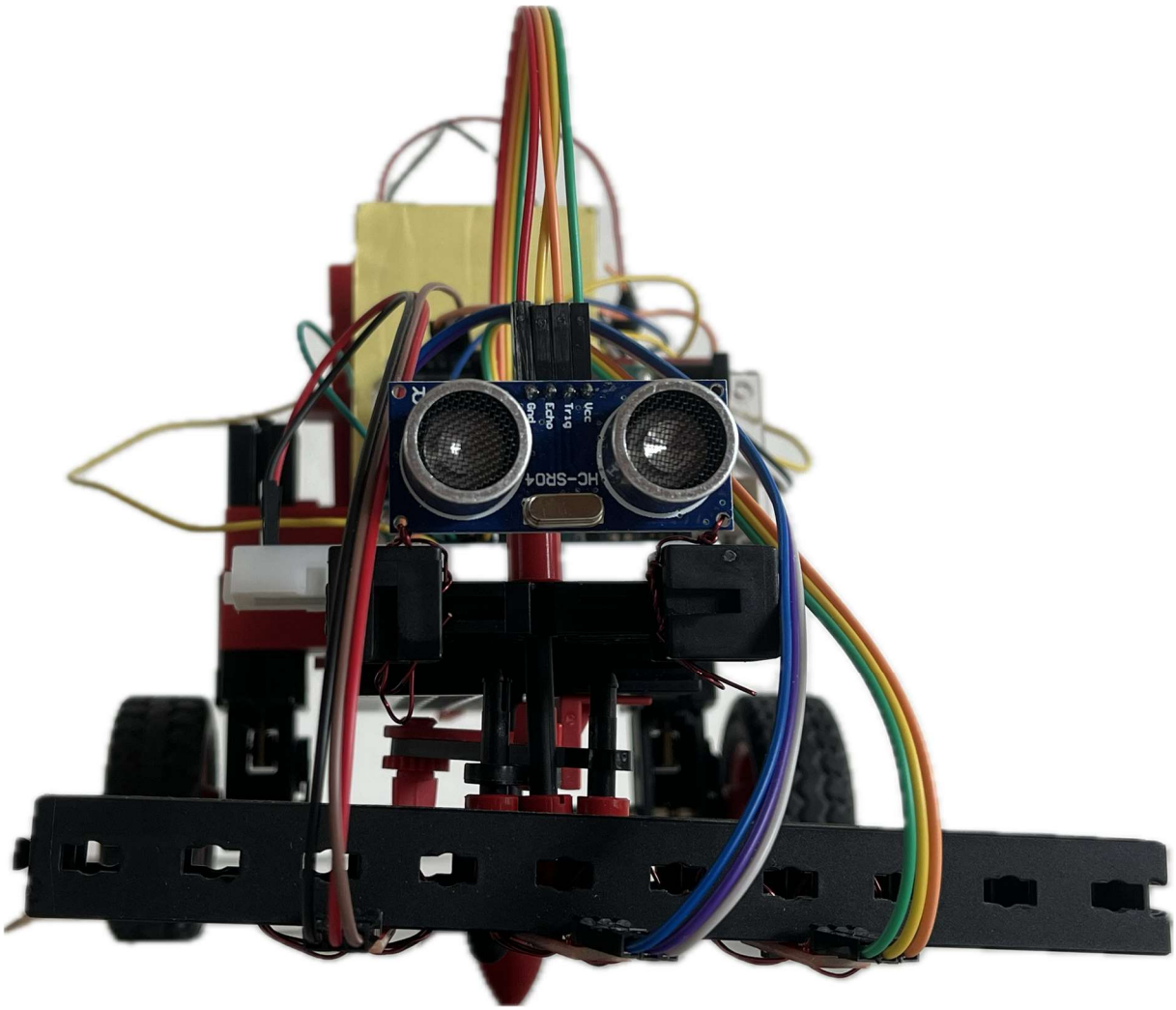
Wie kann man autonomes Fahren mit Arduino und Fischertechnik simulieren?

