (Hydraulikpumpe, Hydraulikfluss, Kettenklappergeräusche). Verwenden Sie das Profil #define FLYSKY_FS_I6S_EXCAVATOR für die Fernsteuerung.
- ESP-NOW-basierte 2,4-GHz-Funkanhängersteuerungsunterstützung
- Anstelle eines Standard-RC-Reglers für Crawler kann ein RZ7886-Motortreiber-IC verwendet werden.
- Optionen zum Schutz vor tiefer Batterieentladung
- Umschaltbarer Kriechgang (mit minimaler virtueller Trägheit)
- Webseite zur WLAN-Konfiguration: 192.168.4.1

Auf der Aufgabenliste:

- Kurvenlicht (an den Rundumleuchtenausgängen)
- Warnblinker werden eingeschaltet, wenn der Motor im AUTO_LIGHTS-Modus abgestellt ist.

Bekannte Probleme:

- Die Arduino IDE Version 1.8.7 oder älter wird nicht unterstützt und führt zu Compilerfehlern!
- Der ESP32 funktioniert nicht unter macOS Big Sur 11.x, aber dieses Problem lässt sich wie hier beschrieben leicht beheben:[Big Sur Fix](#) (für Version 1.04)
- macOS Monterey 10.3 enthält kein Python 2 mehr. Für die Arduino IDE müssen Sie Python 3 installieren und den Pfad in der Datei pPlatform.txt entsprechend anpassen.
<https://forum.arduino.cc/t/mac-os-update-killed-esp32-sketch/969580/24>

So erstellen Sie neue .h-Sounddateien:

Kühnheit:

- Importiere die gewünschte WAV-Audiodatei in Audacity.
- Konvertiere es bei Bedarf in Mono.
- Unten links die Projektfrequenz 22.050 Hz auswählen.
- Suchen Sie im Leerlaufgeräusch nach einem zyklischen Muster (die Anzahl der Zündimpulse entspricht üblicherweise der Zylindernummer), schneiden Sie die Leerlaufprobe exakt auf diese Länge zu und achten Sie genau auf die Nulldurchgänge, um Klickgeräusche zu vermeiden. Der lauteste Peak sollte sich stets am Ende der Probe befinden.
- Verfahren Sie genauso mit dem „Rev“-Sound. Er wird je nach Motor und Drehzahl des „Rev“-Sounds 2- bis 4-mal kürzer sein als der „Idle“-Sound.
- Ändern Sie die „Rate“ (Dropdown-Menü links neben dem Sample) des „rev“-Samples, bis die Länge der des „idle“-Samples entspricht. Dies ist sehr wichtig!
- Duplizieren Sie einen Teil der „Rev“-Probe (diejenige mit der ursprünglichen, unveränderten „Rate“-Drehzahl). Dies ist die „Klopft“-Probe. Kürzen Sie sie auf diese maximale Länge: Leerlauflänge / Anzahl der Zylinder / Drehzahlbereich „MAX_RPM_PERCENTAGE“ (üblicherweise 2-4 oder 200-400 %).
- Passen Sie die Lautstärke aller Samples so an, dass der gesamte Dynamikbereich genutzt wird.
- Sie können auch Hochpass- oder Tiefpassfilter anwenden, um den Klang feinabzustimmen.
- Auswählen > Audio exportieren > Ausgewähltes Audio > WAV > 8-Bit-PCM

Konvertieren Sie die .wav-Datei mit dem geänderten Konvertierungstool (neu in Version 5.2):

- Öffnen Sie den mitgelieferten Konverter „Audio2Header.html“ in Ihrem Browser. Er befindet sich im Ordner „tools“.
- Passen Sie das Exportdateiformat an (keine Änderungen erforderlich).
- Wählen Sie den Exportdateitype je nach Art des zu konvertierenden Tons (Leerlauf, Gas geben, Hupe usw.).
- Öffnen Sie die WAV-Datei, die Sie konvertieren möchten.
- Es wird eine .h-Datei generiert und heruntergeladen.
- Verschieben Sie es in Ihr Verzeichnis „sketch/vehicles/sounds“.

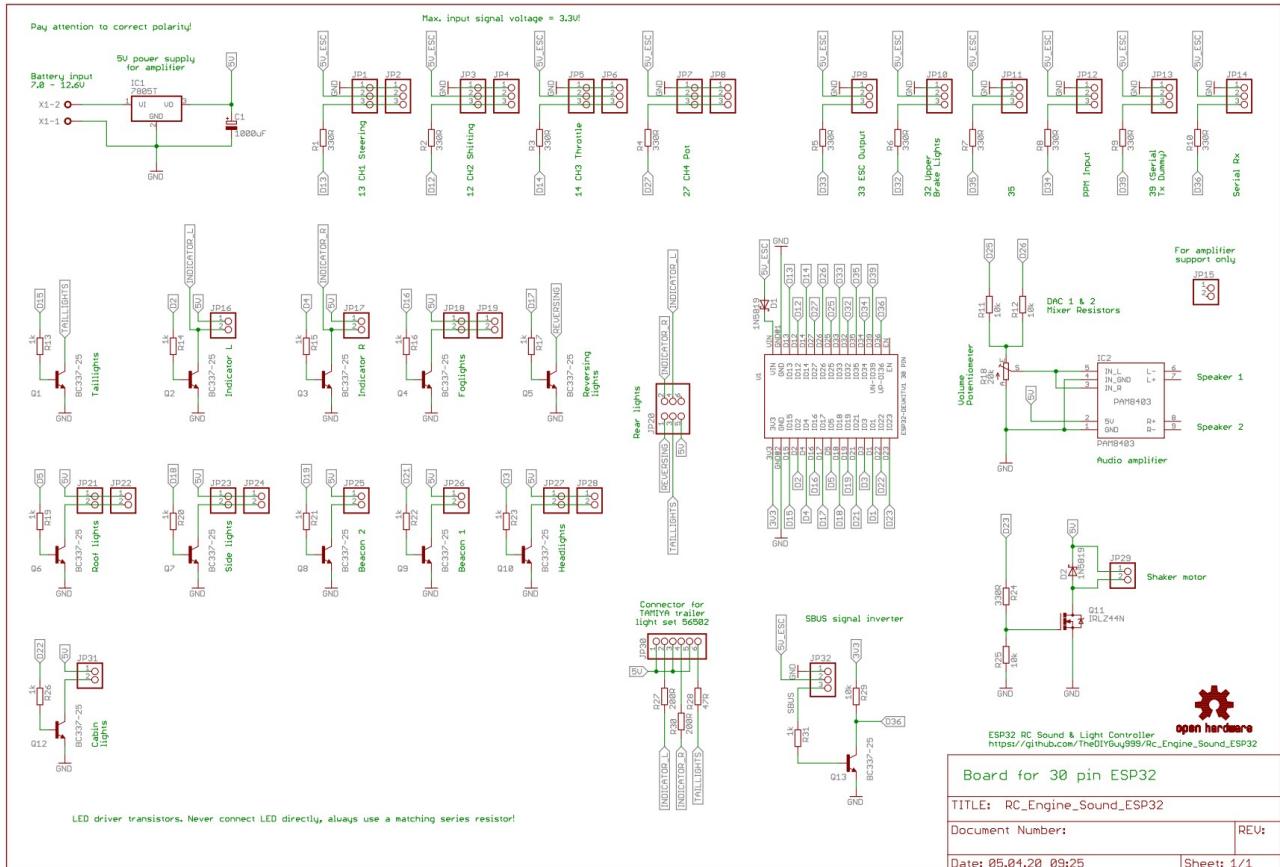
Fügen Sie die neue Header-Datei zusammen mit Ihrem Sound in Ihre Fahrzeuvoreinstellung ein und passen Sie die Einstellungen an, bis Sie zufrieden sind:

- Fügen Sie diese .h-Datei in "Adjustments.h" > "yourVehiclePreset.h" ein.
- Klopgeräuscheinstellungen:
 - "dieselKnockInterval = Anzahl der Zylinder
 - Entfernen Sie die Kommentarzeichen vor „V8“ für V8-Motoren, „R6“ für Reihensechszylinder-Motoren oder „V2“ für V2-Motoren (Harley).
 - Die Einstellung „dieselKnockAdaptiveVolumePercentage“ (das Verhältnis der Lautstärke der „leisen“ Klopfimpulse zu den lauten) ist nur aktiv, wenn „V8“, „R6“ oder „V2“ definiert ist.
- Experimentieren Sie mit den anderen Lautstärken, Start-, End- und Schaltpunkten, bis Sie zufrieden sind.
- Der "rev"-Sound ist optional und nur aktiv, wenn "REV_SOUND" definiert ist (// entfernt).
- Passen Sie den Übergang vom Leerlauf- zum Drehzahl-Sound mithilfe von „revSwitchPoint“, „idleEndPoint“ und „idleVolumeProportionPercentage“ an. Dieser Schritt ist sehr wichtig und kann einen großen Unterschied ausmachen!

