

# Projektarbeit – Ferngesteuertes Auto

TEKO Zürich, 20. Januar 2025

5.Semester



**Fach:**  
**Mikrocomputertechnik**

**Dozent:**  
**Christian Meier**

**Projektleiter:**  
**Noé Hiltbrunner/ Festim Elmazi**

20. Januar 2025 / Version 1.0

# Inhaltsverzeichnis

<b>1</b>	<b>Vorwort.....</b>	<b>3</b>
<b>2</b>	<b>Terminprogramm / Teamorganisation .....</b>	<b>3</b>
<b>3</b>	<b>Pflichtenheft Aufgabenstellung .....</b>	<b>5</b>
<b>4</b>	<b>Elektronik Schema .....</b>	<b>6</b>
<b>5</b>	<b>Aufbau Ferngesteuertes Auto.....</b>	<b>7</b>
<b>6</b>	<b>Flussdiagramm.....</b>	<b>10</b>
<b>7</b>	<b>Arduino Code Modulasierung/ Regeln/ Cleancode.....</b>	<b>11</b>
<b>8</b>	<b>Material/ Hadware .....</b>	<b>15</b>
<b>9</b>	<b>Inbetriebnahme &amp; Test-Protokoll .....</b>	<b>16</b>
<b>10</b>	<b>SOLL-/ IST Abgleich (Fazit) .....</b>	<b>16</b>
<b>11</b>	<b>Quellenverzeichnis.....</b>	<b>17</b>

# **1.Vorwort**

Im Fach Mikrocomputertechnik an der TEKO Zürich haben die Studierenden im 5. Semester die Möglichkeit, eine Projektarbeit zu erstellen.

Diese kann entweder im Team (2er Teams) oder als Einzelarbeit durchgeführt werden.

Die Projektarbeit besteht aus einer ausführlichen schriftlichen Beschreibung sowie einer Präsentation vor Abgabe der Dokumentation.

Unser Ziel war es, etwas Kreatives in diesem vielseitigen Fachgebiet zu entwickeln, da es viele spannende und praxisnahe Arbeitsmöglichkeiten bietet.


## **2.Terminprogramm/ Teamorganisation**

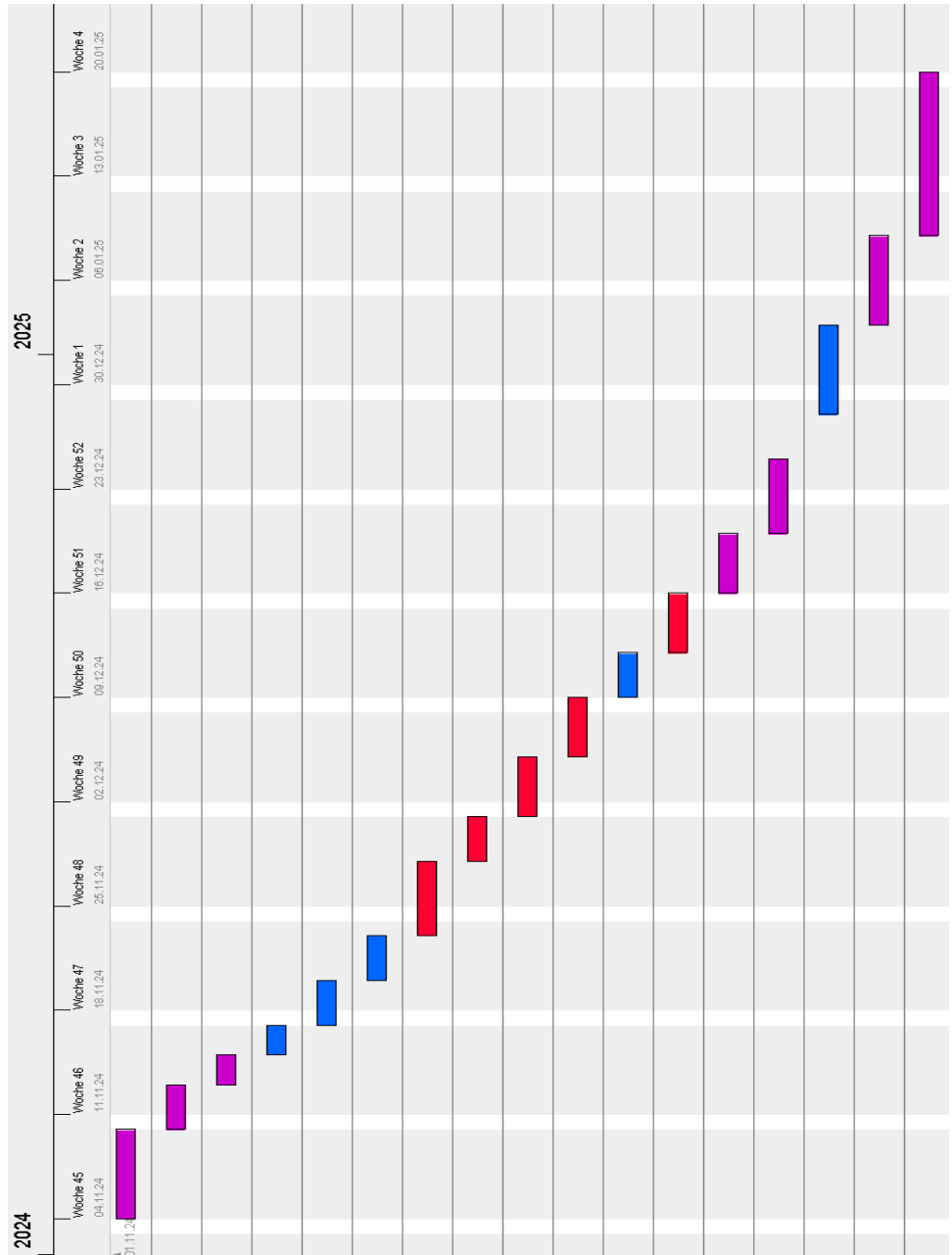
Um die Projektarbeit im Team klar und sauber aufzuteilen, haben wir alle Arbeiten entsprechend im Zeitplan markiert. Meistens haben wir versucht, die Arbeiten immer im Team zu machen, damit beide wussten, wie es vom Anfang bis zum Ende geht. Aber für den Anfang war Noé Hiltbrunner für den technischen Teil und Festim Elmazi für den administrativen Teil zuständig. Die Termine wurden farblich gekennzeichnet, damit man sehen konnte, wer wofür und in welchem Zeitrahmen zuständig war.