`



Franziskanergymnasium Kreuzburg

Autoren: Jakob Hollborn, Paul Maier

Betreuerin: Frau Meininger

Großkrotzenburg, den 28.03.2025

Inhaltsverzeichnis

1. Einleitung

Wir haben ein Modellauto gebaut und programmiert, welches links, rechts, geradeaus und rückwärts fahren, Linien folgen und Hindernissen ausweichen kann. Dazu mehr im Forscherteil. Außerdem haben wir recherchiert, was autonomes Fahren ausmacht. Dazu mehr im Theorieteil.

Wie sind wir auf das Thema Autonomes Fahren gestoßen? Jakob hatte in den Herbstferien 2024 einen Workshop in Frankfurt, in dem er einen mBot programmiert hat, und wir fanden das beide interessant. Ein mBot ist ein Roboterfahrzeug, das man mit einer Programmiersprache wie Scratch (Bunte Codeblöcke aneinanderschieben) programmieren kann.

Was genau wollen wir untersuchen? Wir wollen in dieser Forscherarbeit die Vorteile und Nachteile des Autonomen Fahrens aufzeigen. Und wollen klären, wie man einige von den Nachteilen löst.

Das Klären dieser Frage ist für alle Personen auf dieser Welt wichtig, da es bald überall Autonome Autos geben könnte und diese vielleicht bald zu unserem normalen Alltag dazugehören.

Das Thema ist aktuell, da seit 20 Jahren gesagt wird, dass es in 2 Jahren Autonomes Fahren öffentlich gibt, es ist aber noch nichts passiert, weil die Forschung meint es kommt und es kommt, aber es gibt noch zu viele Probleme.

Wir müssen das Thema Autonomes Fahren einschränken, auf die Fünf Level zum Autonomen Fahren, was ein Auto können muss, die Vor- und Nachteile des Autonomen Fahrens und ob Autonomes Fahren wirklich sinnvoll ist. Es gibt zu viele Aspekte, um alle in unsere Forscherarbeit aufnehmen zu können.

2.Theorieteil: Autonomes Fahren

2.1 Die sechs Level zum Autonomen Fahren

[1]Level 0 heißt: Keine Automatisierung, also fährt der Fahrer vollständig.

Level 1 heißt: Es ist assistiert und zeitweise ohne Füße, also verantwortet der Fahrer die Längs- und Querführung dauerhaft und wird in spezifischen Fällen unterstützt zum Beispiel einem Tempomat.

[1] Vgl. Wienerich, Jan: Schritte zum Autonomen Fahren https://www.zf.com/mobile/de/technologies/automated_driving/stories/6_levels_of_automated_driving.html [Stand 14.3 25]

Level 2 heißt: Es ist teilautomatisiert und zeitweise ohne Hände, also muss der Fahrer das System dauerhaft überwachen.

Level 3 heißt: Hochautomatisiert, also auch mit zeitweisen Blicken von der Straße und der Fahrer darf zeitweise fahrfremden Tätigkeiten nachgehen dies ist bisher nur bis 60 km/h erlaubt.

Level 4 heißt: Vollautomatisiert und Aufmerksamkeit aus also, muss der Fahrer sich nicht mehr zum Eingriff bereithalten.

Level 5 heißt: Autonom, der Mensch ist also Passagier, das heißt den Fahrer gibt es nicht mehr. Der Mensch wird vollständig zum Passagier.

Level 0 und 1 wird heutzutage schon im normalen Straßenverkehr genutzt.

2.2 Was muss ein Auto können?

Ein autonomes Fahrzeug muss dauerhaft die Umgebung überblicken und muss den Weg wissen, den es fahren soll. Außerdem muss es von allein fahren, anhalten, lenken, bremsen etc. können.