Neue Skizze kompilieren:

- Kompilieren und laden Sie den Sketch in der Arduino IDE hoch.
- Der neue Sound sollte jetzt fertig sein.

Schematisches Beispiel (die aktuelle Version finden Sie im PDF!):



Leiterplatte

Enthaltene Leiterplattendateien:

Siehe Hardware-Ordner

Empfohlener Hersteller:

<https://www.pcbway.com> (Inklusive SMD-Bestückungsservice; verwenden Sie Gerbers.zip für die Platine; diese enthält auch BOM.xlsx und CPL.xlsx, falls Sie den SMT-Bestückungsservice nutzen möchten.) Informationen zur Bestellung vorbestückter Platinen finden Sie unter /Eagle_PCB/How To Order Your PCB.pdf

Die einfachste Möglichkeit, die 30-polige SMD-Version mit Displayanschluss zu bestellen:

<https://www.pcbway.com/project/shareproject/>
[Arduino_RC_engine_sound_light_controller_for_ESP32_a9334731.html](https://www.pcbway.com/project/shareproject/Arduino_RC_engine_sound_light_controller_for_ESP32_a9334731.html)

Montageanleitung (für die 30-polige SMD-Version ohne Displayanschluss):

<https://www.youtube.com/watch?v=csQgTfxRd8Y&t=2s>

Montageanleitung (für die 36-polige Version):

https://www.youtube.com/watch?v=Vfaz3CzecG4&list=PLGO5EJJClJBCjIvu8frS7LrEU3H2Yz_so&index=13

Verdrahtung:

- Dieses Gerät ist nicht gegen Verpolung geschützt!
- Verwenden Sie für LED-Stecker immer Vorwiderstände (außer für TAMIYA-Anhängerstecker).
- Maximale Eingangsspannung an den "Sig"-Pins = 3,3 V (Vorsicht bei sehr alten Empfängern, die möglicherweise 5 V liefern)
- Es wird empfohlen, eine Sicherung zwischen Batterie und Soundcontroller/ESC zu verwenden.

Stromversorgung für Audioverstärker, Vibrationsmotor und LED:

Verwenden Sie ein Y-Kabel zwischen Akku, Regler und dem Anschluss „X1“. Der Akkuspannungsbereich liegt zwischen 7 und 12,6 V.

Stromversorgung für ESP32:

- Der ESP32 wird nicht über den Anschluss „X1“ versorgt.
- Die Stromversorgung kann über den Micro-USB-Anschluss erfolgen.
- oder über die +V- und GND-Pinreihe auf der Oberseite der Platine (die Spannung kommt normalerweise vom BEC in Ihrem ESC, der an den "ESC"-Header angeschlossen werden muss).

ESC-Verkabelung:

- Schließen Sie einen Hobbywing 1080 ESC an den ESC-Header (GND, V+ und Sig) an.
- Passen Sie die ESC-Parameter mithilfe der Programmierkarte an, wie oben in „Adjustments.h“ beschrieben.
- Ich empfehle keinen anderen Regler.
- Der Regler wird vom Zähler gesteuert, nicht direkt vom Empfänger. Dadurch kann die einzigartige Funktion der „virtuellen Trägheit“ genutzt werden. HINWEIS: Die Nutzung dieser Funktion erfolgt auf eigene Gefahr! Ich übernehme keine Haftung für eventuelle Schäden. Das System läuft sehr stabil und ich hatte noch nie Probleme, aber man weiß ja nie.
- "Mit „escPulseSpan“ lässt sich die Höchstgeschwindigkeit Ihres Fahrzeugs begrenzen. 500 = keine Begrenzung, Werte über 500 begrenzen die Höchstgeschwindigkeit.

Empfängerverdrahtung für PWM-Servosignale (die gebräuchlichste Verdrahtung):

- Die Kanalzuordnung erfolgt gemäß „remoteSetup.h“ und „remoteSetup.xlsx“ und ist einfach anpassbar (neu in Version 5.5). Es ist wichtig, die Kabel entsprechend der Kanalzuordnung anzuschließen.
- Die Kanäle 5 und 6 werden über die "35"- und "PPM"-Anschlüsse empfangen.
- Mindestens ein Kanal (CH) muss mit einem 3-poligen Kabel angeschlossen werden, sodass GND und V+ (Empfängerversorgung) ebenfalls angeschlossen sind.
- Die CH1-4-Stifteleisten sind paarweise parallel geschaltet. Dadurch können Servosignale durchgeleitet werden, wodurch Y-Kabel überflüssig werden.
- Beachten Sie, dass Sie die Konfiguration wie unten beschrieben ändern müssen, wenn Sie diese Verdrahtungsmethode verwenden möchten.

Empfängerverdrahtung für PPM-Signale:

- Interne Kanalzuordnung wie oben
- Verbinden Sie ein 3-poliges Kabel vom PPM-Anschluss Ihres Receivers mit dem RX-Anschluss (geändert in Version 5.5, vorher PPM) des Soundcontrollers (Sig, V+, GND).
- Beachten Sie, dass Sie die Konfiguration wie unten beschrieben ändern müssen, wenn Sie diese Verdrahtungsmethode verwenden möchten.
- In diesem Modus können 8 Kanäle ausgelesen werden.

Empfängerverdrahtung für SBUS-Signale (empfohlen):

- Interne Kanalzuordnung wie oben
- Verbinden Sie ein 3-poliges Kabel vom SBUS-Anschluss Ihres Receivers mit dem SBUS-Anschluss am Soundcontroller (Sig, V+, GND).
- Der "Sig"-Pin am SBUS-Header ist 5V-tolerant.
- In diesem Modus können 13 Kanäle ausgelesen werden.

Empfängerverdrahtung für IBUS-Signale:

- Interne Kanalzuordnung wie oben
- Verbinden Sie ein 3-poliges Kabel vom IBUS-Anschluss Ihres Receivers mit dem RX-Anschluss des Soundcontrollers (Sig, V+, GND).
- In diesem Modus können 13 Kanäle ausgelesen werden.

Sprecher

- 4- oder 8-Ohm-Lautsprecher sind kompatibel
- Sie können einen oder zwei Lautsprecher verwenden.
- Verwenden Sie niemals zwei Lautsprecher parallel an einem einzigen Lautsprecheranschluss.
- Verwenden Sie niemals zwei parallel geschaltete Header, um einen Lautsprecher anzusteuern.
- Schließen Sie Kondensatoren niemals an die Lautsprecheranschlüsse an.

LED

- Die LEDs müssen mit „gemeinsamer Pluspol“ verdrahtet werden. Das bedeutet, dass die langen Beinchen der LEDs alle miteinander verbunden und an die 5-V-Schiene des integrierten Spannungsreglers angeschlossen werden.
- Alle LEDs (außer denen, die an den TAMIYA-Anhängerstecker angeschlossen sind) benötigen einen Vorwiderstand.
- Berechnen Sie den erforderlichen Widerstand nach folgender Formel:<http://ledcalc.com> (Versorgungsspannung = 5 V)
- Es wird nicht empfohlen, LEDs parallel zu schalten und den Serienwiderstand gemeinsam zu nutzen.
- Unterstützung für WS2812 Neopixel LED (Details und Verdrahtung siehe "6_adjustmentsLights.h")

LCD-Armaturenbrett

Siehe "9_adjustmentsDashboard.h"

