

# HOWTO RUN<sub>YOUR</sub>

# {CODE} RACE!

01100111 01100101 01110100 00100000 01101001 01101110 00100000  
01100011 01101111 01101110 01110100 01100001 01100011 01110100



**codewerk**

## Inhalt

{CODE}.RACER – Überblick .....	2
Hardwarekomponenten .....	3
Ultraschallsensor und Servomotor .....	3
Antriebsmotoren und Motortreiber IC .....	4
ESP WROOM-32 Mikrokontroller .....	4
Taster und LEDs .....	4
Schaltung und Anbindung an den ESP32 .....	4
Codewerk {CODE}.RACER GIT .....	6
Arduino Entwicklungsumgebung .....	7
IDE Download und Installation .....	7
Installieren der ESP32 Support Files .....	7
Installieren der benötigten Libraries .....	9
Kompilieren .....	10
Upload auf den {code}.racer .....	11
{CODE}.RACER API .....	13

## {CODE}.RACER – Überblick

Der `{code}.racer` ist ein Mikrocontroller gesteuertes Fahrzeug. Er basiert auf 2 Antriebsmotoren mit entsprechenden Treiberbaustein, einem Ultraschallsensor, einem Servomotor und dem ESP32  $\mu$ C. Die Hardware und die Verbindungen der einzelnen Komponenten untereinander, wird detailliert in Kapitel **Hardwarekomponenten** beschrieben.

Als Entwicklungsumgebung kann z.B. die Arduino IDE genutzt werden – die Konfiguration und Anbindung wird im Kapitel **Arduino Entwicklungsumgebung** gezeigt.

Der ESP32  $\mu$ C von Espressif ist kompatibel mit der Arduino IDE. Der `{code}.racer` kann in C/C++ programmiert werden – alle nötigen Informationen finden sich in diesem Handbuch. Alternativ – oder ergänzend – steht eine API mit wesentlichen Basisfunktionen zur Verfügung. Details, inklusive Beispiele dazu finden sich im Kapitel **{CODE}.RACER API**.

Alle benötigten und beschriebenen Softwarekomponenten, hier genutzte Arduino Libraries inklusive Codebeispiel, können wie im Kapitel **Codewerk {CODE}.RACER GIT** beschrieben aus unserem Codewerk GIT geholt werden.



## Hardwarekomponenten

Die Hardwareausstattung ist sehr einfach. Abbildung 1 zeigt die einzelnen Komponenten - Tabelle 1 listet die genauen Bezeichnungen der Teile.