2.3 Vor- und Nachteile [2]

2.3.1 Vorteile:

Der erste Vorteil ist mehr Sicherheit, denn der Großteil der Unfälle passiert, weil Menschen einen Fehler machen. Ein weiterer Punkt ist mehr Zeit und Komfort, weil man als Passagier Pause machen und sich in aller Ruhe anderen Dingen widmen kann. Der nächste Vorteil ist mehr Effizienz im Verkehr; durch mehr Abstimmung wird der Verkehrsfluss verbessert und es gibt weniger Staus. Ein weiterer Punkt ist die Barrierefreiheit: Autonomes Fahren ist barrierefreier, da Personen, die zurzeit nicht Autofahren können, zum Beispiel durch eine Behinderung, dann wieder mobiler sind. Sie können durch Autonomes Fahren wieder selbst, ohne eine Person, die sie fahren muss, Auto fahren. Das nächste Argument für Autonomes Fahren ist, dass Parkplätze besser genutzt werden, da sich das Auto selbst einen Parkplatz sucht, nachdem die Passagiere ausgestiegen sind, oder direkt zur nächsten Person fährt, die gerade ein Auto braucht. Ein weiterer praktischer Nutzen von Autonomen Fahren ist die Anwendung in geschlossen Systemen, wie zum Beispiel am Flughafen. Auf dem Flughafen können

[2] Vgl. o.A.: Pro und Kontras für Autonomes Fahren; https://www.swarco.com/de/mobilitaet-der-zukunft/autonomes-fahren/autonomes-fahren-vor-nachteile [Stand: 25.3.25]

Autonome Fahrzeuge, dann ohne Fahrer Koffer und andere Waren transportieren. [3]Dies wurde 2023 am Flughafen Frankfurt getestet. Wenn man das Autonome Fahren in einem geschlossenen System anwendet, umgeht man auch Probleme wie den Mischverkehr. So könnten zukünftig Innenstädte Auto frei sein, indem nur Autonome Fahrzeuge in der Stadt unterwegs sind, dies führt zu weniger Verkehr. Ein weiter Punkt ist die Tatsache, dass das Auto auch von mehreren Leuten nacheinander genutzt werden kann, da es eigenständig von der einen zur anderen Person fährt. Zuletzt wäre noch der reduzierte CO_2 Verbrauch zu nennen, durch eine effizientere Fahrweise wird weniger CO_2 ausgestoßen. Ein positiver Nebeneffekt wäre, dass Fahrzeuge selbstständig Parkplätze außerhalb der Städte aufsuchen können. Die Parkplätze innerhalb der Städte können dann durch Grünflächen ersetzt werden.

2.3.2 Nachteile:

Der erste Nachteil ist die technische Entwicklung: es müssten viele Daten in kürzester Zeit übermittelt werden, damit das Fahrzeug sich zurechtfindet. Des Weiteren kann ein Fehler im System einen Unfall verursachen, und da man nicht alles in Tests simulieren kann, muss das System lernfähig sein, also eine Künstliche Intelligenz beinhalten. Das nächste Problem sind die hohen Anschaffungskosten. Da es sehr aufwändig in der Produktion und Forschung ist, werden autonome Autos viel kosten. Eine Lösung dafür wäre die Politik: sie könnte Zuschüsse für autonome Fahrzeuge auf den Weg bringen. So spart der Käufer beim Kauf Geld und autonome Autos werden attraktiver. Ein weiterer Nachteil ist die Datensammlung: Autos, die autonom fahren, müssten immer im Internet sein und würden Daten über den Fahrer sammeln, zum Beispiel über Fahrverhalten und Standortdaten. Ein weiteres großes Problem auf den Straßen wäre der gemischte Verkehr: Es wird zu Beginn der automen Technologie auf den Straßen „normale“ und autonome Autos geben. Das ist ein Problem da die „Normalen“ nicht mit der neuen autonomen Technologie kommunizieren können und dadurch Probleme auftreten würden. Ursache dafür ist, dass nicht alle Fahrzeuge die gleichen Kommunikationssysteme haben. Was auch schwierig sein wird, sind rechtliche Aspekte. Autonome Fahrzeuge müssten anders bewertet werden als normale Autos, da niemand hinter dem Steuer sitzt. Die rechtliche Lage bei Unfällen muss neu geregelt werden. Das letzte Problem ist auch noch der Verlust von Arbeitsplätzen. Nun werden die Fahrzeuge für Unternehmen nicht mehr von Menschen gesteuert, daher fallen die Arbeitsplätze weg, wie z. B. Lkw-Fahrer.