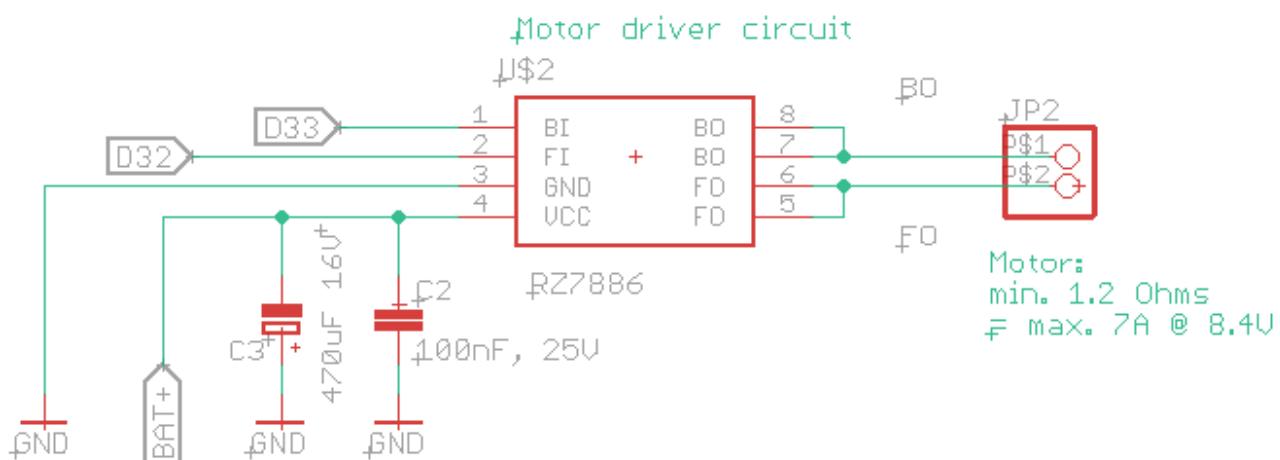
Shaker

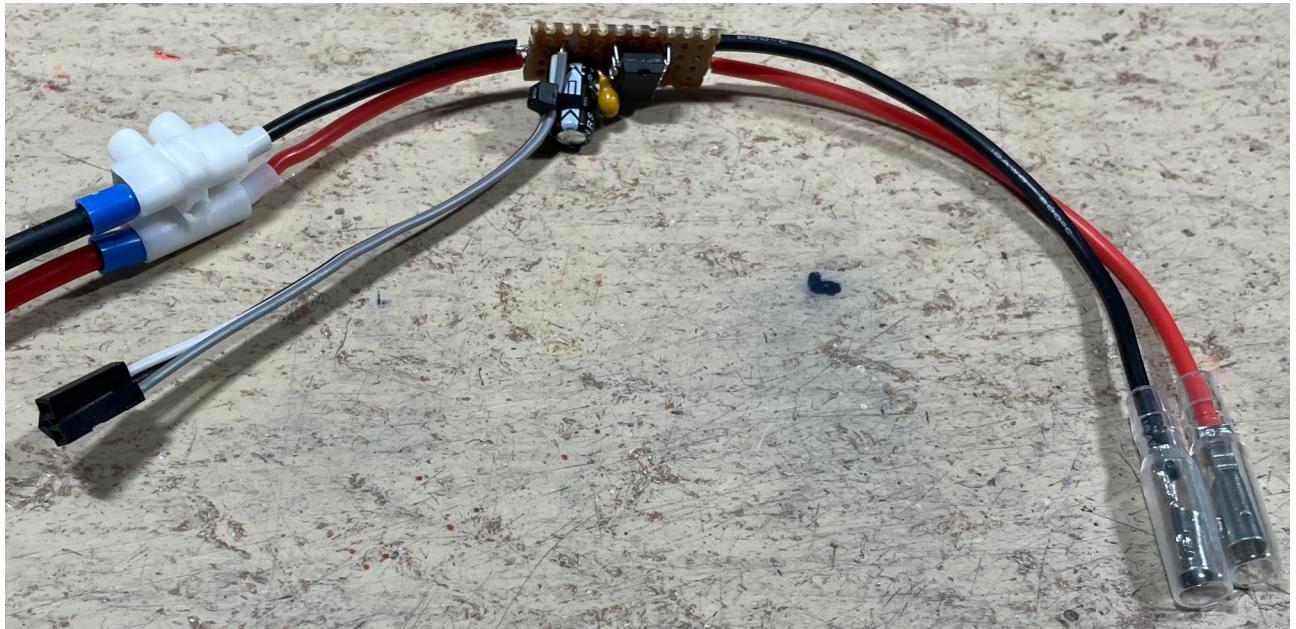
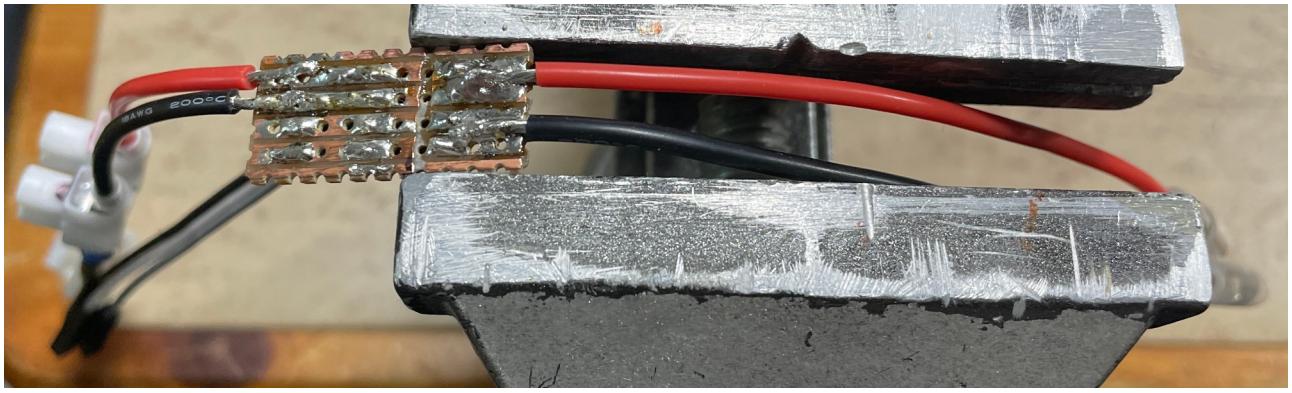
- Der Shaker dient zur Simulation von Motorschwingungen. Die Geschwindigkeit kann in der Fahrzeugkonfiguration angepasst werden und variiert je nach Motorzustand und Drehzahl.
- Es muss an den „Shaker“-Anschluss angeschlossen werden und wird vom integrierten 5V-Regler versorgt. Die negative Seite wird vom integrierten MOSFET geschaltet.
- Bitte beachten Sie, dass der verwendete MOSFET ein Logikpegel-MOSFET sein muss. Andernfalls funktioniert der Shaker nicht!
- Der Motor sollte nicht mehr als etwa 300 mA bei 5 V ziehen. Ich verwende einen Rüttelmotor von GT Power.

RZ7886 7A Gleichstrommotortreiber anstelle eines ESC:

Es ist für kleinere Fahrzeuge wie WPL gedacht.

Prototyp:





Bestellen Sie Ihre Motortreiberplatine hier:

[https://www.pcbway.com/project/shareproject/
RZ7886_based_ESC_for_ESP32_Sound_and_Light_Controller_f8f4a805.html](https://www.pcbway.com/project/shareproject/RZ7886_based_ESC_for_ESP32_Sound_and_Light_Controller_f8f4a805.html)

Software:

Erforderliche Software zum Hochladen und Bearbeiten von Code:

- Arduino IDE (nicht empfohlen):<https://www.arduino.cc/en/Main/Software>
- oder Visual Studio Code mit PlatformIO-Erweiterung (empfohlen):
<https://code.visualstudio.com>
- Visual Studio benötigt diese Software ebenfalls (VS Code muss anschließend neu gestartet werden), um Bibliotheken synchronisieren zu können:<https://git-scm.com/download>

Herunterladen und Vorbereiten des Codes mit der Arduino IDE:

- Laden Sie den Code hier herunter (klicken Sie auf „Code > ZIP herunterladen“):
https://github.com/TheDIYGuy999/Rc_Engine_Sound_ESP32
- Entpacken Sie den Ordner gegebenenfalls.
- Entfernen Sie den Teil "-master" aus dem Ordernamen.
- Installieren Sie zuerst die unten aufgeführten Bibliotheken und Boarddefinitionen und starten Sie die Arduino IDE neu.
- Öffnen Sie "scr.ino" mit einem Doppelklick (die Arduino IDE sollte starten).