Blau= Festim Elmazi

Rot= Noé Hiltbrunner

Violet= Zusammen (Teamorganisation)

			
	Vorgang	Anfang	Ende
	Ideen gesammelt	04.11.24	09.11.24
	Material Bestellung	10.11.24	12.11.24
	Arbeitsaufteilung	13.11.24	14.11.24
	Terminprogramm erstellt	15.11.24	16.11.24
	Dokumentation gestartet	17.11.24	19.11.24
	Pflichtenheft erstellt	20.11.24	22.11.24
	Elektronik-Schema erstellt	23.11.24	27.11.24
	Grundriss gezeichnet	28.11.24	30.11.24
	Projekt zusammengebaut	01.12.24	04.12.24
	Software gestartet	05.12.24	08.12.24
	Dokumentation bearbeitet	09.12.24	11.12.24
	Software bearbeitet	12.12.24	15.12.24
	Hardware aufgebaut	16.12.24	19.12.24
	Software fertig erstellt	20.12.24	24.12.24
	Flussdiagramm sämtlicher SW-Module	28.12.24	02.01.25
	Hardware Test	03.01.25	08.01.25
	Dokumentation fertig erstellt	09.01.25	19.01.25



### 3. Pflichtenheft Aufgabenstellung

Wir wollen ein ferngesteuertes Auto mit Arduino-Technologie bauen. Nach vielen Ideen hat uns dieses Konzept am meisten begeistert.

Das Auto wird auf einer selbst entworfenen Basis gebaut. Dazu erstellen wir mit der Software Trimble Nova 18 einen Grundriss. Als Material verwenden wir Holz, da es leicht und gut zu bearbeiten ist.

Das Fahrzeug wird mit vier Motoren ausgestattet, einer pro Rad. Die Steuerung erfolgt über eine Arduino-Fernbedienung, deren Tastenbelegung wir definieren und anpassen, um eine präzise Bedienung zu gewährleisten.

Folgende Funktionen sollen realisiert werden:

Vorwärts- und Rückwärtsfahren: Die Grundbewegungen des Autos.

Stopp: Eine Taste, die das Auto sofort zum Stehen bringt.

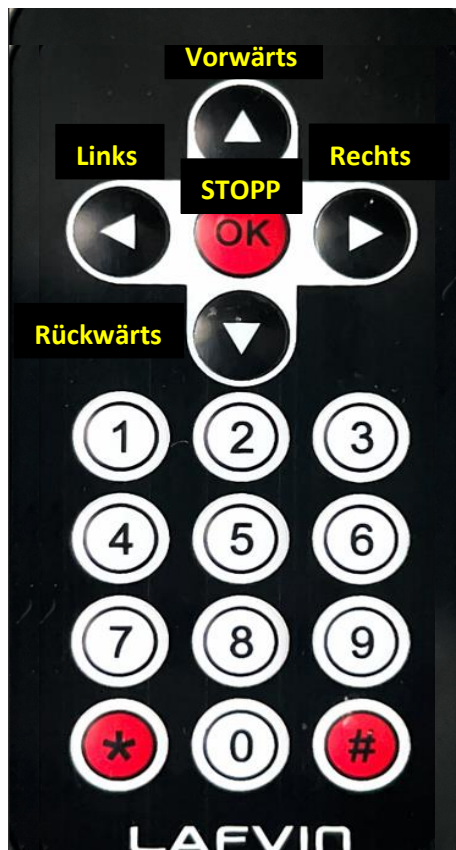
Seitliche Positionierung: Zwei weitere Tasten ermöglichen es, das Auto im Stillstand nach rechts oder links auszurichten.

Besonderheiten:

Die Räder sind nicht drehbar. Richtungsänderungen werden nur durch die Geschwindigkeitsänderung der einzelnen Motoren erreicht.