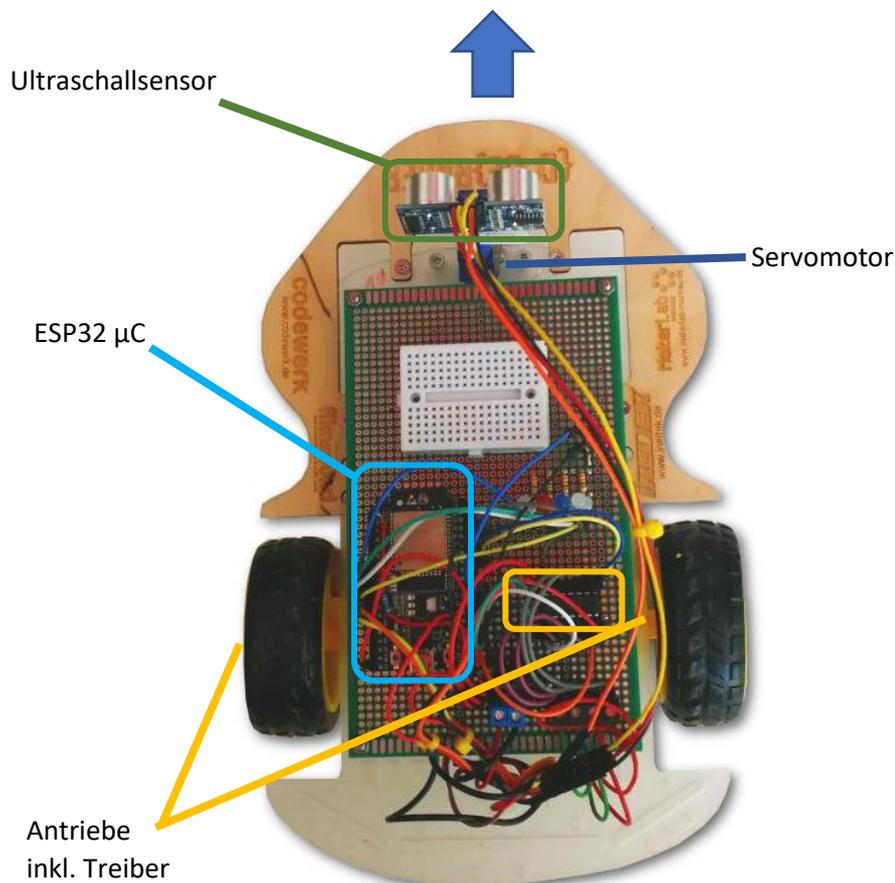


Abbildung 1 - Hardwareschema {code}.racer

Tabelle 1 - Liste der Hardwarekomponenten

Ultraschallsensor	HC-SR04
Servomotor	SG90
Motortreiber	L293D
Mikrokontroller	ESP32 DevKitC
Antriebsmotoren	5V Getriebemotoren, Teil des Smart Car Chassis vom Roboter Bausatz Shop

### Ultraschallsensor und Servomotor

Zur Erfassung der Umgebung dient ein Ultraschallsensor. Dieser ist auf einen Servomotor montiert und um bis zu maximal 90° in beide Richtungen horizontal schwenkbar. Eine vertikale Schwenkung ist nicht möglich. Es sind Entfernungen bis zu ca. 2m mit einer Genauigkeit – je nach Entfernung - von etwa 0,5 bis 2cm messbar. Die API unterstützt eine Messung bis etwa 1m.

## Antriebsmotoren und Motortreiber IC

Angetrieben wird der `{code}.racer` von zwei Getriebemotoren, deren Richtung und Geschwindigkeit unabhängig von einander eingestellt werden können. Somit ist eine Lenkung und Kalibrierung der Laufrichtung recht einfach möglich. Eine direkte Messung der Verfahwege ist nicht vorgesehen.

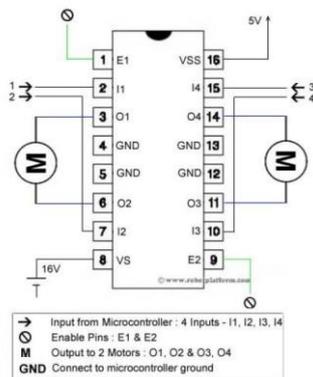


Abbildung 2 - L293D Motortreiber

Da der ESP32, die Motoren nicht direkt ansteuern kann, ist ein L293D IC zwischen geschaltet. Mit diesem lassen sich zwei H-Brücken zur Ansteuerung der beiden Antriebsmotoren realisieren.

Der L293D ist direkt an die PWM Kanäle des ESP32 angeschlossen und kann die Motoren mit 600mA (peak 1,2A) pro Kanal und bis zu 36V versorgen.

## ESP WROOM-32 Mikrokontroller

Die programmierbare Steuereinheit des `{code}.racer` ist der neuste ESP  $\mu$ C von Espressif. Benutzt wird das ESP32 DevKitC Modul – ein kleines Board, das neben dem ESP auch Schnittstellen enthält um den Schaltkreis per USB zu programmieren, ebenso Spannungswandler, Flash, RAM usw. Der ESP ist sehr preisgünstig (<15€). Neben seinen zwei Cores die mit bis zu 240MHz laufen können, bietet er eine immense Liste von Features und Schnittstellen – wie WiFi, Ethernet, Bluetooth, Bluetooth LE, kapazitive Touchsensoren, AD-, DA-Wandler, SPI, I2C, Hallensoren, SD-Card Ansteuerung.



Der ESP32 ist kompatibel mit der Arduino IDE und wird von allen gängigen für Arduino verfügbaren Bibliotheken unterstützt. Somit ist die Anbindung von externer Hardware und Sensoren an den  $\mu$ C relativ einfach.

## Taster und LEDs

Ein Taster und 4 LEDs ermöglichen eine einfache Zustandsanzeige bzw. Interaktion mit dem `{code}.racer`. Die API nutzt die LEDs zur Anzeige der Fahrrichtung, der Taster kann z.B. genutzt um gezielt zu starten bzw. anzuhalten.

## Schaltung und Anbindung an den ESP32

Die Schaltung und das Mapping der  $\mu$ C Pins an die externen Aktoren und den Ultraschallsensor zeigt Abbildung 3. Wird die Arduino `{code}.racer` Library, die die API zur Verfügung gestellt geladen, ist die externe Hardware im C/C++ Code über die symbolischen Namen aus der Abbildung ansprechbar. Der Taster kann dann beispielsweise per `digitalRead(TASTERPIN)` abgefragt werden. Wird die Library nicht eingebunden wird der Taster einfach per Pin - hier `digitalRead(17)` – abgefragt. Das ist auf alle anderen externen benötigten Signale so übertragbar.

