HOWTO RUNYOUR



codewerk

Inhalt

{CODE}.RACER – Überblick
Hardwarekomponenten
Ultraschallsensor und Servomotor3
Antriebsmotoren und Motortreiber IC 4
ESP WROOM- 32Mikrokontroller
Taster und LEDs
Schaltung und Anbindung an den ESP324
Codewerk {CODE}.RACER GIT
Arduino Entwicklungsumgebung
IDE Download und Installation
Installieren der ESP32 Support Files7
Installieren der benötigten Libraries9
Kompilieren
Upload auf den {code}.racer11
{CODE}.RACER API

{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 µC. 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 μ 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

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 Verfahrwege ist nicht vorgesehen.



Abbildung 2 - L293D Motortreiber

Da der ESP32, die Motoren nicht direkt ansteuern kann, ist ein L293D IC zwischen geschalten. 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}.racers ist der neuste ESP µC 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 Touchsenoren, AD- , DA-Wandler, SPI, I2C, Hallsenoren, 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 μ 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 μ 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 <u>https://www.arduino.cc/</u> und im Netz.Dieses Kapitel zeigt die nötigen Schritte um die IDE für das {code}.racer Projekt einzurichten.

IDE Download und Installation

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

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

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:

- sketch nov07a | Arduino 1.8.3 × Datei Bearbeiten Sketch Werkzeuge Hilfe Neu Strg+N Q Öffnen... Strg+O -Letzte öffnen Sketchbook >re, to run once: Beispiele Schließen Strg+W Speichern Stra+S Speichern unter... Strg+Umschalt+S e, to run repeatedly: Seite einrichten Strg+Umschalt+P Stra+P Drucken Voreinstellungen Strg+Komma Beenden Strg+Q Arduino Nano, ATmega328P auf COM
- 1. Installierte Arduino IDE öffnen. Datei->Voreinstellungen auswählen.

2. <u>https://dl.espressif.com/dl/package_esp32_index.json</u> als Boardverwalter URL eintragen oder durch Komma getrennt hinzufügen. OK. Library Pfad merken. OK.



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

💿 sketch_nov07a Ardu	ino 1.8.3	-		×		
Datei Bearbeiten Sketch 🛚	Werkzeuge Hilfe					
sketch nov07a	Automatische Formatierung Sketch archivieren Kodierung korrigieren & neu lad	Strg	+T			∆ Boardverwalter
1 ^{II} void setup() { 2 // put your	Serieller Plotter	Strg Strg	+Umsch +Umsch	nalt+M nalt+L		Arduino AVR-Boards Arduino Yún Arduino/Genuino Uno
4 } 5 6 void loop() {	WiFi101 Firmware Updater Board: "Arduino Nano"			>	•	Arduino Duemilanove or Diecimila Arduino Nano
7 // put your 8 9 }	Prozessor: "ATmega328P" Port Boardinformationen holen			>	•	Arduino/Genuino Mega or Mega 2560 Arduino Mega ADK Arduino Leonardo
	Programmer: "AVRISP mkII" Bootloader brennen			>		Arduino Leonardo ETH Arduino/Genuino Micro Arduino Esplora
						Arduino Mini Arduino Ethernet Arduino Fio
	Arduino Nano,	ATmega	328P auf (COM5		Arduino BT LilyPad Arduino USB

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

💿 Boardverwalter	>
Typ Alle SP32	
esp32 by Espressit Systems In diesem Paket enthalten orards: ESP32 Daw Module, MariOS LoLin32. More info	Installieren
	Schließe

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

💿 sketch_nov07a Arduin	io 1.8.3 -	- (×	
Datei Bearbeiten Sketch We	erkzeuge Hilfe				
	Automatische Formatierung	Strg+T			
sketch nov07a	Sketch archivieren				Arduino Yún Mini
sketen_novoru	Controller Maniferen & neu lader	1 Charles			Arduino Industrial 101
1 void setup() {	Serieller Monitor	Strg+U	mscn	alt+M	Linino One
3	Serieller Plotter	Strg+L	msch	alt+L	Arduino Uno WiFi
4 }	WiFi101 Firmware Updater				ESP32 Arduino
5	Board: "Arduino Nano"			>	TTGO LoRa32-OLED V1
7 // put your	Prozessor: "ATmega328P"			>	ESP32 Dev Module
8	Port				XinaBox CW02
9 }	Boardinformationen holen				ESP32 Wrover Module
	boardimormationen noien				ESP32 Pico Kit
	Programmer: "AVRISP mkII"			>	SparkFun ESP32 Thing
	Bootloader brennen				u-blox NINA-W10 series (ESP32)
					Widora AIR
					Electronic SweetPeas - ESP320
					Nano32
	Arduino Nano, AT	rmega328	P auf (COM5	LOLIN D32
					LOLIN D32 PRO

💿 sketch_nov07a Arduino 1.8.3	- 🗆	×			
Datei Bearbeiten Sketch Werkzeuge Hilfe					
		<u>19-</u>			
sketch_nov07a	i i	-			
1 ¹¹ void setup() {		^			
<pre>2 // put your setup code here, to 3</pre>	^{run once:} ESP32 De	v Module, Disabled,	Default, QIO,	80MHz, 4MB (33	2Mb), 921600
4 }					
5					
6 void loop() {					
7 // put your main code here, to p	run repeatedly:				
8					
9 }					

7. Done! ઉ

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.

Codewerk GIT	PC/Mac Arduino
$coderacer_hackathon_x-day \ \land \ Arduino \ \land \ libraries$	Dieser PC > Dokumente > Arduino > libraries
Name ^	Name
📕 CodeRacer	readme.txt
ESP32Servo	ESP32Servo
	CodeRacer

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

		SPIFFS	>			
💿 sketch_nov08a A	rduino 1.8.3	Ticker	>	_		×
Datei Bearbeiten Skete	ch Werkzeuge Hilfe	Update	>			
Neu Öffnen	Strg+N Strg+O	WebServer WiFi	>			<mark>,0</mark> ,
Letzte öffnen Sketchbook	>	WiFiClientSecure Beispiele aus eigenen Biblio	> otheken			•
Beispiele	>	CodeRacer	>	esp32	coderac	er_beispie
Schließen Speichern Speichern unter	Strg+W Strg+S Strg+Umschalt+S	ESP32Servo	>			
Seite einrichten Drucken	Strg+Umschalt+P Strg+P	e, to run repeatedly:				
Voreinstellungen	Strg+Komma					~
Beenden	Strg+Q					
	20500					
	ESP32 D	Dev Module, Disabled, Default, QIO, 80M	IHz, 4MB (32MI	b), 921600,	None auf (COM5

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

Upload auf den {code}.racer

- 1. Das Board korrekt auf ,ESP32 Dev Module ' einstellen.
- 2. Den ESP32 auf dem {code}.racer per μ 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 mehere Ports gelistet – einfach mal den {code}.racer ab- und wieder einstecken.



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

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



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