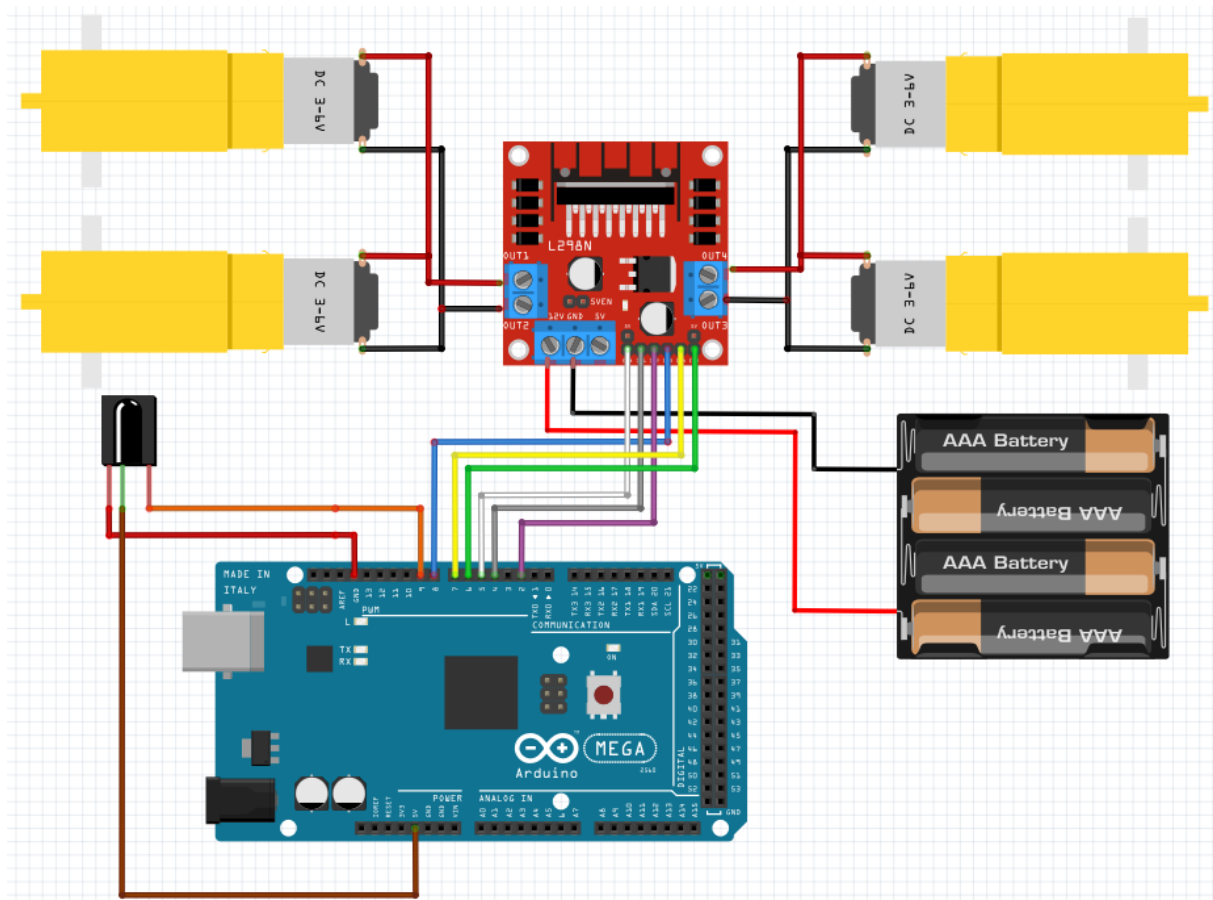
Ziel ist es, ein Auto zu entwickeln, das sich stabil, präzise und effizient steuern lässt.

Nachfolgend eine Übersicht über die Fernbedienung und die Funktionen der einzelnen Tasten.



## 4. Elektronik Schema

Die Verkabelung wurde anhand eines Elektronikschemas durchgeführt. Der Aufbau erfolgte entsprechend diesem Schema.

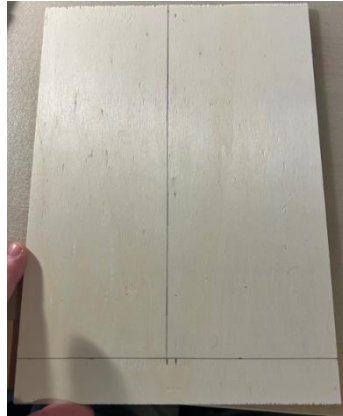


Wichtig ist, dass die Batterie, die auch einen Netzstecker verfügt, mit dem Arduino Mega 2560 verbunden wird, damit das Auto auch ohne Kabelverbindung zum Laptop betrieben werden kann.

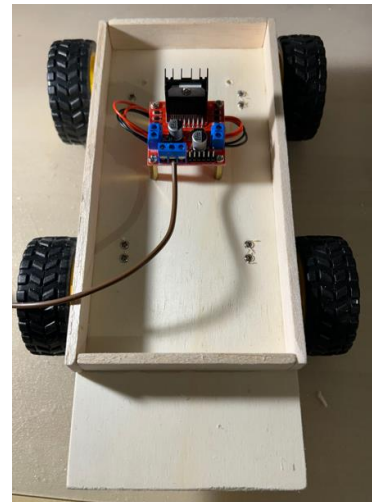
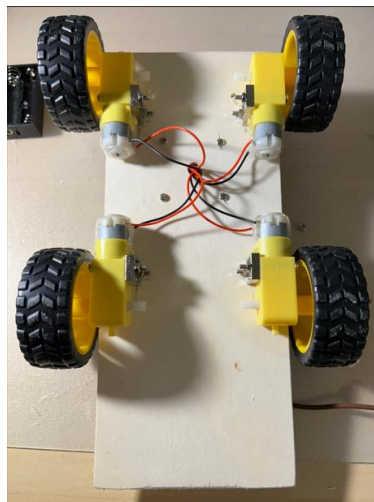
SBC-MotoDriver 2	Batterie	DC Motoren Links	DC Motoren Rechts	Arduino Mega 2560	IR Sensor
12V	X			X	
GND	X			X	
OUT1		X (Rot +)			
OUT2		X (Schwarz -)			
OUT3			X (Schwarz -)		
OUT4			X (Rot +)		
				GND	X (G)
				5V	X (R)
ENA				X (PIN 5)	
IN1				X (PIN 4)	
IN2				X (PIN 2)	
IN3				X (PIN 8)	
IN4				X (PIN 7)	
ENB				X (PIN 6)	
				PIN 9	X (Y)

## 5. Aufbau Ferngesteuertes Auto

Um sicherzustellen, dass das Auto groß genug, aber nicht zu groß wird, mussten wir zunächst den Platzbedarf für die gesamte Elektronik ermitteln. Nachfolgend finden Sie die Grundrisspläne sowie eine Übersicht der praktischen Schritte, die wir durchgeführt haben, ergänzt durch kurze Beschreibungen.