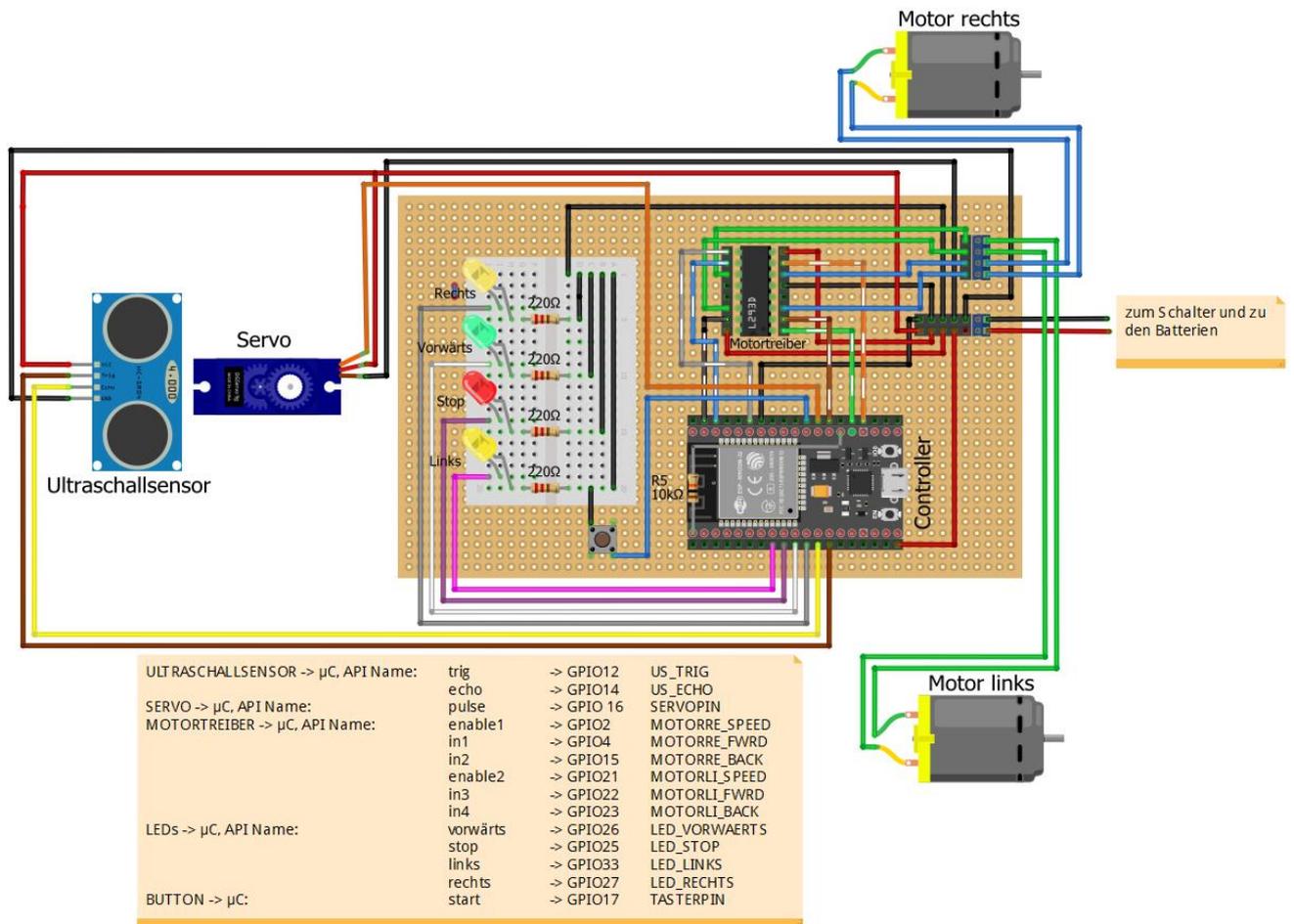


Abbildung 3 - Schematics und Pinmapping {code}.racer

Codewerk {CODE}.RACER GIT

<https://git.itsblue.de/Fenoglio/coderacer/tree/master/>

Hier ist die Dokumentation und alle benötigten Arduino Installationsfiles sowie Libraries.

# Arduino Entwicklungsumgebung

Die Arduino Grundlagen und sehr übersichtliche Tutorials zu einfachen kleinen Projekten in und mit dieser Umgebung finden sich direkt auf der Arduino Homepage <https://www.arduino.cc/> und im Netz. Dieses Kapitel zeigt die nötigen Schritte um die IDE für das `{code}.racer` Projekt einzurichten.

