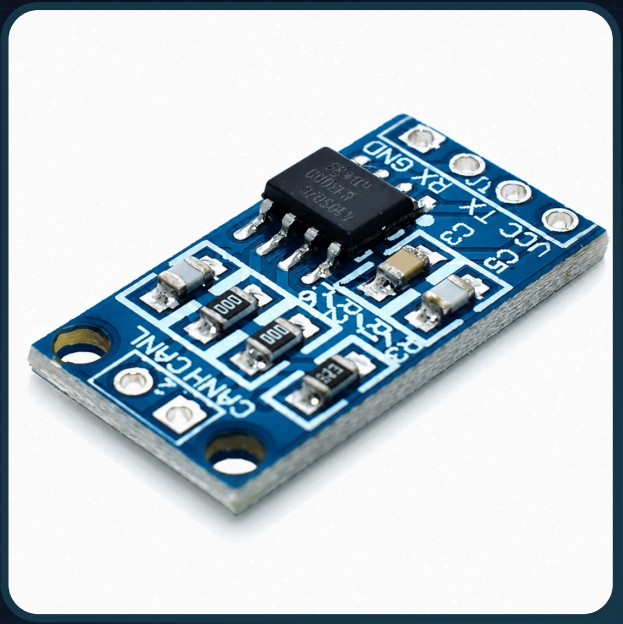
CAN-BUS · ESP32

CAN-Bus TJA1050

Der Transceiver, der deinen ESP32 an den CAN-Bus bringt – wie in Autos und Maschinen. Erster Selbsttest in rund 20 Minuten, ganz ohne zweiten Teilnehmer.



DATENRATE **bis 1 Mbit/s**	BUS **CANH / CANL**
VERSORGUNG **5 V DC**	CONTROLLER **ESP32 TWAI**

01
Verkabeln
4 Leitungen zum ESP32

02
Einrichten
IDE und Board-Paket

03
Hochladen
Beispiel-Sketch

04
Testen
CAN-Selbsttest

01 Auf einen Blick

Der TJA1050 übersetzt die Logikpegel in die differenziellen Signale CANH und CANL. Den eigentlichen CAN-Controller bringt der ESP32 mit („TWAI"). Der Sketch sendet im Selbsttest-Modus und liest seine eigenen Nachrichten über den Transceiver zurück.

Technische Daten

Transceiver	**TJA1050**
Datenrate	**bis 1 Mbit/s**
Versorgung	**5 V DC**
Logik	**5 V (RXD-Ausgang)**
Bus	**CANH, CANL**
Abschluss	**120 Ω auf der Platine (laut Hersteller)**
Maße	**22 × 11 mm**

Lieferumfang

1 × **TJA1050 CAN-Bus-Schnittstellenmodul**

! RXD liefert 5 V
Am ESP32 gehört ein Spannungsteiler in die RX-Leitung: 1 kΩ in Reihe, 2 kΩ nach GND – macht aus 5 V etwa 3,3 V.

i Transceiver ≠ Controller
Am Arduino UNO fehlt der CAN-Controller – dort braucht es zusätzlich einen MCP2515. ESP32 und UNO R4 haben ihn eingebaut.

Pinbelegung

PIN	FUNKTION	SPANNUNG	HINWEIS
● **GND**	Masse	0 V	Gemeinsam mit ESP32 und Bus.
● **RX**	Empfangsausgang	5 V	Über Spannungsteiler an GPIO4.
● **TX**	Sendeeingang	3,3 V reicht	ESP32 GPIO5.
● **VCC**	Versorgung	5 V DC	An VIN des ESP32.
● **CANH**	CAN High	Busleitung	Zu CANH aller Teilnehmer – hier frei.
● **CANL**	CAN Low	Busleitung	Zu CANL aller Teilnehmer – hier frei.

Die 4-polige Seite ist mit GND · RX · TX · VCC beschriftet (GND hat das eckige Lötauge), am anderen Ende sitzen CANH und CANL.