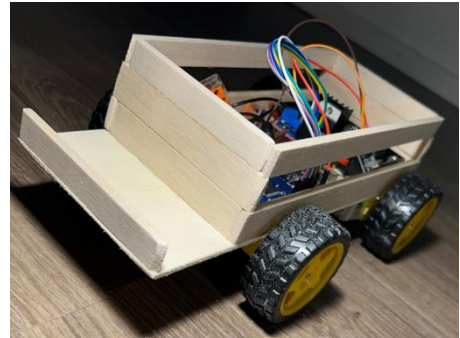
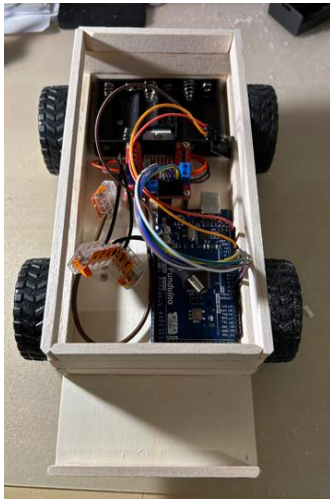


Am Anfang haben wir gewöhnliches Holz gekauft, wobei wir darauf geachtet haben, dass es so leicht wie möglich ist, damit das Auto keine Probleme beim Fahren hat. Das Holz hat eine Dicke von 4 mm. Darauf haben wir die Maße für den Unterboden des Autos aufgezeichnet. Mit einer Stichsäge haben wir das Holz genau zugeschnitten und anschließend die Kanten mit Schleifpapier sauber bearbeitet.

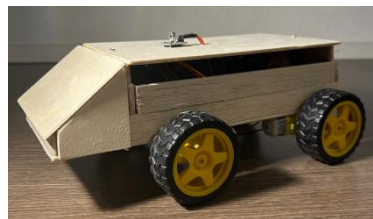


An den Seiten des Unterbodens haben wir zusätzlich Holz angeklebt, damit die verdrahtete und installierte Elektronik nicht herausfallen kann. Anschließend haben wir die Gleichstrommotoren seitlich montiert und verkabelt. Dazu wurde ein kleines Loch gebohrt, durch das die Kabel geführt wurden, die dann mit dem SBC-MotoDriver 2 Entwicklungsboard verbunden wurden.



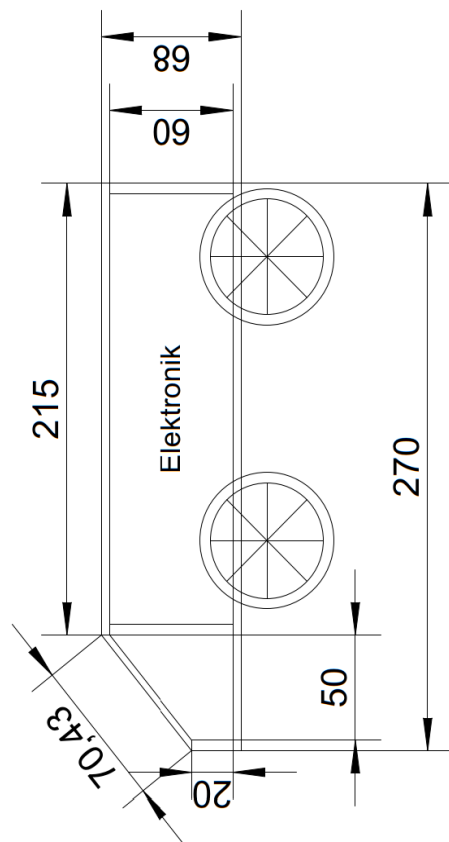
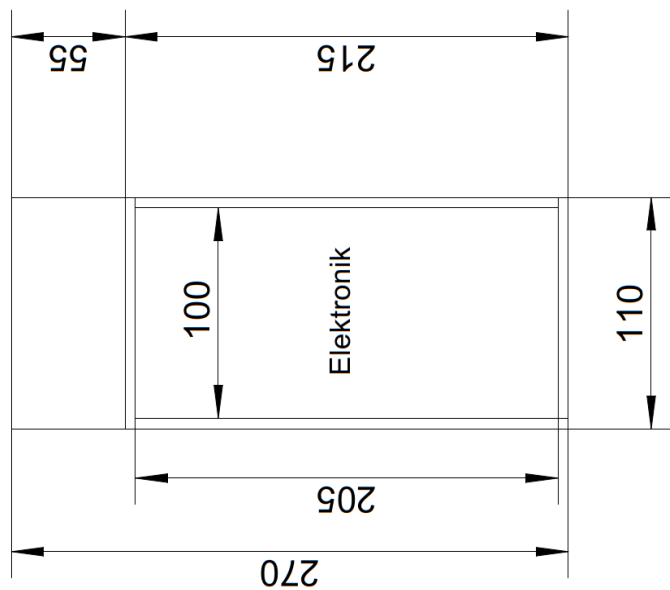


Anschließend haben wir die gesamte Elektronik eingebaut und verkabelt. Um die Batterie mit dem Arduino Mega 2560 und dem Entwicklungsboard SBC-MotoDriver 2 zu verbinden, haben wir den Netzstecker der Batterie durchtrennt und die Verbindungen mit Wago-Klemmen hergestellt. Nach der Verkabelung konnten wir erste Funktionstests durchführen.