## IDE Download und Installation

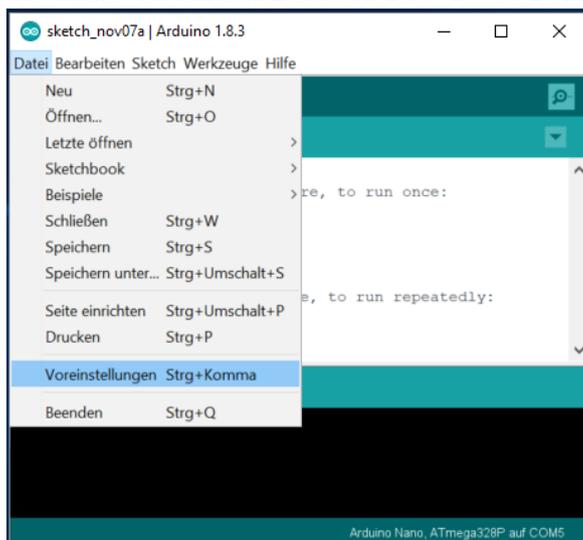
Die Arduino IDE ist für Windows, Mac und Linux verfügbar und kostenlos. Die aktuellen Versionen können auf der Arduino Homepage <https://www.arduino.cc/en/Main/Software> heruntergeladen werden. Die Version 1.8.7 liegt auch im Codewerk GIT.

Detaillierte Installationsbeschreibungen für alle unterstützten Plattformen findet man auch auf der Arduino Homepage <https://www.arduino.cc/en/Guide/HomePage>.

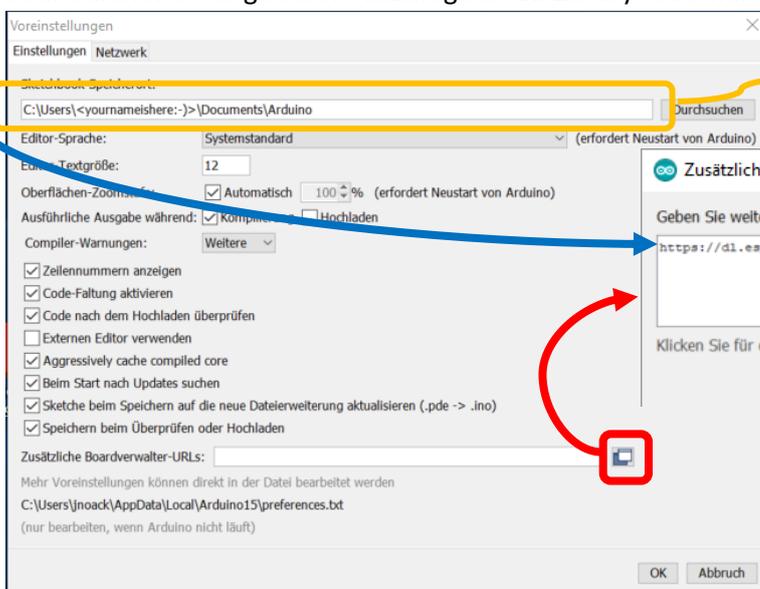
## Installieren der ESP32 Support Files

Da der ESP32 kein „echter“ Arduino ist, müssen sogenannte Board-Files einmalig installiert werden. Dies funktioniert am einfachsten direkt in der IDE. Die einzelnen Schritte sind im Folgenden beschrieben:

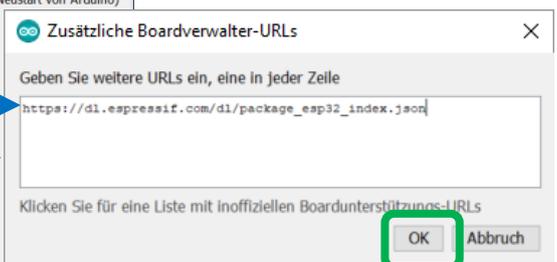
1. Installierte Arduino IDE öffnen. Datei->Voreinstellungen auswählen.