Erforderliche ESP32-Boarddefinition (nicht erforderlich, wenn Visual Studio Code als IDE verwendet wird):

- Installieren Sie es gemäß:<https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/>
- Die Verwendung von Version 1.04, 1.05 oder neueren Versionen führt zu Neustartproblemen!

```
Board: "ESP32 Dev Module"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Flash Size: "4MB (32Mb)"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"
PSRAM: "Disabled"
Port: "/dev/cu.SLAB_USBtoUART"
Get Board Info
```

- Passen Sie die Einstellungen entsprechend an:

Erforderliche Bibliotheken. Sie müssen ALLE installieren, wenn Sie die Arduino IDE verwenden. Nicht erforderlich, wenn Visual Studio Code als IDE verwendet wird.

Status-LED:<https://github.com/TheDIYGuy999/statusLED>

- SBUS:<https://github.com/TheDIYGuy999/SBUS>
- rcTrigger:<https://github.com/TheDIYGuy999/rcTrigger>
- IBUS:<https://github.com/bmellink/IBusBM>
- TFT:https://github.com/Bodmer/TFT_eSPI
- FastLED:<https://github.com/FastLED/FastLED>

Laden Sie die Dateien auf die gleiche Weise wie den Hauptcode oben herunter. Speichern Sie die Ordner im Pfad „Arduino/libraries“. Installieren Sie sie wie folgt:<https://www.arduino.cc/en/Guide/Libraries>

Hochladen des Codes auf das Board:

- WICHTIG: Je nach Mainboard müssen Sie möglicherweise die „BOOT“-Taste gedrückt halten, wenn die IDE nur „Verbinden..._____“ anzeigt. Lassen Sie die Taste los, sobald der Upload beginnt.
- Unter macOS Big Sur muss folgende Korrektur angewendet werden (nur Arduino IDE):[Big Sur Fix](#)

Überblick über Visual Studio Code (anstelle der Arduino IDE):

The screenshot shows the Visual Studio Code interface with the following components:

- EXPLORER**: Shows the project structure for "RC_ENGINE_SOUND_ESP32". A callout points to ".pio" with the text "Platform IO is required".
- EDITOR**: Displays the "platformio.ini" file. A callout points to the [env:esp32dev] section with the text "Board definition".

```
[env:esp32dev]
platform = espressif32
board = esp32dev
framework = arduino
```
- EDITOR**: Displays the "src" directory containing multiple adjustment files (e.g., C_1_adjustmentsVehicle.h, C_2_adjustmentsRemote.h, etc.). A callout points to this area with the text "make your adjustments here".
- EDITOR**: Displays the main sketch file "src.ino". A callout points to this area with the text "main sketch".
- TOOLS**: Shows "Serial monitor", "compile", and "upload" buttons.
- EDITOR**: Displays the "platformio.ini" file again, highlighting the "lib_deps" section with the text "These libraries are installed automatically".

```
lib_deps =
  SPI
  https://github.com/TheDIYGuy999/statusLED
  https://github.com/TheDIYGuy999/SBUS
  https://github.com/TheDIYGuy999/rcTrigger
  https://github.com/bmellink/IBusBM
  https://github.com/Bodmer/TFT_eSPI
  https://github.com/FastLED/FastLED
```

Anpassungen in "X_Xyz.h":

Fahrzeugauswahl:

Beachten Sie, dass in Version 5.5 die frühere Konfigurationsdatei „Adjustments.h“ in mehrere Dateien aufgeteilt wurde.

Entfernen Sie die Kommentarzeichen (/) vor dem gewünschten Fahrzeug in der Datei „Adjustments.h“. Entfernen Sie niemals die Kommentarzeichen vor mehr als einem Fahrzeug! Beachten Sie, dass Sie den Code nach der Änderung der Einstellungen erneut hochladen müssen. Andernfalls werden die Änderungen nicht wirksam. Wählen Sie „Skizze“ > „Hochladen“, um den Code hochzuladen. Wichtig: Heben Sie Ihr Fahrzeug beim Hochladen immer vom Boden ab.

Um ein neues Fahrzeug zu erstellen, kopieren Sie die Datei vehicles/00_Master.h und speichern Sie sie unter dem Namen Ihres Fahrzeugs. Bearbeiten Sie anschließend die Einstellungen nach Ihren Wünschen und fügen Sie Verknüpfungen zu den gewünschten Sounddateien hinzu. Fügen Sie danach eine Verknüpfung zu Ihrer vehicle.h-Datei hinzu (siehe Beispiele unten) und entfernen Sie die Kommentarzeichen.

```

#include <Arduino.h>

// FAHRZEUGEINSTELLUNGEN
***** // Wählen Sie die gewünschte Fahrzeugvoreinstellung aus (entfernen Sie die Kommentarzeichen vor der gewünschten Voreinstellung, entfernen Sie //, niemals mehr als eine)

// Master -----
##include "vehicles/00_Master.h" // Diese Master-Preset-Datei enthält alle verfügbaren Sounddateien, die in bestehenden Fahrzeug-Presets nicht verwendet werden.

// US-Trucks -----
##include "vehicles/CaboverCAT3408.h" // Frontlenker-Lkw mit Caterpillar 3408 V8 Dieselmotor ##include "vehicles/PeterbiltDetroit8v92.h" // Peterbilt 359 mit Detroit 8V92 V8 Zweitakt-Dieselmotor ##include "vehicles/KenworthW900ADetroit8V71.h" // Kenworth W900A mit Detroit 8V71 V8 Zweitakt-Dieselmotor

##include "vehicles/KenworthW900ACAT3408.h" // Kenworth W900A mit Caterpillar 3408 V8 Diesel (King Hauler)
##include "vehicles/CAT3408OpenPipes.h" // Kenworth W900A mit Caterpillar 3408 V8 Dieselmotor und offenen Röhren
##include "vehicles/KenworthW900ACAT3408new.h" // Kenworth W900A mit Caterpillar 3408 V8 Dieselmotor (guter Basslautsprecher erforderlich)
##include "vehicles/KenworthCummins335.h" // Kenworth aus den 1950er Jahren mit Cummins 335 R6 Dieselmotor ##include "vehicles/MackSuperLiner.h" // Mack Super Liner
##include "vehicles/M35.h" // AM General M35 "Deuce and a half" Militär-Lkw, turbogeladener R6 Universal-Kraftstoffmotor
##include "vehicles/US_Firetruck.h" // US-Feuerwehrfahrzeug mit CAT3408 V8 und Allison 6-Gang-Automatikgetriebe (mit Hupen- und Sirenenschleifenfunktion)
##include "vehicles/FreightlinerCummins350.h" // Freightliner Frontlenker mit Cummins 350 R6 Dieselmotor

// EU-Lkw -----
##include "vehicles/Tatra813.h" // Tatra 813 8x8 V12 Diesel Militär-Lkw (alte Version zum Vergleich, nicht verwenden)
##include "vehicles/Tatra813new.h" // Tatra 813 8x8 V12 Diesel Militär-Lkw ##include "vehicles/UnimogU1000.h" // Unimog U 1000 mit turbogeladenem R6 Dieselmotor inkl. Feuerwehr-Martinshorn
##include "vehicles/MercedesActros1836.h" // Mercedes Actros 1863 oder 3363 Lkw mit R6 Diesel //
##include "vehicles/MercedesActrosV6.h" // Mercedes Actros V6 Renntruck inkl. quietschender Reifen //
##include "vehicles/ScaniaV8_50ton.h" // SCANIA V8 50-Tonnen-Lkw. Unbekanntes Modell. Schlechte Qualität! ##include "vehicles/ScaniaV8.h" // SCANIA V8 Lkw, unbekanntes Modell
##include "vehicles/1000HpScaniaV8.h" // 1000 PS starker Scania V8-Lkw mit offenen Auspuffrohren. Wahnsinniger Sound! Gute Basslautsprecher erforderlich.
##include "vehicles/Scania143.h" // SCANIA 143 V8 – die Legende! Meiner Meinung nach der am besten klingende.