Was du zusätzlich brauchst

ESP32 DevKit USB-C
WROOM-32, CH340C – 3,3-V-Logik
`MKR-ESP32-005`

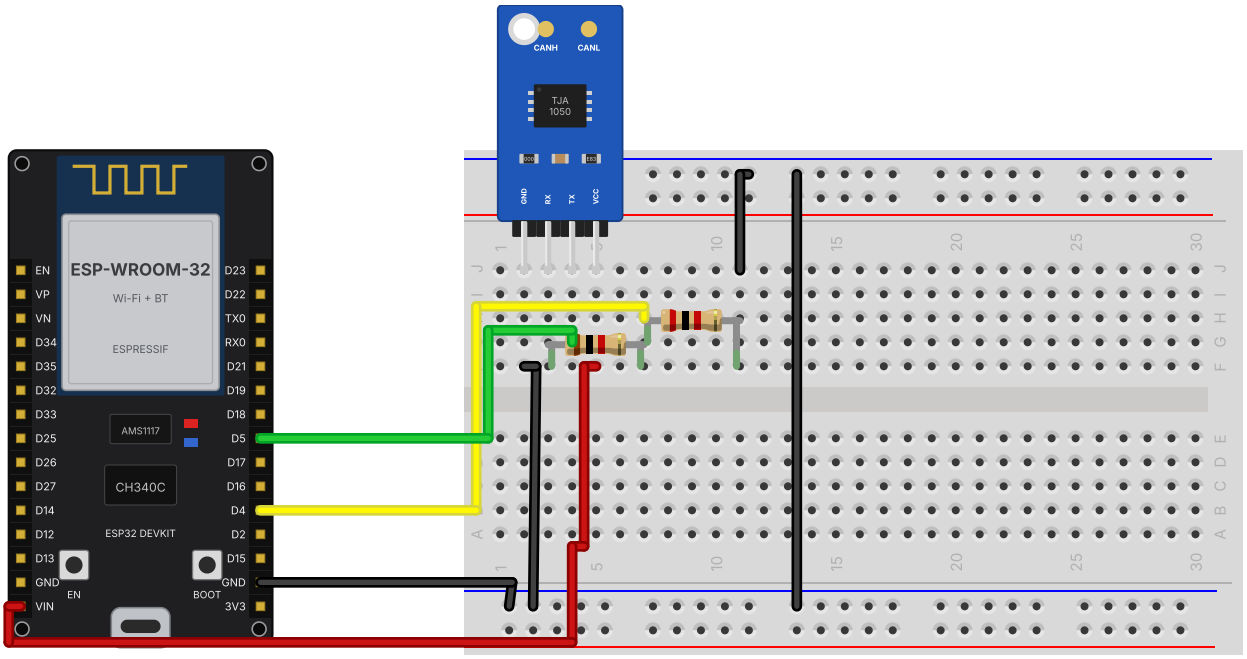
Breadboard + Jumper-Set
400 Kontakte, 40 Dupont-Kabel
`MKR-SET-BB400-JUMPER`

Widerstände 1 kΩ + 2 kΩ
Spannungsteiler für RX
`600x Widerstände SET`

02 Verkabeln

Vier Leitungen und zwei Widerstände. CANH und CANL bleiben für den Selbsttest frei.

STECKPLATINEN-ANSICHT



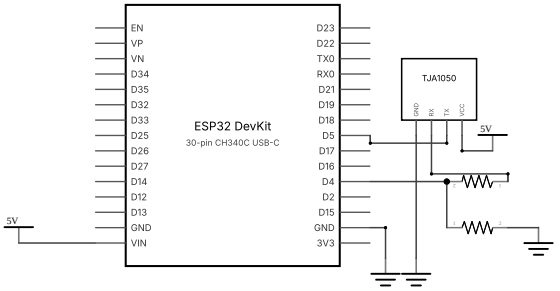
Verbindungen

MODUL		ESP32 DEVKIT	LEITUNG
VCC	→	VIN	5 V
GND	→	GND	Masse
TX	→	D5	Senden
RX	→	D4	über 1 kΩ / 2 kΩ

Spannungsteiler
RX → 1 kΩ → Knoten → GPIO4. Vom selben Knoten 2 kΩ nach GND.