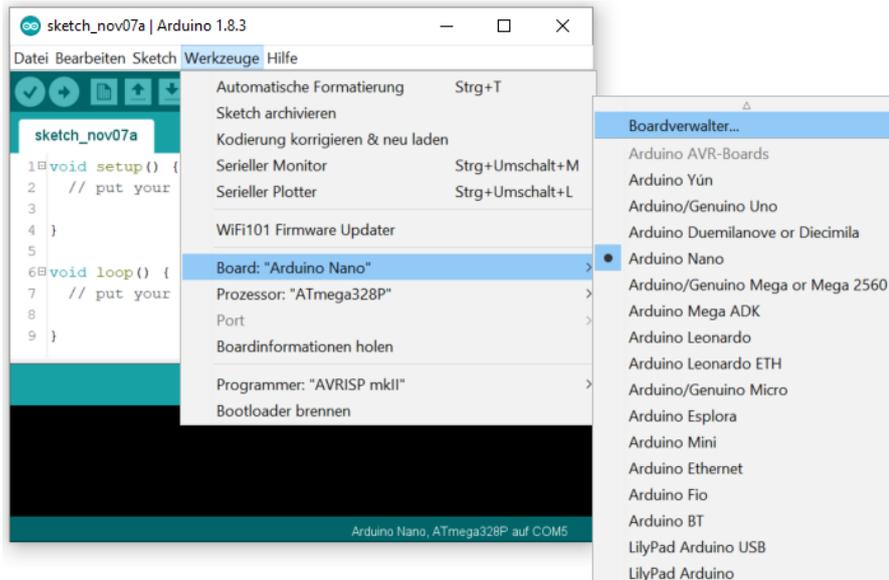
2. [https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json) als Boardverwalter URL eintragen oder durch Komma getrennt hinzufügen. OK. Library Pfad merken. OK.



Library-Pfad. Für später merken.



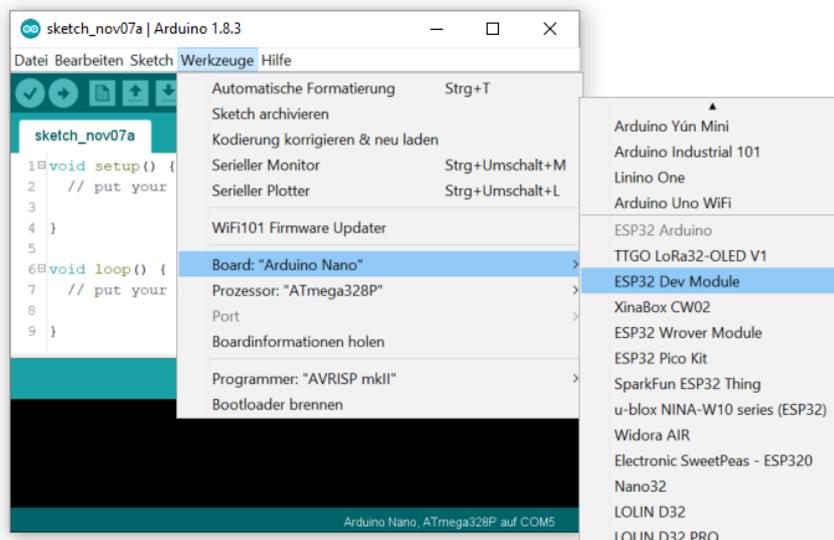
3. Werkzeuge->Board:...-> Bordverwalter



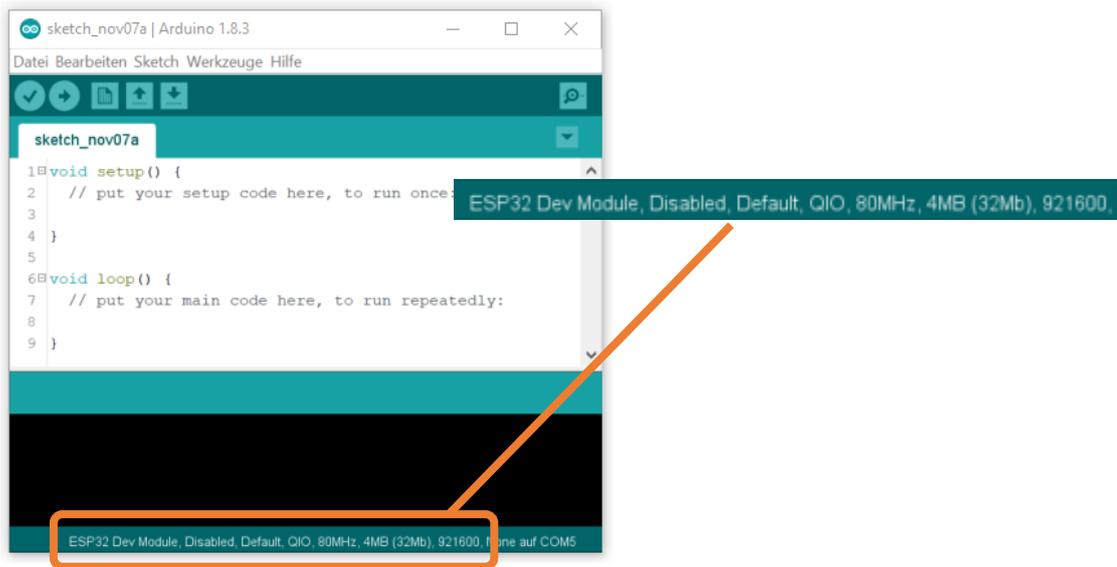
4. Im Boardverwalter nach 'ESP32' suchen. More Info und Installieren für das gefundene Paket auswählen. Nach der Installation das Fenster Schließen.