```

```
##include "vehicles/ScaniaV8Firetruck.h" // SCANIA V8 Feuerwehrfahrzeug, automatisches Allison 6-Gang-  
Getriebe mit Drehmomentwandler, Martinshorn-Sirene  
##include "vehicles/VolvoFH16_750.h" // Volvo FH16 750 Lkw. Reihensechszylinder, 750 PS, offene Auspuffanlage! //  
#include "vehicles/VolvoFH16_OpenPipe.h" // Volvo FH Lkw. Reihensechszylinder, offene Auspuffanlage, alternative  
Version  
##include "vehicles/ManTgx.h" // MAN TGX 680 V8 LKW  
##include "vehicles/ManKat.h" // MAN KAT V8 Diesel, Lkw der Bundeswehr ##include  
"vehicles/MagirusDeutz256.h" // Magirus Deutz 256, luftgekühlter V8-Diesel-Lkw ##include  
"vehicles/MagirusMercur125.h" // Magirus Mercur, luftgekühlter V6-Diesel-Lkw ##include  
"vehicles/Saurer2DM.h" // Saurer 2DM R6, Schweizer Diesel-Lkw  
  
// Russische Lastwagen -----  
##include "vehicles/Ural4320.h" // URAL 4320 6x6 V8 Diesel Militär-Lkw  
# include "vehicles/Ural375D.h" // URAL 375D 6x6 V8 Benzin-Militär-Lkw  
##include "vehicles/URAL375.h" // URAL 375D 6x6 V8 Benzin-Militär-Lkw (neue Version mit besserem V8-  
Sound, aber guter Basslautsprecher erforderlich)  
##include "vehicles/GAZ66.h" // GAZ-66 V8 Benzin-Militär-Lkw  
  
// Russische Panzer -----  
##include "vehicles/IS3.h" // IS-3 Kampfpanzer aus dem Zweiten Weltkrieg, V12-Dieselmotor (Dual-ESC-Modus, guter Basslautsprecher  
erforderlich)  
  
// Traktoren -----  
##include "vehicles/KirovetsK700.h" // Russischer Kirovets K700 Monstertraktor. Extremer  
Turbosound!  
  
// Bagger -----  
##include "vehicles/Caterpillar323Excavator.h" // Caterpillar 323 Bagger (verwende das  
Fernsteuerungsprofil "FLYSKY_FS_I6S_EXCAVATOR")  
  
// Dumper -----  
##include "vehicles/Benford3TonDumper.h" // Benford 3-Tonnen-Dumper  
  
// US-Motorräder -----  
##include "vehicles/HarleyDavidsonFXSB.h" // Harley Davidson FXSB V2 Motorrad  
  
// US-Autos -----  
##include "vehicles/ChevyNovaCoupeV8.h" // 1975 Chevy Nova Coupé V8 //  
#include "vehicles/1965FordMustangV8.h" // 1965 Ford Mustang V8 ##include  
"vehicles/Chevy468.h" // Chevy 468 Big Block V8  
  
// EU-Autos -----  
##include "vehicles/VwBeetle.h" // VW Käfer / Beetle  
##include „vehicles/JaguarXJS.h“ // Jaguar XJS V12, Schaltgetriebe ##include  
„vehicles/JaguarXJSautomatic.h“ // Jaguar
```

```
///#include "vehicles/LaFerrari.h" // Ferrari LaFerrari, V12, 6-Gang-Doppelkupplungsgetriebe

// US SUV & Pickups -----
///#include "vehicles/JeepGrandCherokeeTrackhawk.h" // Jeep Grand Cherokee Trackhawk V8
Monster-SUV mit Kompressor, 6-Gang-Automatik
///#include "vehicles/FordPowerstroke.h" // Ford Powerstroke 7.3l V8 Diesel, 6-Gang-Automatik (guter
Basslautsprecher erforderlich)
///#include "vehicles/RAM2500_Cummins12V.h" // Dodge RAM 2500 mit Reihensechszylinder, 12V Cummins 5,9l
Dieselmotor, Schaltgetriebe
///#include "vehicles/RAM2500_Cummins12Vautomatic.h" // Dodge RAM 2500 mit Reihensechszylinder, 12V
Cummins 5,9l Dieselmotor, Automatikgetriebe
///#include "vehicles/GMCsierra.h" // GMC Sierra V8 Pickup, 3-Gang-Automatikgetriebe
///#include "vehicles/ChevyNovaCoupeV8_P407.h" // 1975 Chevy Nova Coupé V8, Sonderversion für HG-
P407, 3-Gang-Automatikgetriebe
///#include "vehicles/JeepWranglerRubicon392V8.h" // 2021 Jeep Wrangler Rubicon HEMI 392 V8 (Anlasser muss
überarbeitet werden)
///#include "vehicles/JeepWranglerRubicon392V8_2.h" // 2021 Jeep Wrangler Rubicon HEMI 392 V8
(wahnsinniger Bass!)
```

// EU SUV -----

```
///#include "vehicles/DefenderV8Automatic.h" // Land Rover Defender 90 V8 Automatik (sehr schöner
V8 mit viel Bass)
///#include "vehicles/DefenderV8OpenPipeAutomatic.h" // Land Rover Defender 90 V8 Automatik, offene
Auspuffrohre (optimiert für kleinere Lautsprecher)
///#include "vehicles/DefenderV8CrawlerAutomatic.h" // Land Rover Defender 90 V8 automatischer
Kriechgang
///#include "vehicles/DefenderTd5.h" // Land Rover Defender 90 Td5 R5 Diesel
```

// Asiatische SUVs -----

```
///#include "vehicles/LandcruiserFJ40.h" // Landcruiser FJ40 mit Reihensechszylinder-Benzinmotor
///#include "vehicles/LandcruiserFJ40Diesel.h" // Landcruiser FJ40 mit Reihensechszylinder-12H-Turbodieselmotor

///#include "vehicles/LandcruiserFJ40Diesel2.h" // Landcruiser FJ40 mit Reihensechszylinder-12H-
Turbodieselmotor
///#include "vehicles/HiluxDiesel.h" // Hilux Diesel mit Reihensechszylinder 12H Turbo-Dieselmotor (für HG-P407)
```

// US-Lokomotiven -----

```
///#include "vehicles/UnionPacific2002.h" // Union Pacific SD70M Lokomotive von 2002 mit riesigem, niedrig
drehendem 16-Zylinder-Dieselmotor.
```

// Ebenen -----

```
///#include "vehicles/MesserschmittBf109.h" // Messerschmitt Bf 109, deutsches V12-Flugzeug aus dem Zweiten Weltkrieg
```

// Standard-Dieselmotoren -----

```
##include "vehicles/generic6zylDiesel.h" // Generischer Reihensechszylinder-Dieselmotor, ohne Turbolader, Schaltgetriebe  
(optimiert für kleinere Lautsprecher)
```

Auswahl des Schnittstellentyps (Kommunikationsmodus):

Beachten Sie, dass der Standardkommunikationsmodus SBUS ist. Sie müssen ihn wie folgt ändern, wenn Sie klassische RC-Servosignale verwenden möchten.

```
// KOMMUNIKATIONSEINSTELLUNGEN
```

```
*****
```

```
*****
```

```
// Wählen Sie den Empfängerkommunikationsmodus (niemals mehr als einen einkommentieren!) !!!
```

```
Stellen Sie diese ein, bevor Sie Ihren Empfänger und Regler anschließen !!!
```

```
// PWM-Servosignal-Kommunikation (CH1 - CH4, 35, PPM-Header, 6 Kanäle, channelSetup.h)
```

```
-----
```

```
// PWM-Modus aktiv, wenn SBUS, IBUS und PPM deaktiviert sind (// vor #define)
```

```
// SBUS-Kommunikation (SBUS-Header, 13 Kanäle. Dies ist mein bevorzugtes  
Kommunikationsprotokoll)-----
```

```
##define SBUS_COMMUNICATION // Steuersignale werden über die SBUS-Schnittstelle empfangen (für  
klassische PWM-RC-Signale auskommentieren)
```

```
boolean sbusInverted = true; // false = an nicht standardmäßiges (invertiertes) SBUS-Signal angeschlossen (z. B. vom  
"Micro RC"-Empfänger)
```

```
// IBUS-Kommunikation (RX-Header, 13 Kanäle nicht empfohlen, keine Ausfallsicherheit bei Kontaktproblemen in der  
IBUS-Verkabelung!) -----
```

```
##define IBUS_COMMUNICATION // Steuersignale werden über die IBUS-Schnittstelle empfangen (für  
klassische PWM-RC-Signale auskommentieren)
```

```
// SUMD-Kommunikation (RX-Header, 12 Kanäle, für Graupner-Fernbedienungen) ----- //#define  
SUMD_COMMUNICATION // Steuersignale werden über die SUMD-Schnittstelle empfangen (für klassische  
PWM-RC-Signale auskommentieren)
```

```
// PPM-Kommunikation (RX-Header, 8 Kanäle, funktioniert einwandfrei, aber die Kanalsignale sind etwas unruhig)
```

```
-----
```

```
##define PPM_COMMUNICATION // Steuersignale werden über die PPM-Schnittstelle empfangen (für  
klassische PWM-RC-Signale auskommentieren)
```

PPM (Mehrkanal-Pulspausenmodulation, angeschlossen an den "RX"-Anschluss, 8 Kanäle)

// KOMMUNIKATIONSEINSTELLUNGEN

// Wählen Sie den Empfängerkommunikationsmodus (niemals mehr als einen einkommentieren!) !!!