[3] Vgl. Knöss, Kevin: Projekt testet autonome Fahrzeuge für Flughafen https://www.journal-frankfurt.de/journal_news/Urbanes-Frankfurt-58/Frankfurt-University-of-Applied-Sciences-Projekt-testet-autonome-Fahrzeuge-fuer-Flughafen-42889.html [Stand: 28.3.25]

2.4 Fazit: Ist Autonomes Fahren sinnvoll

Die Antwort ist ja und nein. Es gibt Punkte, die dafürsprechen, wie zum Beispiel der Komfort und geregelter Verkehrsfluss. Was dagegenspricht ist zum Beispiel der Verlust von Arbeitsplätzen, wie zum Beispiel bei Transportunternehmen. Zu Beginn werden die autonomen Fahrzeuge durch die neue Technologie sehr teuer sein und dies führt dazu, dass nur wenige Personen sich dies leisten können. Bei der Einführung des Autonomen Fahrens wird es lange Zeit Mischverkehr geben. Die „normalen" Autos müssen in gewisser Weise mit den neuen Autos kommunizieren. Das kann zu Problem führen. Die Frage, ob Autonomes Fahren sinnvoll ist, kann nicht zu 100% beantwortet werden. Es ist eine Herausforderung, die es für Hersteller und Politik zu lösen gilt.

Wir finden Autonomes Fahren ist sinnvoll, aber es müssen die verschiedenen Aspekte gelöst werden.

2.5 Navigation

Möglich für die Navigation wären GPS, und an Stellen, wo dieses nicht empfangen werden kann, zum Beispiel in Tiefgaragen oder in Tunneln, ist Linienfolgen nützlich. GPS ist eine Abkürzung für *Global Positioning System*[4]. Es ist ein von Satelliten gestütztes Ortungssystem. Um zu funktionieren, muss das Gerät, im Fall von autonomem Fahren das Auto, die Signale von mindestens vier Satelliten empfangen, die dauerhaft ihre Position und die Zeit zur Erde senden. Es gibt 30 GPS-Satelliten, die dauerhaft um die Erde kreisen[5]. Das Auto errechnet dann seine eigene Position. Da das aber schwerer zu programmieren ist, nutzen wir nur Linienfolgen. Hier hat das Fahrzeug mindestens drei Sensoren oder eine Kamera, um Linien zu erkennen. Wir nutzen für unser Auto drei Infrarotsensoren. Das Fahrzeug versucht dauerhaft mit dem mittleren Sensor über der Linie, und mit den anderen neben der Linie zu sein. Das heißt, wenn die Linie unter dem linken Sensor ist, muss das Auto nach links lenken, wenn die Linie unter dem rechten Sensor ist, muss das Auto nach rechts lenken, sonst soll es geradeaus fahren.

2.6 Hindernisumfahrung

Hindernisumfahrung und Umgebungswahrnehmung sind wichtig, weil das Fahrzeug sonst jedes Hindernis auf der Straße rammen würde. Möglich dafür wären Kameras, Infrarotsensoren und Ultraschallsensoren.

---

[4] Kerber, Benjamin Niklas Samuel: Wie GPS funktioniert; https://praxistipps.chip.de/wie-funktioniert-gps-einfach-erklaert_41414 [Stand: 24.3.25]
[5] Ohl, Lisa: GPS - diese Bedeutung steckt hinter dem Begriff; https://praxistipps.chip.de/gps-bedeutung-der-abkuerzung-was-heisst-das_137002 [Stand: 24.3.25]

3. Forscherteil

3.1 Material

Wir nutzen für unser Modell einen Arduino Uno R3, den Funduino-Ultraschallsensor HC-SR04, die Motortreiberplatte L293D, zwei Fischertechnik XS Motoren, Arduino-Kabel, einen 9-Volt-Block und andere Fischertechnik-Bauteile.

3.1.1 Arduino, Funduino & BerryBase

Arduino ist eine Marke, welche Mikrocontroller, also kleine Platinen, und Sensoren zum selbst Programmieren herstellt. Funduino und BerryBase sind Marken, die Mikrocontroller und Sensoren herstellen, welche mit Arduino kompatibel sind. Alle Sensoren und Mikrocontroller laufen mit 3,5 Volt bis 5 Volt. Programmmieren kann man die Mikrocontroller aller drei Marken mit der Software Arduino IDE, die Programmiersprache ist abgewandeltes C++ mit Bibliotheken. Bibliotheken sind verschiedene Codes, die von anderen Personen erstellt wurden und den man in seinen Code implementieren kann.