5. Unter Werkzeuge->Board:...-> gibt es jetzt ein ESP32 Dev Module. Evtl. etwas nach unten scrollen. Das Board auswählen ...



6.

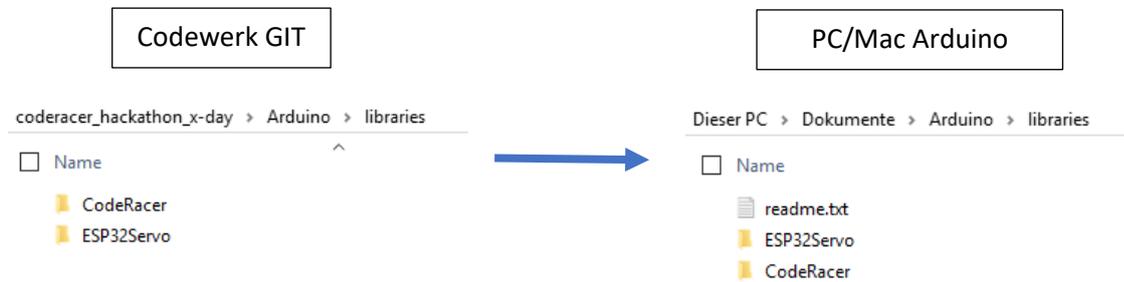


7. Done! 😊

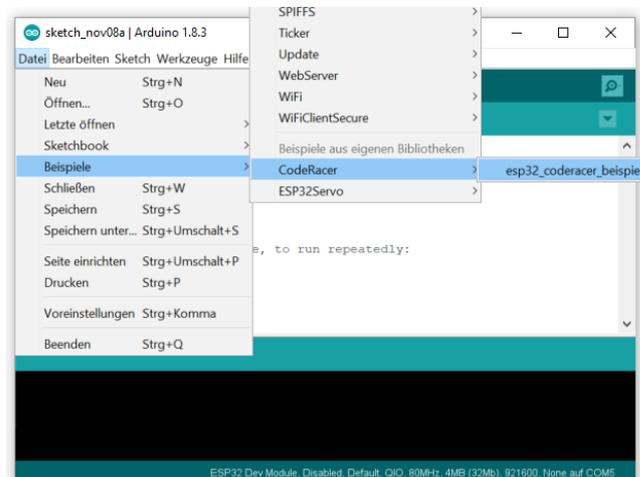
### Installieren der benötigten Libraries

Es werden zwei externe Libraries benötigt: ‚ESP32Servo‘ um den Servomotor anzusteuern und ‚CodeRacer‘ um die API nutzen zu können. Beide Libraries befinden sich im Codewerk GIT im Ordner Arduino/libraries.

Die beiden Verzeichnisse aus dem GIT in den ‚gemerkten‘ Library-Pfad in den Ordner ‚libraries‘ kopieren.



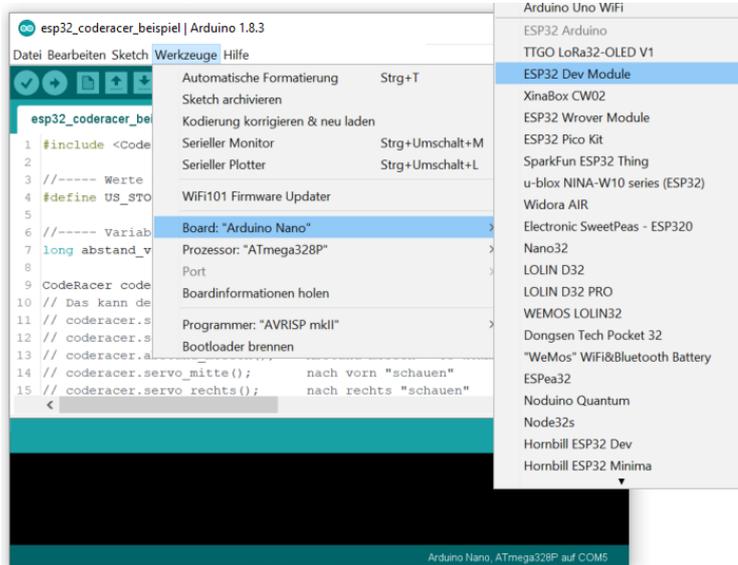
Nach einem Neustart der Arduino IDE finden sich jetzt unter Datei/Beispiele Einträge für ESP32Servo und CodeRacer (evtl. ganz nach unten scrollen).