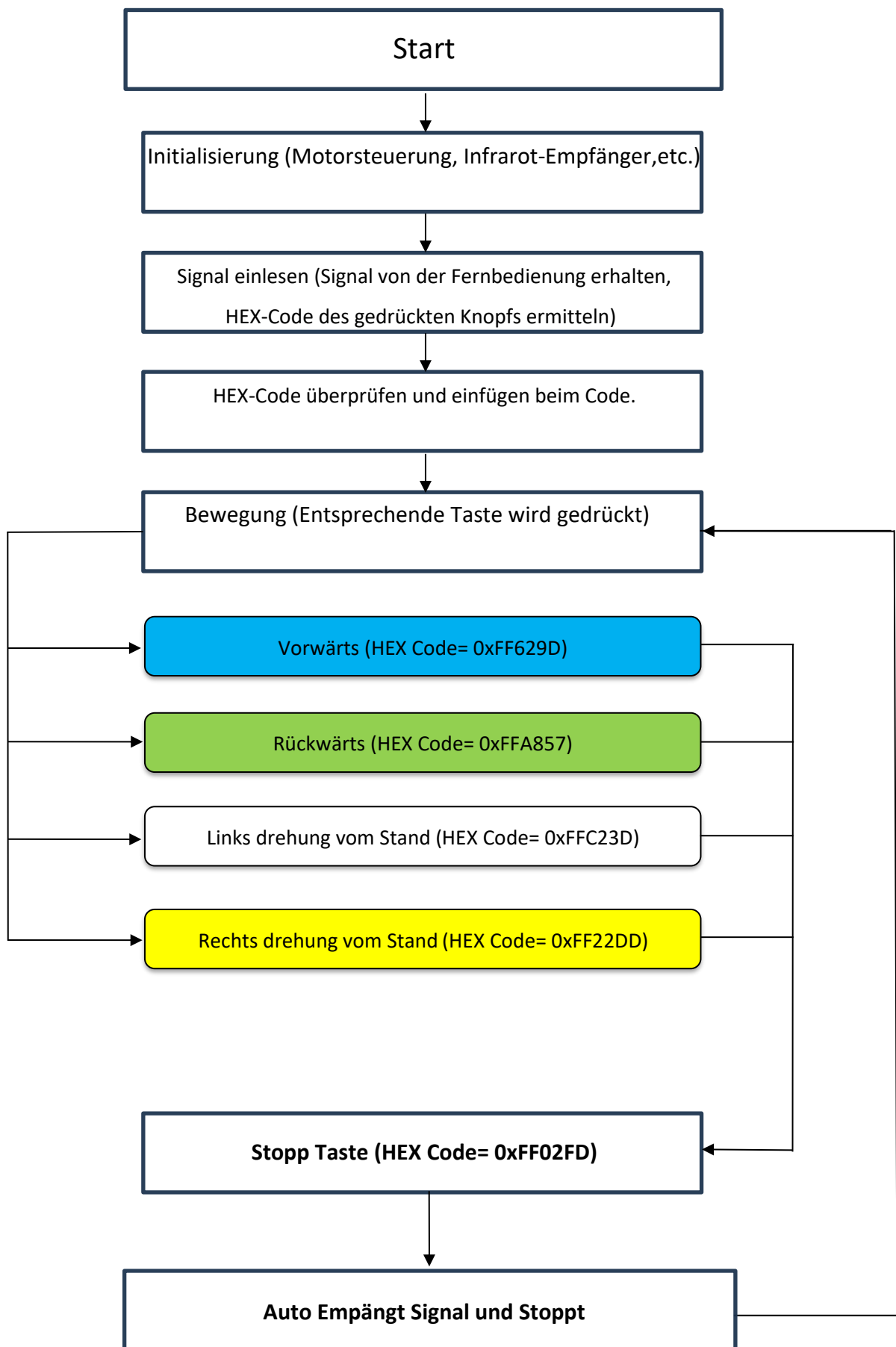
Schließlich haben wir ein Dach auf das Auto montiert, um die gesamte Elektronik abzudecken und zu schützen. Gleichzeitig ermöglicht es die Konstruktion, das Dach bei Bedarf einfach abzuschrauben, falls Änderungen vorgenommen werden müssen. Auf dem Dach wurde auch der IR-Sensor installiert, damit er die Signale der Fernbedienung optimal empfangen kann.





		Projekt-Nr.: 12240807		Plan-Nr.: 5		Fertiggestelltes Auto TEKO Mikrocomputertechnik		Dispo
Rev.	Änderung	Datum	Gez.	Gepr.	Datum	Gezeichnet:	mhi	
						Geprüft:	mhi	
						Datum:	30.11.2024	
						Maßstab:	1:50	

## 6. Flussdiagramm




## 7. Arduino Code Modulasierung/ Regeln/ Cleancode

Der Code ist so aufgebaut, dass das Fahrzeug mit einer IR-Fernbedienung gesteuert wird. Die IRremote-Bibliothek liest die Signale der Fernbedienung und die Pin-Definitionen legen fest, wie die Motoren und der IR-Empfänger angeschlossen werden. Im Setup-Abschnitt werden die Motorsteuerepins als Ausgänge konfiguriert und der IR-Empfänger aktiviert. Die Loop-Funktion prüft ständig, ob ein Signal von der Fernsteuerung empfangen wird. Abhängig von diesem Signal wird das Fahrzeug vorwärts, rückwärts, nach links oder rechts gesteuert oder gestoppt. Die Motoren werden mit PWM angesteuert, um die Geschwindigkeit zu regeln. Zusätzlich sendet der Code Statusmeldungen über die serielle Schnittstelle, um den aktuellen Zustand zu überwachen.