3.1.2 Fischertechnik

Fischertechnik ist eine Marke, die Baukästen für Kinder und Jugendliche herstellt. Die Teile halten durch eine, wie sie Fischertechnik nennt, ‚Zapfen-Nut-Verbindung', zusammen. Die meisten Teile können an allen sechs Seiten zusammengesteckt werden. Fischertechnik stellt auch Mikroelektronik her, aber da diese mit 9 Volt läuft und teurer ist, nutzen wir diese, bis auf die Motoren, nicht.

3.2 Welche Probleme hatten wir

Ein Bauteil, welches Linien mit Infrarotstrahlung Linien folgen soll, hat nicht funktioniert, sondern willkürlich Werte ausgespuckt.

Wie wir diese gelöst haben

Das defekte Bauteil haben wir durch ein Bauteil einer anderen Marke ausgetauscht.

3.3 Was kann unser Fahrzeug und wie gut es funktioniert

Unser Fahrzeug kann geradeaus und rückwärts fahren. Es kann beliebig weit sich auf der Stelle nach rechts und links drehen und kann Hindernissen einer bestimmten Größe ausweichen, indem es mit dem Ultraschallsensor ein Hindernis erkennt, sich 90° nach rechts dreht, kurz geradeaus fährt, nach links dreht, jetzt parallel zum ursprünglichen Weg fährt, sich nach links dreht, genauso lange fährt, wie es am Anfang nach der Rechtsdrehung gefahren ist, dann nach rechts dreht und dann weiter geradeaus fährt, fährt das Fahrzeug auf derselben Linie, wie am Anfang bevor es dem Hindernis ausgewichen ist. Es kann Linien folgen, indem es dauerhaft versucht, nur mit dem mittleren Liniensensor über der Linie zu sein.

Wenn der linke Sensor die Linie erkennt, dann dreht sich das Fahrzeug nach links, bis der mittlere Sensor die Linie erkennt. Wenn der rechte Sensor die Linie erkennt, dreht es sich nach rechts bis der Sensor die Linie erkennt. Sonst fährt es geradeaus.

4. Quellen

Kerber, Benjamin Niklas Samuel: Wie GPS funktioniert; https://praxistipps.chip.de/wie-funktioniert-gps-einfach-erklaert_41414 [Stand: 24.3.25]

Knöss, Kevin: Projekt testet autonome Fahrzeuge für Flughafen; https://www.journal-frankfurt.de/journal_news/Urbanes-Frankfurt-58/Frankfurt-University-of-Applied-Sciences-Projekt-testet-autonome-Fahrzeuge-fuer-Flughafen-42889.html [Stand: 28.3.25

Ohl, Lisa: GPS - diese Bedeutung steckt hinter dem Begriff; https://praxistipps.chip.de/gps-bedeutung-der-abkuerzung-was-heisst-das_137002 [Stand: 24.3.25]

Wienrich Jan: Automatisiertes Fahren: Stufen zum selbstfahrenden Fahrzeug; https://www.zf.com/mobile/de/technologies/automated_driving/stories/6_levels_of_automated_driving.html [Stand: 14.3.2025]

o.A.: Pro und Kontras für Autonomes Fahren; https://www.swarco.com/de/mobilitaet-der-zukunft/autonomes-fahren/autonomes-fahren-vor-nachteile [Stand: 25.3.25]

5. Anhang

Codeelemente

Hier haben wir Code-Teile aus unserem Code eingefügt und erklärt, was diese machen.

1. Deklarierung

Code

```
// Dies ist ein Kommentar
/*
Das hier auch*/
// Pins für die Motoren; Wird gebraucht für die Codebeispiele 2 bis 4
const int M1_v = 2;
const int M1_r = 3;
const int M2_v = 4;
const int M2_r = 5;

// Pins für den Ultraschallsensor; Wird gebraucht für Codebeispiele 4
const int trig = 6;
const int echo = 7;
// Variablen für Distanz und Zeit für den Ultraschallsensor
long time;
long distance;

//Pins für den Linienfolger; Wird gebraucht für die Codebeispiele
const int LF_r = A0;
const int LF_m = A1;
const int LF_l = A2;

void setup(){
  pinMode(M1_v, OUTPUT);
  pinMode(M1_r, OUTPUT);
  pinMode(M2_v, OUTPUT);
  pinMode(M2_r, OUTPUT);
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
  anhalten();
}
void anhalten();
  digitalWrite(M1_v, LOW);
  digitalWrite(M1_r, LOW);
  digitalWrite(M2_v, LOW);
  digitalWrite(M2_r, LOW);
}
```