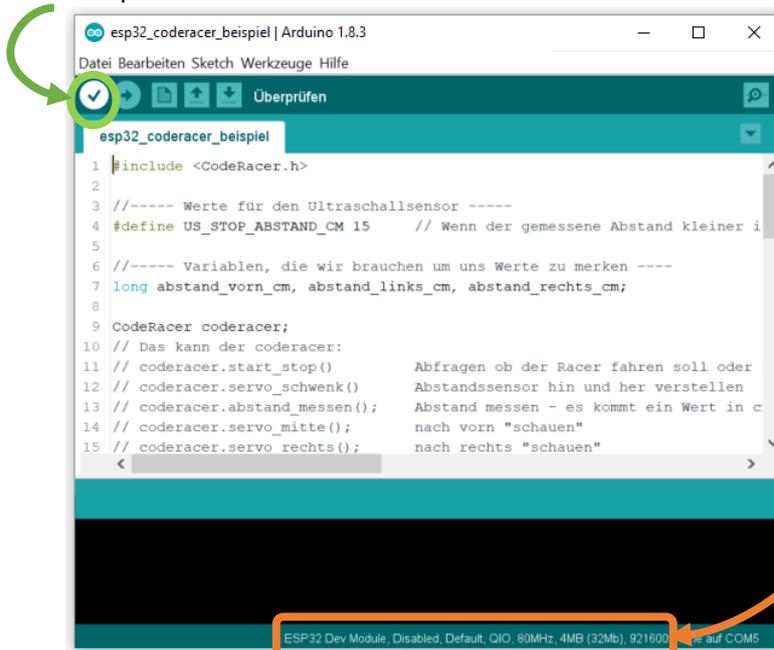
## Kompilieren

Nach einem Klick auf das esp32\_coderacer\_beispiel Beispiel öffnet sich ein neues Fenster. Auf das Beispiel selbst wird hier nicht im Detail eingegangen, da der Code ausführlich kommentiert ist. Sind die beiden Libraries und das Boardfile korrekt installiert und korrekt eingestellt, kompiliert der Beispielcode jetzt erfolgreich:

### 1. Einstellen bzw. prüfen des ,ESP32 Dev Module ` Boards

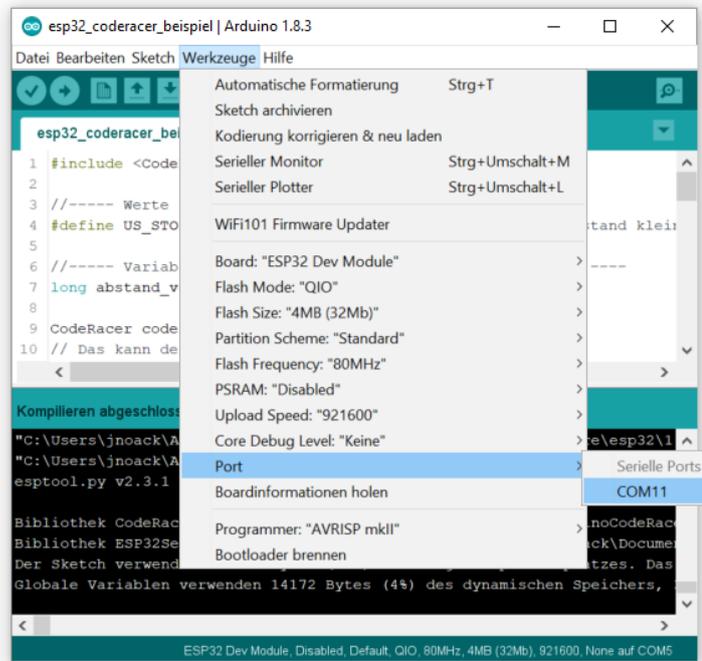


### 2. Kompilieren ...



## Upload auf den {code}.racer

1. Das Board korrekt auf ‚ESP32 Dev Module‘ einstellen.
2. Den ESP32 auf dem {code}.racer per  $\mu$ USB Kabel mit dem PC/Mac verbinden.
3. Den seriellen Port für die Verbindung einstellen:



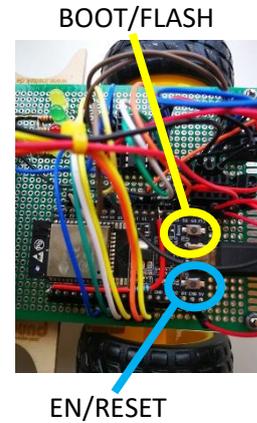
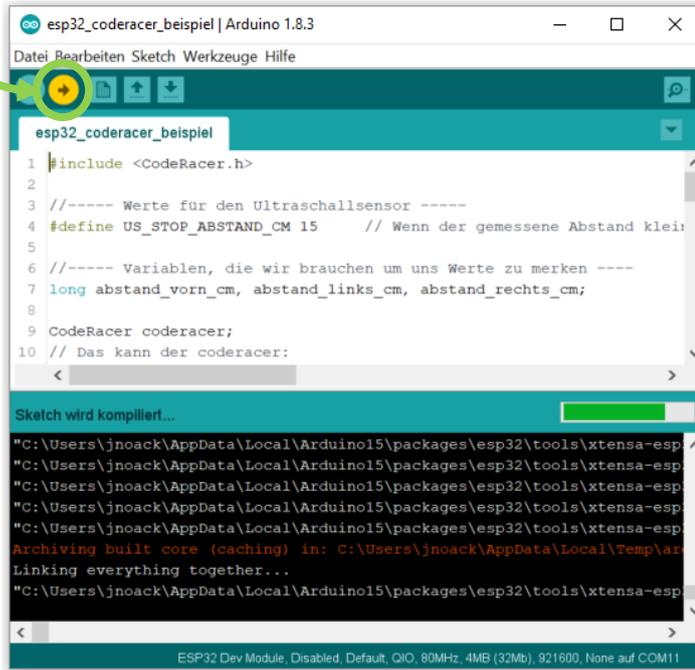
Der Port sieht je nach System anders aus – die Nummer wird auch eine andere sein. Sind mehrere Ports gelistet – einfach mal den {code}.racer ab- und wieder einstecken.



Wird keine Auswahl angeboten – oder der Port Eintrag ist nicht anwählbar, dann ist eventuell noch kein serieller Port im Betriebssystem eingerichtet – oder das USB Kabel unterstützt eventuell keine Datenübertragung.

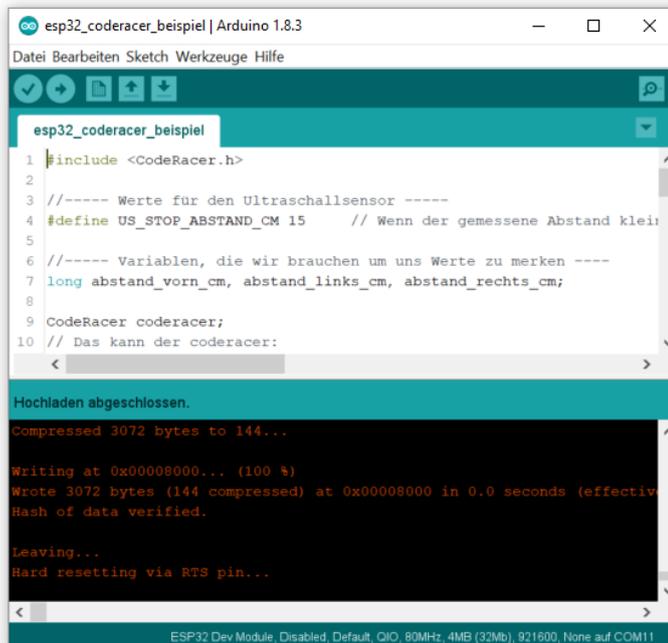
Die Installation des Windows 10 Treibers für den seriellen Port ist z.B. hier detailliert beschrieben: <https://startingelectronics.org/software/arduino/installing-arduino-software-windows-10/>

#### 4. Upload ...



Sollte im Statusfenster ‚Connecting ...---...---...‘ angezeigt werden, dann am ESP32 der BOOT/FLASH Button während dieser Meldung solange gedrückt halten bis eine Verbindung hergestellt wurde. Sollte das ebenfalls nicht funktionieren dann den EN/RST Button während der Meldung gedrückt halten. Dieses Problem ist bekannt und daran wird bei espressif gearbeitet.

#### 5. Fertig ...



## {CODE}.RACER API

Die Beschreibung der `{code}.racer` API ist auf dem GIT unter folgendem Link zu finden:

<https://git.itsblue.de/Fenoglio/coderacer/tree/master/Doku/coderacerAPI.pdf>