Zwei Teilnehmer
Für echten Busbetrieb CANH mit CANH und CANL mit CANL verdrillt verbinden, GND mitführen, an beiden Enden 120 Ω.

SCHALTPLAN



Verdrahtung erstellt mit Fritzing

03 Einrichten & Hochladen

Schritte 1 bis 3 machst du nur ein einziges Mal – danach ist dein ESP32 für alle weiteren Projekte startklar.

1 Arduino IDE installieren

Lade die aktuelle Arduino IDE 2 von `arduino.cc/en/software` herunter und installiere sie.

2 ESP32-Board-Paket installieren

Im Boardverwalter (Board-Symbol links) nach **esp32** suchen und **esp32 von Espressif Systems** installieren. Den CH340C-Treiber bringen Windows 10/11 und macOS meist mit. Ausführlich mit Bildern: unser Guide zum ESP32 DevKit.

3 Board und Port wählen

ESP32 per USB-C anschließen. In der Board-Auswahl **ESP32 Dev Module** wählen und den neuen COM-Port (Windows) bzw. `/dev/ttyUSB0` (Linux) / `/dev/cu.wchusbserial…` (macOS).

4 Keine Bibliothek nötig

Der CAN-Treiber *TWAI* steckt bereits im ESP32-Board-Paket – weiter mit Schritt 5.

5 Sketch hochladen

Neuen Sketch anlegen, den Code von Seite 5 einfügen und mit dem Pfeil-Symbol hochladen (`Strg` + `U`). Unten erscheint „Hochladen abgeschlossen“.

6 Seriellen Monitor öffnen

`Strg` + `Umschalt` + `M`, unten rechts **115200 Baud** einstellen. Jede Sekunde sendet der ESP32 eine Nachricht und meldet, was zurückkam.

BEISPIELAUSGABE · SERIELLER MONITOR · 115200 BAUD

```
CAN bereit: 500 kbit/s, Selbsttest
Empfangen: ID 0x123 Daten 00 AB
Empfangen: ID 0x123 Daten 01 AB
Empfangen: ID 0x123 Daten 02 AB
```

i Schnelltest

Das erste Datenbyte zählt bei jeder Nachricht hoch. Kommt „Nichts empfangen“, ist die Strecke ESP32 → TX → Transceiver → RX → ESP32 unterbrochen.

04 Der erste Sketch

Startet den CAN-Controller des ESP32 im Selbsttest-Modus, sendet jede Sekunde eine Nachricht und liest sie wieder ein.

MAKEROO_TJA1050_Start.ino

```
1  // MAKEROO · TJA1050 CAN-Modul · Selbsttest mit dem ESP32 (TWAI)
2  #include "driver/twai.h"
3
4  const gpio_num_t CAN_TX = GPIO_NUM_5, CAN_RX = GPIO_NUM_4;   // RX ueber Spannungsteiler!
5  uint8_t zaehler = 0;
6
7  void setup() {
8    Serial.begin(115200);
9    twai_general_config_t g = TWAI_GENERAL_CONFIG_DEFAULT(CAN_TX, CAN_RX, TWAI_MODE_NO_ACK);
10   twai_timing_config_t t = TWAI_TIMING_CONFIG_500KBITS();
11   twai_filter_config_t f = TWAI_FILTER_CONFIG_ACCEPT_ALL();
12   if (twai_driver_install(&g, &t, &f) != ESP_OK || twai_start() != ESP_OK) {
13     Serial.println("TWAI-Start fehlgeschlagen!");
14     while (true) delay(10);
15   }
16   Serial.println("CAN bereit: 500 kbit/s, Selbsttest");
17 }
18
19 void loop() {
20   twai_message_t tx = {};
21   tx.identifier = 0x123;
22   tx.data_length_code = 2;
23   tx.data[0] = zaehler++;
24   tx.data[1] = 0xAB;
25   tx.self = 1;                                // eigene Nachricht mitlesen
26   twai_transmit(&tx, pdMS_TO_TICKS(100));
27   twai_message_t rx;
28   if (twai_receive(&rx, pdMS_TO_TICKS(100)) == ESP_OK) {
29     Serial.printf("Empfangen: ID 0x%03X  Daten %02X %02X\n",
30                   (unsigned)rx.identifier, rx.data[0], rx.data[1]);
31   } else {
32     Serial.println("Nichts empfangen - Verkabelung pruefen");
33   }
34   delay(1000);
35 }
```

Zeile 4
Pins
TX an GPIO5, RX an GPIO4 – beide frei wählbar.