Jeder Code braucht eine Deklarierung, um dem Arduino zu sagen, wo die Sensoren in den Arduino gesteckt sind. Um das zu speichern, nutzen wir Konstanten (`const int`) und im `void setup(){}` nutzen wir `pinMode()`. Mit `pinMode()` sagen wir dem Arduino zwei Sachen.

Erstens, dass dieser Pin benutzt wird und zweitens, ob es ein Input oder Output ist. Das benötigen wir nur bei digitalen Pins. Digitale Pins können digitale Signale (`HIGH` oder `LOW`) senden und empfangen. Analoge Pins können analoge Signale (`0` bis `1024`) in Stromstärken senden. Konstanten sind Variablen, bei denen man nach der Deklarierung nicht mehr den Wert umschreiben kann. Man braucht außerdem Variablen, um Werte zu speichern und zu ändern (`int` oder `long` (`long` ist eine Variable, die größere Werte als `int` speichern kann)). Werte sind Zahlen, `void setup` ist ein Teil vom Code, der einmal vor `void loop` ausgeführt wird. Zur Sicherheit stoppt das Auto die Fahrt, bevor es losgefahren ist.

2. Losfahren, geradeausfahren(3s), anhalten

Codebeispiel

```
void loop{
  geradeaus(3000); //3s geradeaus fahren
  beenden();       //Anhalten
}
//Funktion zum Vorwärtsfahren
void geradeaus(int time) {
  digitalWrite(M1_v, HIGH);
  digitalWrite(M1_r, LOW);
  digitalWrite(M2_v, HIGH);
  digitalWrite(M2_r, LOW);
  delay(time);
}
//Funktion zum Anhalten
void beenden() {
  digitalWrite(M1_v, LOW);
  digitalWrite(M1_r, LOW);
  digitalWrite(M2_v, LOW);
  digitalWrite(M2_r, LOW);
  while(true){}
}
```

Der Code besteht aus zwei Teilen. Geradeausfahren(losfahren) und beenden (anhalten). Damit das Auto geradeaus fährt, muss an die Pluspole der Motoren (`M1_v`, `M2_v`) ein Signal (`HIGH`) gesendet werden und an die Minuspole (`M1_r`, `M2_r`) kein Signal (`LOW`). Um den Code zu beenden, müssen zuerst die Motoren ausgeschaltet werden, indem auf `M1_v`, `M2_v`, `M1_r` und `M2_r` kein Signal (`LOW`) gesendet wird und dann eine Schleife (`while`) gestartet wird, die so lange läuft, solange der Wert in den Klammern `true` ist, aber da sich der Wert in den Klammern nie ändert, läuft die Schleife unendlich lange. Dies ist notwendig, weil man den Arduino Uno R3 nicht ausschalten kann.

3. Losfahren, geradeausfahren(3s), 90° nach links lenken, geradeausfahren(3s), 180° nach rechts lenken, geradeausfahren(3s), anhalten

Codebeispiel

```
void loop{
  geradeaus(3000); //3s geradeaus fahren
  lenken(1,90);    //90° nach links lenken
  geradeaus(3000); //3s geradeaus fahren
  lenken(0,180);   //180° nach rechts lenken
  geradeaus(3000); //3s geradeaus fahren
  beenden();       //Anhalten
}
//Funktion zum Vorwärtsfahren
void geradeaus(int time) {
  digitalWrite(M1_v, HIGH);
  digitalWrite(M1_r, LOW);
  digitalWrite(M2_v, HIGH);
  digitalWrite(M2_r, LOW);
  delay(time);
}
//Funktion zum Anhalten
void beenden() {
  digitalWrite(M1_v, LOW);
  digitalWrite(M1_r, LOW);
  digitalWrite(M2_v, LOW);
  digitalWrite(M2_r, LOW);
  while(true){}
}
//Funktion zum Lenken
void lenken(int lr/*l=1, r=0*/, int grad) {
  digitalWrite(M1_v, 1 - lr);
  digitalWrite(M1_r, lr);
  digitalWrite(M2_v, lr);
  digitalWrite(M2_r, 1 - lr);
  delay(grad*13.1111111111111111);
}
```

Hier ist jetzt das Lenken neu. Um nach rechts zu lenken, muss der rechte Motor rückwärts (`M2_v==LOW` und `M2_r==HIGH`) und der linke vorwärts (`M1_v==HIGH` und `M1_r==LOW`) drehen.