Bei der Erstellung des Codes wurden wir von ChatGPT unterstützt, insbesondere bei der Klärung technischer Details und logischen Strukturen. Zusätzlich wurden GitHub Vorlagen und Code von Drittanbietern in unsere Software integriert, um bewährte Ansätze für die Fahrzeugsteuerung und die Verwendung von IR-Fernbedienungen effizient in das Projekt zu integrieren. Quellenangaben sind im Quellenverzeichnis verlinkt.

Der beigefügte Teil des Codes definiert die grundlegenden Anschlüsse und Einstellungen für ein Fahrzeug, das mit einer IR-Fernbedienung gesteuert wird. Es folgt eine einfache Erklärung:



```
1 #include <IRremote.h>
2
3 // Pin-Definitionen für IR-Empfänger und Motorsteuerung
4 #define IR_PIN 9 // Pin für IR-Empfänger
5 #define Lpwm_pin 5 // PWM für linke Motoren
6 #define Rpwm_pin 6 // PWM für rechte Motoren
7 #define pinLB 2 // Richtung: Linke Motoren rückwärts
8 #define pinLF 4 // Richtung: Linke Motoren vorwärts
9 #define pinRB 7 // Richtung: Rechte Motoren rückwärts
10 #define pinRF 8 // Richtung: Rechte Motoren vorwärts
11
12 IRrecv irrecv(IR_PIN); // IR-Empfänger initialisieren
13 decode_results results;
14
15 // Variablen für den Fahrzustand
16 bool moving_forward = false;
17 bool moving_backward = false;
```

- #include <IRremote.h>= Damit der Arduino die Signale der IR-Fernbedienung empfangen und verstehen kann, wird diese Bibliothek benötigt.

- Pin-Definitionen für den IR-Empfänger und die Motorsteuerung hier legen wir die Namen für die Pins auf dem Arduino fest:

IR\_PIN= Der IR-Empfänger wird an Pin 9 angeschlossen.

Lpwm\_PIN und Rpwm\_PIN= Steuert die Geschwindigkeit des linken und rechten Motors.

pinLB, pinLF, pinRB, pinRF= Steuert die Drehrichtung der Motoren (vorwärts/rückwärts).

-IR-Empfänger initialisieren:

Der IR-Empfänger ist so eingestellt, dass er Signale von Pin 9 empfängt."results" speichert die empfangenen Signale.

-Variablen für den Fahrzustand: Diese Variablen speichern, ob das Auto vorwärts oder rückwärts fährt.

```
Arduino Mega or Mega 2560
sketch_Auto.ino
18
19 // Funktionen für Bewegungen
20 void go_forward(unsigned char speed_val) {
21   moving_forward = true;
22   moving_backward = false;
23
24   // Motor-Richtung einstellen
25   digitalWrite(pinLB, HIGH);
26   digitalWrite(pinLF, LOW);
27   digitalWrite(pinRB, HIGH);
28   digitalWrite(pinRF, LOW);
29
30   // Feinjustierung der Geschwindigkeiten
31   unsigned char left_motor_speed = speed_val; // Geschwindigkeit für linken Motor
32   unsigned char right_motor_speed = speed_val - 10; // Geschwindigkeit für rechten Motor leicht reduziert
33
34   analogWrite(Lpwm_pin, left_motor_speed);
35   analogWrite(Rpwm_pin, right_motor_speed);
36
37   Serial.println("Fahre vorwärts...");
38 }
```

-Die Funktion "go\_forward" steuert die Motoren, damit das Auto vorwärts fährt. Sie legt die Richtung und Geschwindigkeit der Motoren fest. So fährt das Auto möglichst geradeaus. Die Debug-Nachricht hilft, den Status während des Tests zu überwachen.

-Motor-Richtung einstellen: Die Motoren werden so eingestellt, dass das Auto fährt. "HIGH" bedeutet "Strom fließt" und "LOW" bedeutet "kein Strom fließt".