Zeile 9
Selbsttest
TWAI_MODE_NO_ACK sendet auch ohne zweiten Teilnehmer, der sonst bestätigen müsste.

Zeile 10
Datenrate
500 kbit/s – bei Autos und vielen Geräten üblich.

Zeile 21-24
Nachricht
ID, Länge und zwei Datenbytes.

Zeile 25
Eigene Nachricht
Der Controller nimmt die gesendete Nachricht selbst wieder an.

Zeile 28-30
Empfangen
Wartet bis 100 ms auf eine Nachricht und gibt sie aus.

Geprüft: kompiliert für ESP32 Dev Module (Board-Paket esp32 3.3.12) (269.3 kB Flash = 21 %, 22 kB RAM).

05 Wenn es nicht klappt

SYMPTOM	LÖSUNG
„Nichts empfangen“	TX an GPIO5? RX über den Teiler an GPIO4? VCC wirklich an 5 V (VIN)? Der TJA1050 läuft nicht an 3,3 V.
„TWAI-Start fehlgeschlagen!“	Pins im Sketch prüfen; GPIO6–11 sind für den Flash reserviert.
Am Auto kommen keine Daten	Datenrate prüfen (meist 500 kbit/s), Modus auf `TWAI_MODE_NORMAL` stellen und nur lesen. Am OBD-Stecker: Pin 6 CANH, Pin 14 CANL.
Bus mit mehreren Modulen stört	Abschluss nur an den beiden Enden – an Modulen dazwischen den 120-Ω-Widerstand entfernen.
Upload bricht ab	Beim „Connecting...“ die BOOT-Taste des ESP32 gedrückt halten, bis der Upload startet.

06 Wie geht's weiter?

Mit ESP32 & ESPHome

Gleiche Verdrahtung wie hier – in ESPHome als *esp32_can*:

```
canbus:
  - platform: esp32_can
    tx_pin: GPIO5
    rx_pin: GPIO4
    can_id: 4
    bit_rate: 500kbps
    on_frame:
      - can_id: 0x123
        then: ...
```

Projektideen

Fahrzeugdaten
OBD-II mitlesen (nur lesen!)

Sensornetz
Mehrere ESP32 über lange Kabel

Modellbahn
Weichen und Signale über CAN

Maschinen
Daten aus Industriegeräten

Passt dazu

ESP32 DevKit USB-C
WROOM-32, CH340C – 3,3-V-Logik
`MKR-ESP32-005`

2x Nano V3
Weitere Busknoten mit MCP2515
`MKR-NANO-V3-2SET-007`

ESP32 Gehäuse aus Acryl
Schützt das Board
`MKR-ESP32-HOUSING-045`

0,96" OLED-Display
Werte direkt anzeigen
`MKR-0.96-OLED-015`

Sicherheitshinweise. Elektronikmodul für Ausbildungs-, Entwicklungs- und Experimentierzwecke. Nur mit Kleinspannung betreiben, niemals an Netzspannung. Nicht geeignet für Kinder unter 14 Jahren; Nutzung nur unter Aufsicht. Elektroaltgeräte gehören nicht in den Hausmüll – Rückgabe kostenlos bei kommunalen Sammelstellen.

Verantwortlicher Wirtschaftsakteur (GPSR): Alexander Prez · makeroo · Hinterm Bahnhof 4 · 90513 Zirndorf · support@makeroo.de
Fragen? Schreib uns mit einem Foto deiner Verkabelung.
Arduino ist eine Marke der Arduino SA und dient nur der Kompatibilitätsangabe.