4. Losfahren, geradeausfahren; wenn Ultraschallsensor erkennt Hindernis→anhalten

```
void loop{
  trigger();
  geradeaus(0);
  if(distance<8){
    beenden();
  }
}
void geradeaus(int time){
  digitalWrite(M1_v, HIGH);
```

```
  digitalWrite(M1_r, LOW);
  digitalWrite(M2_v, HIGH);
  digitalWrite(M2_r, LOW);
  delay(time);
}
void beenden() {
  digitalWrite(M1_v, LOW);
  digitalWrite(M1_r, LOW);
  digitalWrite(M2_v, LOW);
  digitalWrite(M2_r, LOW);
  while(true){}
}
void trigger(){
  // Sender kurz ausschalten um Störungen des Signals zu vermeiden
  digitalWrite(trig, LOW);
  delay(10);
  // Signal für 10 Mikrosekunden senden
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  // Sender ausschalten
  digitalWrite(trig, LOW);
  // pulseIn → Zeit messen, bis das Signal zurückkommt
  time = pulseIn(echo, HIGH);
  /*
  Entfernung in cm berechnen
  Zeit/2 → nur eine Strecke soll berechnet werden
  Umrechnung in cm
  */
  distance = (time / 2) * 0.03432;
}
```

Der Abstandssensor schickt ein Signal (`trig`) und zählt die Zeit, bis das Signal zurückkommt (`echo`). Dann errechnet der Arduino aus der Zeit die Distanz, wie weit der Gegenstand entfernt ist.

5. Losfahren, Linie folgen

```
void loop() {
  lineFollow();
}
void geradeaus(int time){
  digitalWrite(M1_v, HIGH);
  digitalWrite(M1_r, LOW);
  digitalWrite(M2_v, HIGH);
  digitalWrite(M2_r, LOW);
  delay(time);
}
//Funktion zum Lenken
```

```
void lenken(int lr/*l=1, r=0*/, int grad) {
  digitalWrite(M1_v, 1 - lr);
  digitalWrite(M1_r, lr);
  digitalWrite(M2_v, lr);
  digitalWrite(M2_r, 1 - lr);
  delay(grad*13.1111111111111111);
}
void lineFollow() {
  if
  (analogRead(LF_r)>980 &&
  analogRead(LF_m)<980 &&
  analogRead(LF_l)<980){
    while(analogRead(LF_m)<980){
      lenken(0, 0);
    }
  }
  else if
  (analogRead(LF_r)<980 &&
  analogRead(LF_m)>980 &&
  analogRead(LF_l)<980){
    while(analogRead(LF_m)>980){
      geradeaus(0);
    }
  }
  else if
  (analogRead(LF_r)<980 &&
  analogRead(LF_m)<980 &&
  analogRead(LF_l)>980){
    while(analogRead(LF_m)<980){
      lenken(1, 0);
    }
  }
  else if
  (analogRead(LF_r)<980 &&
  analogRead(LF_m)>980 &&
  analogRead(LF_l)>980){
    while(analogRead(LF_l)>980){
      lenken(1, 0);
    }
  }
  else if
  (analogRead(LF_r)>980 &&
  analogRead(LF_m)>980 &&
  analogRead(LF_l)<980){
    while(analogRead(LF_r)>980){
      lenken(0, 0);
    }
```

```
  }
  else{geradeaus(0);}
}
```

Hier ist jetzt der Liniensensor neu. Wenn der rechte Liniensensor eine Linie erkennt, lenkt das Auto nach rechts, wenn der linke die Linie erkennt, lenkt dieser nach links. Sonst fährt das Auto geradeaus.