Stellen Sie diese ein, bevor Sie Ihren Empfänger und Regler anschließen !!!

// PWM-Servosignal-Kommunikation (CH1 - CH4, 35, PPM-Header, 6 Kanäle, channelSetup.h)

// PWM-Modus aktiv, wenn SBUS, IBUS und PPM deaktiviert sind (/ vor #define)

// SBUS-Kommunikation (SBUS-Header, 13 Kanäle. Dies ist mein bevorzugtes

Kommunikationsprotokoll)-----

//#define SBUS_COMMUNICATION // Steuersignale werden über die SBUS-Schnittstelle empfangen (für
klassische PWM-RC-Signale auskommentieren)

boolean sbusInverted = true; // false = an nicht standardmäßiges (invertiertes) SBUS-Signal angeschlossen (z. B. vom
"Micro RC"-Empfänger)

// IBUS-Kommunikation (RX-Header, 13 Kanäle nicht empfohlen, keine Ausfallsicherheit bei Kontaktproblemen in der
IBUS-Verkabelung!) -----

//#define IBUS_COMMUNICATION // Steuersignale werden über die IBUS-Schnittstelle empfangen (für
klassische PWM-RC-Signale auskommentieren)

// SUMD-Kommunikation (RX-Header, 12 Kanäle, für Graupner-Fernbedienungen) ----- //#define
SUMD_COMMUNICATION // Steuersignale werden über die SUMD-Schnittstelle empfangen (für klassische
PWM-RC-Signale auskommentieren)

// PPM-Kommunikation (RX-Header, 8 Kanäle, funktioniert einwandfrei, aber die Kanalsignale sind etwas unruhig)

#define PPM_COMMUNICATION // Steuersignale werden über die PPM-Schnittstelle empfangen (für
klassische PWM-RC-Signale auskommentieren)

SBUS (empfohlen, Standardeinstellung, an den "SBUS"-Anschluss angeschlossen, 13 Kanäle)

// KOMMUNIKATIONSEINSTELLUNGEN

// Wählen Sie den Empfängerkommunikationsmodus (niemals mehr als einen einkommentieren!) !!!

Stellen Sie diese ein, bevor Sie Ihren Empfänger und Regler anschließen !!!

// PWM-Servosignal-Kommunikation (CH1 - CH4, 35, PPM-Header, 6 Kanäle) ----- //
PWM-Modus aktiv, wenn SBUS, IBUS und PPM deaktiviert sind (/ vor #define)

```

// SBUS-Kommunikation (RX-Header, 13 Kanäle. Dies ist mein bevorzugtes
Kommunikationsprotokoll)-----
#define SBUS_COMMUNICATION // Steuersignale werden über die SBUS-Schnittstelle empfangen (für
klassische PWM-RC-Signale auskommentieren)
boolean sbusInverted = false; // false = an nicht standardmäßiges (invertiertes) SBUS-Signal angeschlossen (z. B. von
meinem "Micro RC"-Empfänger)
uint32_t sbusBaud = 100000; // Standardwert ist 100000. Versuchen Sie, ihn zu verringern, falls die Kanäle instabil
ankommen. Der Arbeitsbereich liegt bei etwa 96000–104000.

// IBUS-Kommunikation (RX-Header, 13 Kanäle nicht empfohlen, keine Ausfallsicherheit bei Kontaktproblemen in der
IBUS-Verkabelung!) -----
##define IBUS_COMMUNICATION // Steuersignale werden über die IBUS-Schnittstelle empfangen (für
klassische PWM-RC-Signale auskommentieren)

// SUMD-Kommunikation (RX-Header, 12 Kanäle, für Graupner-Fernbedienungen) ----- ##define
SUMD_COMMUNICATION // Steuersignale werden über die SUMD-Schnittstelle empfangen (für klassische
PWM-RC-Signale auskommentieren)

// PPM-Kommunikation (RX-Header, 8 Kanäle, funktioniert einwandfrei, aber die Kanalsignale sind etwas unruhig)
-----

##define PPM_COMMUNICATION // Steuersignale werden über die PPM-Schnittstelle empfangen (für
klassische PWM-RC-Signale auskommentieren)

```

Nicht standardmäßiges SBUS-Signal (falls Ihr Empfänger ein nicht standardmäßiges SBUS-Signal sendet):

```

boolean sbusInverted = false; // false = an nicht standardmäßiges (invertiertes) SBUS-Signal angeschlossen (z. B. vom
"Micro RC"-Empfänger)

```

SBUS-Standardsignal (Standardeinstellung, wird in den meisten Fällen verwendet)

```

boolean sbusInverted = true; // false = an nicht standardmäßiges (invertiertes) SBUS-Signal angeschlossen (z. B. vom
"Micro RC"-Empfänger)

```

**SBUS-Baudraten-Feineinstellung. 100000 ist der Standardwert. Bei manchen Empfängern können die Anzeigen
und andere Funktionen aufgrund von Bitfehlern zufällig ausgelöst werden. Mit dieser Variable lässt sich die
Baudrate feinjustieren, um dieses Problem zu beheben.**

```

uint32_t sbusBaud = 100000; // Standardwert ist 100000. Versuchen Sie, ihn zu verringern, falls Ihre Kanäle instabil
ankommen. Der Arbeitsbereich liegt bei etwa 96000 – 104000.

```

IBUS (nicht empfohlen, keine Ausfallsicherheit bei Kontaktproblemen in der IBUS-Verkabelung! "RX"-Anschluss, 13 Kanäle)

// KOMMUNIKATIONSEINSTELLUNGEN

// Wählen Sie den Empfängerkommunikationsmodus (niemals mehr als einen einkommentieren!) !!!

Stellen Sie diese ein, bevor Sie Ihren Empfänger und Regler anschließen !!!

// PWM-Servosignal-Kommunikation (CH1 - CH4, 35, PPM-Header, 6 Kanäle, channelSetup.h)

// PWM-Modus aktiv, wenn SBUS, IBUS, SERIAL und PPM deaktiviert sind (/ vor #define)

// SBUS-Kommunikation (SBUS-Header, 13 Kanäle. Dies ist mein bevorzugtes

Kommunikationsprotokoll)-----

//#define SBUS_COMMUNICATION // Steuersignale werden über die SBUS-Schnittstelle empfangen (für klassische PWM-RC-Signale auskommentieren)

boolean sbusInverted = true; // false = an nicht standardmäßiges (invertiertes) SBUS-Signal angeschlossen (z. B. vom "Micro RC"-Empfänger)

// IBUS-Kommunikation (RX-Header, 13 Kanäle nicht empfohlen, keine Ausfallsicherheit bei Kontaktproblemen in der IBUS-Verkabelung!) -----

#define IBUS_COMMUNICATION // Steuersignale werden über die IBUS-Schnittstelle empfangen (für klassische PWM-RC-Signale auskommentieren)

// SUMD-Kommunikation (RX-Header, 12 Kanäle, für Graupner-Fernbedienungen) ----- //#define SUMD_COMMUNICATION // Steuersignale werden über die SUMD-Schnittstelle empfangen (für klassische PWM-RC-Signale auskommentieren)

// PPM-Kommunikation (RX-Header, 8 Kanäle, funktioniert einwandfrei, aber die Kanalsignale sind etwas unruhig)

//#define PPM_COMMUNICATION // Steuersignale werden über die PPM-Schnittstelle empfangen (für klassische PWM-RC-Signale auskommentieren)

SUMD (Für Graupner-Fernbedienungen, "RX"-Anschluss, 12 Kanäle)

// KOMMUNIKATIONSEINSTELLUNGEN

// Wählen Sie den Empfängerkommunikationsmodus (niemals mehr als einen einkommentieren!) !!!

Stellen Sie diese ein, bevor Sie Ihren Empfänger und Regler anschließen !!!