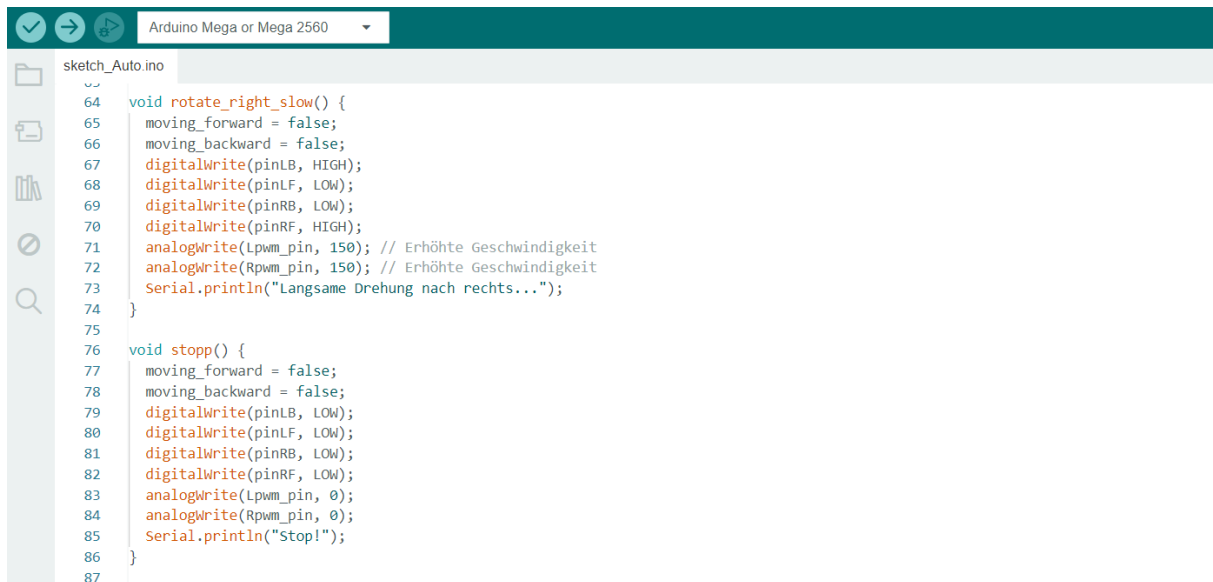
-Feinjustierung der Geschwindigkeiten: Der linke Motor fährt genauso schnell weiter, der rechte etwas langsamer (speed\_val - 10), damit das Auto geradeaus fährt, wenn die Motoren nicht gleich schnell sind.

```
Arduino Mega or Mega 2560
sketch_Auto.ino
38 }
39
40 void go_backward(unsigned char speed_val) {
41   moving_forward = false;
42   moving_backward = true;
43   digitalWrite(pinLB, LOW);
44   digitalWrite(pinLF, HIGH);
45   digitalWrite(pinRB, LOW);
46   digitalWrite(pinRF, HIGH);
47   analogWrite(Lpwm_pin, speed_val);
48   analogWrite(Rpwm_pin, speed_val);
49   Serial.println("Fahre rückwärts...");
50 }
51
52 void rotate_left_slow() {
53   moving_forward = false;
54   moving_backward = false;
55   digitalWrite(pinLB, LOW);
56   digitalWrite(pinLF, HIGH);
57   digitalWrite(pinRB, HIGH);
58   digitalWrite(pinRF, LOW);
59   analogWrite(Lpwm_pin, 150); // Erhöhte Geschwindigkeit
60   analogWrite(Rpwm_pin, 150); // Erhöhte Geschwindigkeit
61   Serial.println("Langsame Drehung nach links...");
62 }
63 }
```

-Die Funktion "go\_backward" fährt das Auto rückwärts, indem die Richtung der Motoren angepasst wird.

-Die Funktion "rotate\_left\_slow" dreht das Auto langsam nach links, indem die Motoren in entgegengesetzte Richtungen laufen. Beide Funktionen geben zusätzlich Statusmeldungen aus, um die Aktionen zu überwachen.

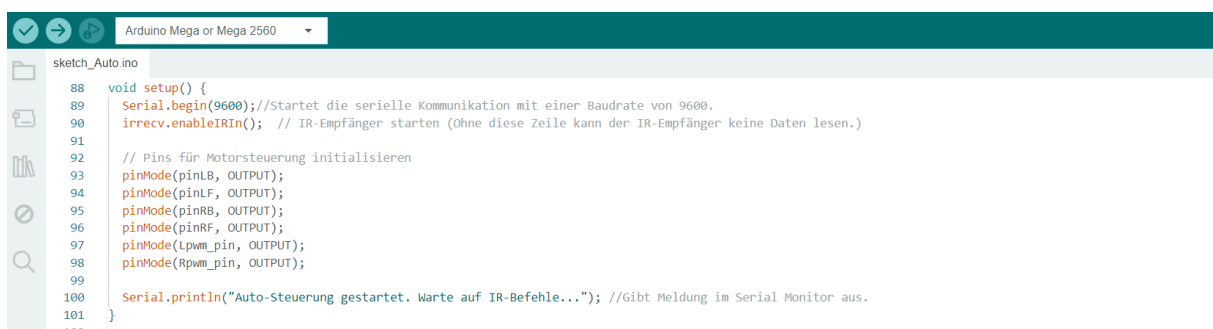
-Die Befehle "analogWrite(Lpwm\_pin, 150)" und "analogWrite(Rpwm\_pin, 150)" sorgen dafür, dass beide Motoren mit einer festen Geschwindigkeit laufen (PWM-Wert 150).



```
64 void rotate_right_slow() {
65   moving_forward = false;
66   moving_backward = false;
67   digitalWrite(pinLB, HIGH);
68   digitalWrite(pinLF, LOW);
69   digitalWrite(pinRB, LOW);
70   digitalWrite(pinRF, HIGH);
71   analogWrite(Lpwm_pin, 150); // Erhöhte Geschwindigkeit
72   analogWrite(Rpwm_pin, 150); // Erhöhte Geschwindigkeit
73   Serial.println("Langsame Drehung nach rechts...");
74 }
75
76 void stopp() {
77   moving_forward = false;
78   moving_backward = false;
79   digitalWrite(pinLB, LOW);
80   digitalWrite(pinLF, LOW);
81   digitalWrite(pinRB, LOW);
82   digitalWrite(pinRF, LOW);
83   analogWrite(Lpwm_pin, 0);
84   analogWrite(Rpwm_pin, 0);
85   Serial.println("Stop!");
86 }
87
```

-Mit "rotate\_right\_slow" wird das Auto nach rechts gedreht, indem der linke Motor vorwärts und der rechte Motor rückwärts läuft.

-Mit "stopp" wird das Auto vollständig gestoppt, indem die Motoren deaktiviert und ihre Geschwindigkeit auf 0 gesetzt wird.



```
88 void setup() {
89   Serial.begin(9600); //Startet die serielle Kommunikation mit einer Baudrate von 9600.
90   irrecv.enableIRIn(); // IR-Empfänger starten (Ohne diese Zeile kann der IR-Empfänger keine Daten lesen.)
91
92   // Pins für Motorsteuerung initialisieren
93   pinMode(pinLB, OUTPUT);
94   pinMode(pinLF, OUTPUT);
95   pinMode(pinRB, OUTPUT);
96   pinMode(pinRF, OUTPUT);
97   pinMode(Lpwm_pin, OUTPUT);
98   pinMode(Rpwm_pin, OUTPUT);
99
100   Serial.println("Auto-Steuerung gestartet. Warte auf IR-Befehle..."); //Gibt Meldung im Serial Monitor aus.
101 }
102
```

-Die setup-Funktion richtet die serielle Kommunikation ein, aktiviert den IR-Empfänger und konfiguriert die Pins für die Motorsteuerung als Ausgänge. Zusätzlich wird eine Debug-Nachricht ausgegeben, um zu bestätigen, dass das Auto bereit ist, auf IR-Befehle zu reagieren. Dies ist die Basis, damit die Steuerung des Autos funktioniert.