// PWM-Servosignal-Kommunikation (CH1 - CH4, 35, PPM-Header, 6 Kanäle) ----- // PWM-Modus aktiv, wenn SBUS, IBUS und PPM deaktiviert sind (/ vor #define)

// SBUS-Kommunikation (SBUS-Header, 13 Kanäle. Dies ist mein bevorzugtes Kommunikationsprotokoll)-----

```

#define SBUS_COMMUNICATION // Steuersignale werden über die SBUS-Schnittstelle empfangen (für
klassische PWM-RC-Signale auskommentieren)
boolean sbusInverted = false; // false = an nicht standardmäßiges (invertiertes) SBUS-Signal angeschlossen (z. B. von
meinem "Micro RC"-Empfänger)

// IBUS-Kommunikation (RX-Header, 13 Kanäle nicht empfohlen, keine Ausfallsicherheit bei Kontaktproblemen in der
IBUS-Verkabelung!) -----
#define IBUS_COMMUNICATION // Steuersignale werden über die IBUS-Schnittstelle empfangen (für
klassische PWM-RC-Signale auskommentieren)

// SUMD-Kommunikation (RX-Header, 12 Kanäle, für Graupner-Fernbedienungen) -----
#define SUMD_COMMUNICATION // Steuersignale werden über die SUMD-Schnittstelle empfangen (für
klassische PWM-RC-Signale auskommentieren)

// PPM-Kommunikation (RX-Header, 8 Kanäle, funktioniert einwandfrei, aber die Kanalsignale sind etwas unruhig)
-----
#define PPM_COMMUNICATION // Steuersignale werden über die PPM-Schnittstelle empfangen (für
klassische PWM-RC-Signale auskommentieren)

```

Einstellungen in "vehicles/yourVehiclePreset.h" anpassen:

Shaker

Passen Sie die Schüttelleistung an die verschiedenen Motorzustände an Ihre Bedürfnisse an:

```

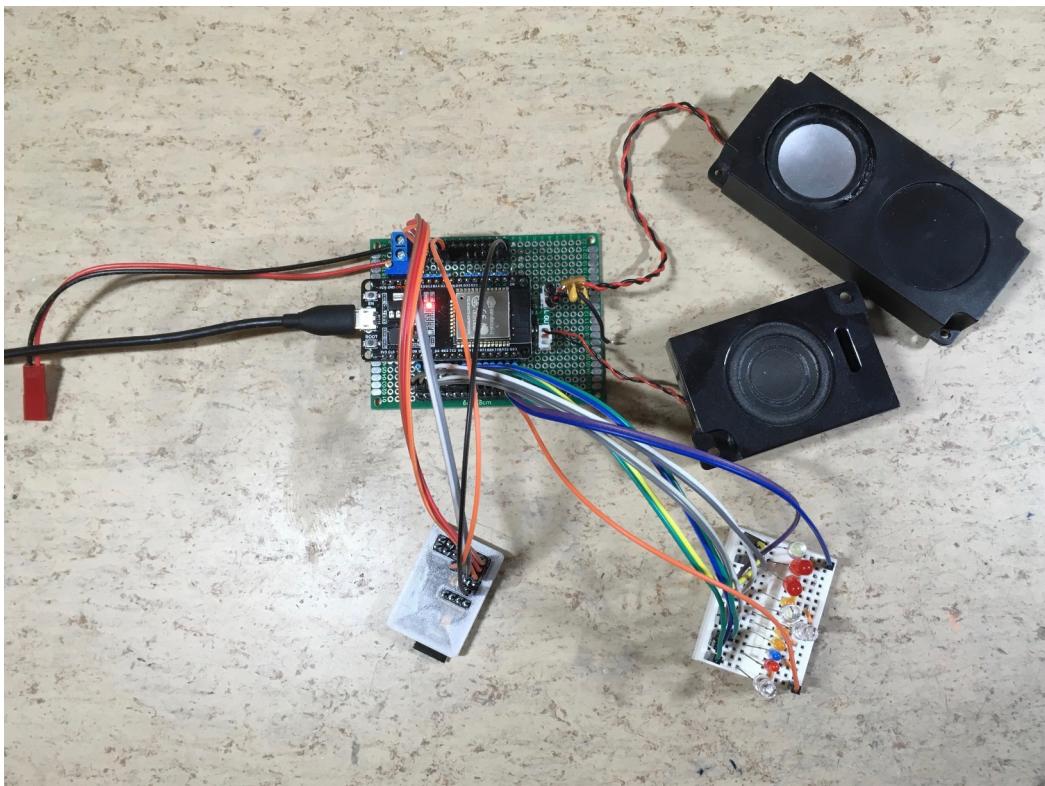
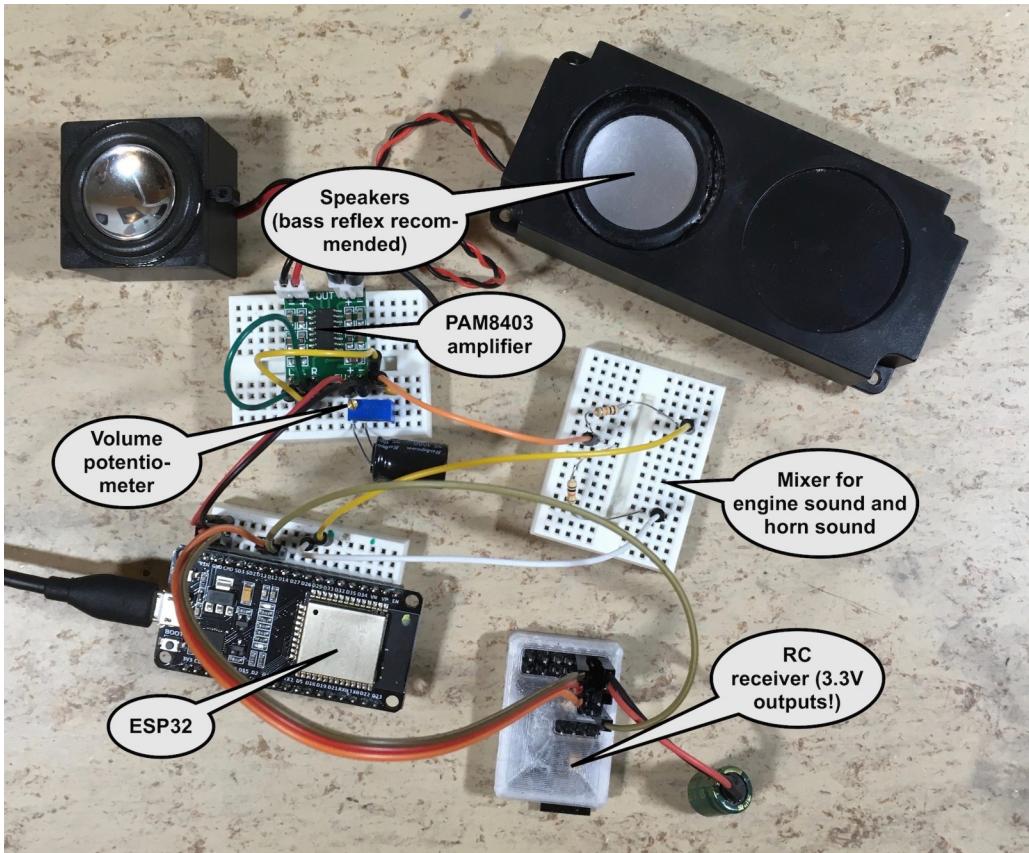
// Parameter des Schwingungserregers (Simulation von Motorschwingungen)

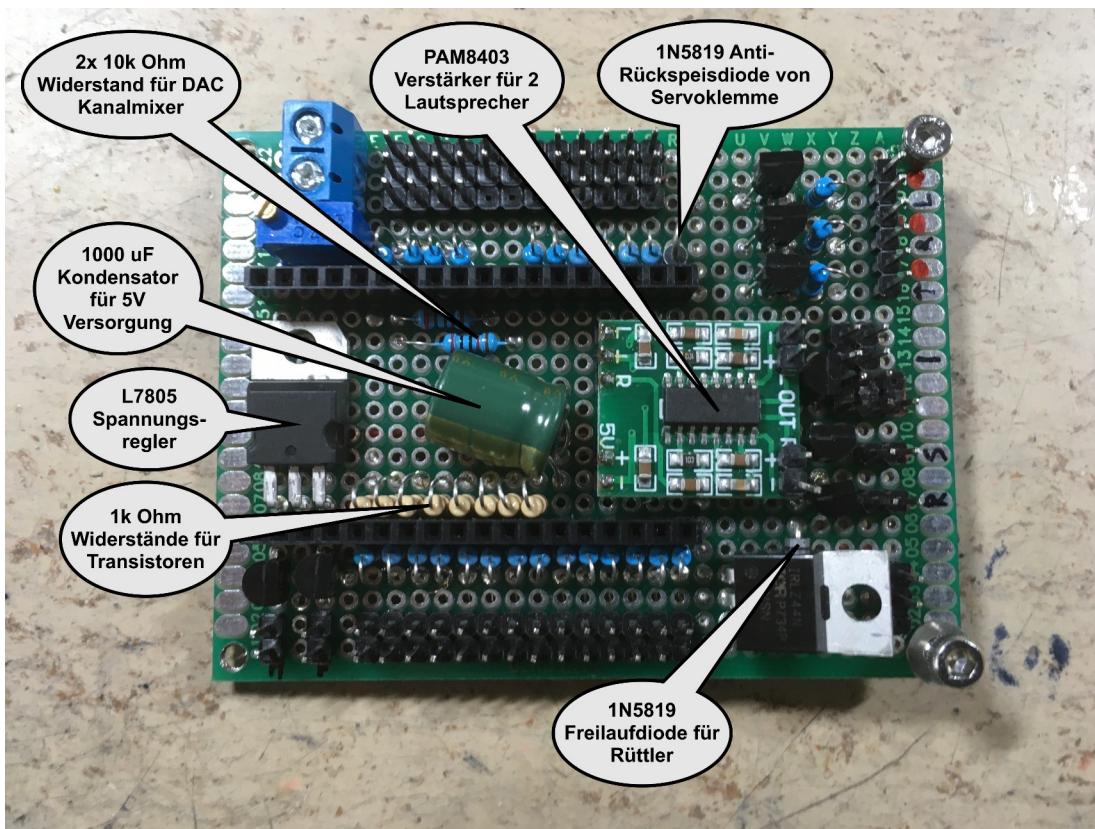
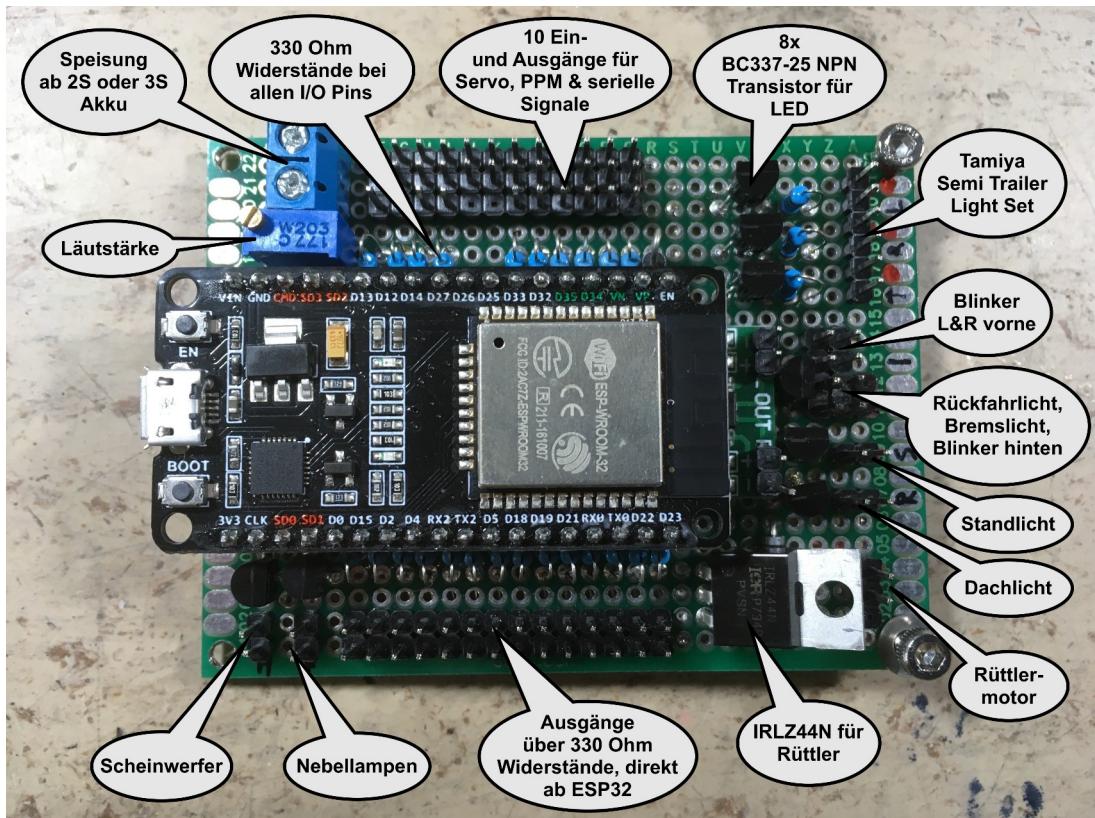
const uint8_t shakerStart = 100; // Shakerleistung beim Motorstart (max. 255, ca. 100) const
uint8_t shakerIdle = 49; // Shakerleistung im Leerlauf (max. 255, ca. 49)
const uint8_t shakerFullThrottle = 40; // Shaker-Leistung bei Vollgas (max. 255, ca. 40) const
uint8_t shakerStop = 60; // Shaker-Leistung bei Motorstillstand (max. 255, ca. 60)

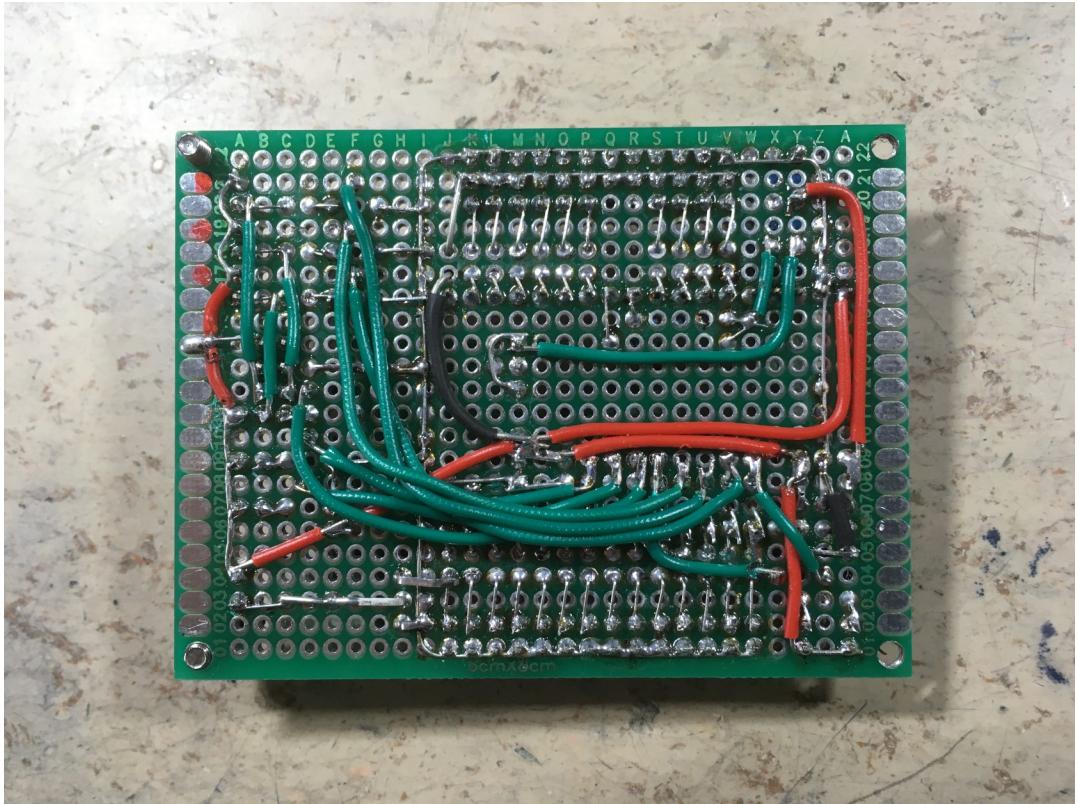
```

Fortsetzung folgt...

Prototypen:







2019–2023, TheDIYGuy999