-pinMode legt fest, dass die Pins als Ausgänge (OUTPUT) benutzt werden.

Diese Pins steuern, in welche Richtung die Motoren gehen und wie schnell sie sind:

pinLB und pinLF: Links-Motoren

pinRB und pinRF: Rechts-Motoren

Lpwm\_pin und Rpwm\_pin: PWM-Geschwindigkeit der Motoren

```

102
103 void loop() {
104     if (irrecv.decode(&results)) {
105         switch (results.value) {
106             case 0xFF629D: // Vorwärts
107                 go_forward(255); // Maximale Geschwindigkeit
108                 break;
109             case 0xFFA857: // Rückwärts
110                 go_backward(255); // Maximale Geschwindigkeit rückwärts
111                 break;
112             case 0xFF02FD: // Stop
113                 stopp();
114                 break;
115             case 0xFFC23D: // Langsame Drehung nach links
116                 rotate_left_slow();
117                 break;
118             case 0xFF22DD: // Langsame Drehung nach rechts
119                 rotate_right_slow();
120                 break;
121             default:
122                 Serial.println("Unbekanntes Signal. Stoppe Motoren.");
123                 stopp();
124                 break;
125         }
126         irrecv.resume(); // IR-Empfänger bereit für das nächste Signal
127     }
128 }
129

```

-Die Loop-Funktion prüft ständig, ob die IR-Fernbedienung ein Signal empfangen hat. Je nachdem wird das Auto vorwärts, rückwärts, nach links oder nach rechts gesteuert oder gestoppt.

Wenn das Signal unbekannt ist, wird das Auto sicherheitshalber angehalten. Dann ist der IR-Empfänger bereit, um neue Signale zu empfangen.

-Durch Drücken der Taste mit dem Signal 0xFF629D wird das Fahrzeug mit maximaler Geschwindigkeit (255) vorwärts bewegt. Zunächst musste der HEX-Code durch Probieren der Fernbedienung herausgefunden werden. Anschließend konnte dieser bei den entsprechenden Befehlen eingesetzt werden.

-Mit "irrecv.resume();" wird die Reset-Funktion des IR-Empfängers aktiviert, um den Empfang des nächsten Signals zu ermöglichen.

## 8. Material/ Hardware

Komponenten	Anzahl	Beschreib
 <a href="#">Link: Einwellen Getriebemotor mit Kabel 3-6VDC</a>	4Stk.	Einwellen-Getriebemotor mit Kabel 3-6VDC
 <a href="#">Link: Auto Rad passend mit Getriebemotor</a>	4Stk.	Auto Rad-Passend mit Getriebemotor
 <a href="#">Link: Joy-it SBC-MotoDriver 2 Entwickler-Platine</a>	1Stk.	Joy-it SBC-MotoDriver 2 Entwickler-Platine
 <a href="#">Link: IR Fernbedienung</a>	1Stk.	IR Fernbedienung
 <a href="#">Link: IR-Sensor</a>	1Stk.	IR-Sensor
 <a href="#">Link: Jumper-Kabel</a>	1Stk.	Jumper-Kabel
 <a href="#">Link: Arduino Mega 2560</a>	1Stk.	Arduino Mega 2560
 <a href="#">Link: 6AA Batterie mit Netzstecker</a>	1Stk.	6AA Batterie mit Netzstecker



## 9. Inbetriebnahme & Test-Protokoll

Testprotokoll	Ja	Nein
Spannung angeschlossen?	X	
Ground angeschlossen?	X	
Auto fährt Vorwärts?	X	
Auto fährt Rückwärts?	X	
Auto dreht sich nach links?	X	
Auto dreht sich nach rechts?	X	
Auto STOPPT?	X	

## 10. SOLL-/ IST Abgleich (Fazit)

Das Projekt "Ferngesteuertes Auto" wurde erfolgreich abgeschlossen. Die gesteckten Ziele wurden erreicht und das Auto funktioniert wie geplant. Die Fernsteuerung arbeitet zuverlässig und der Grundriss des Autos wurde exakt wie vorgesehen umgesetzt. Zu Beginn des Projekts gab es einige Herausforderungen, insbesondere bei der Konfiguration der Steuerung und der Zuordnung der Tasten für die Fahrzeuglenkung.

Die Teamarbeit war ein voller Erfolg. Das liegt daran, dass wir alles gut geplant haben. So gab es keinen Stress. Wir hatten einen detaillierten Zeitplan und sind diszipliniert geblieben. Das hat zum Erfolg des Projekts beigetragen.

## 11. Quellenverzeichnis

- [ChatGPT](#) (Code Hilfe und Rechtschreibung)
- [IR-Car-Arduino/ir car.ino at main · rcaerbannog/IR-Car-Arduino · GitHub](#)
- [Fernbedienung - Arduino Tutorial - Einfach & Verständlich](#)
- [Arduino Auto bauen + Anleitung + Code | Arduino-Garten.de](#)
- [RC-Car gesteuert durch IR-Fernbedienung - International / Deutsch - Arduino Forum](#)