Unser Hauptcode

```
// Pins für den L293D
const int M1_v = 2; // IN1 (vorwärts)
const int M1_r = 3; // IN2 (rückwärts)
const int M2_v = 4; // IN3 (vorwärts)
const int M2_r = 5; // IN4 (rückwärts)

// Pins für den HC-SR04
const int trig = 6;
const int echo = 7;
//Variablen für die Distanz und Zeit
long time;
long distance;

float zeitBisAus;

void setup() {
  pinMode(M1_v, OUTPUT);
  pinMode(M1_r, OUTPUT);
  pinMode(M2_v, OUTPUT);
  pinMode(M2_r, OUTPUT);
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);

  anhalten();
}

void loop() {
  trigger();
  lineFollow();
  if (distance < 8){
    umfahren();
  }
}

// Funktion zum Vorwärtsfahren
void geradeaus(int time) {
  digitalWrite(M1_v, HIGH);
```

```
  digitalWrite(M1_r, LOW);
  digitalWrite(M2_v, HIGH);
  digitalWrite(M2_r, LOW);
  delay(time);
}

// Funktion zum Rückwärtsfahren
void zur_ck(int time) {
  digitalWrite(M1_v, LOW);
  digitalWrite(M1_r, HIGH);
  digitalWrite(M2_v, LOW);
  digitalWrite(M2_r, HIGH);
  delay(time);
}

// Funktion zum Vorwärtsfahren
void lenken(int lr/*l=1, r=0*/,int grad) {
  digitalWrite(M1_v, 1 - lr);
  digitalWrite(M1_r, lr);
  digitalWrite(M2_v, lr);
  digitalWrite(M2_r, 1 - lr);
  delay(grad*13.111111111111111);
}
// Funktion zum Stoppen
void anhalten() {
  digitalWrite(M1_v, LOW);
  digitalWrite(M1_r, LOW);
  digitalWrite(M2_v, LOW);
  digitalWrite(M2_r, LOW);
}
//Funktion zum Ultraschallsensor auslesen
void trigger(){
  // Sender kurz ausschalten um Störungen des Signals zu vermeiden
  digitalWrite(trig, LOW);
  delay(10);
  // Signal für 10 Mikrosekunden senden
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  // Sender ausschalten
  digitalWrite(trig, LOW);
  // pulseIn → Zeit messen, bis das Signal zurückkommt
  time = pulseIn(echo, HIGH);
  /*
  Entfernung in cm berechnen
  Zeit/2 -> nur eine Strecke soll berechnet werden
  Umrechnung in cm
  */
```

```
  distance = (time / 2) * 0.03432;
}
//Funktion zum umfahren eines Hindernisses
void umfahren() {
  lenken(0, 84);  // rechts
  geradeaus(2000);// geradeaus
  lenken(1, 84);  // links
  geradeaus(2000);// geradeaus
  lenken(1, 84);  // links
  geradeaus(2000);// geradeaus
  lenken(0, 84);  // rechts
}
//Funktion zum beenden des Codes
void ende() {
  anhalten(); // Fahrt stoppen, dass es nicht ewig in eine Richtung läuft
  while(true){} // eine unendliche Schleife, damit der Rest des Codes
nicht mehr drankommt. (Den Arduino Uno R3 kann man nicht ausschalten.)
}
void lineFollow() {
  if // wenn rechter Liniensensor eine Linie bemerkt → lenke nach rechts
bis Mitte erkennt die Linie
  (analogRead(A0)>980 &&
  analogRead(A1)<980 &&
  analogRead(A2)<980){
    lenken(0, 0);
  }
  else if // wenn mittlerer Liniensensor bemerkt Linie → fahre geradeaus
bis Mitte keine Linie erkennt
  (analogRead(A0)<980 &&
  analogRead(A1)>980 &&
  analogRead(A2)<980){
      geradeaus(0);

  }
  else if // wenn linker Liniensensor bemerkt Linie → lenke nach links bis
Mitte erkennt die Linie
  (analogRead(A0)<980 &&
  analogRead(A1)<980 &&
  analogRead(A2)>980) {
      lenken(1, 0);

  }
  else if
  (analogRead(A0)<980 &&
  analogRead(A1)>980 &&
  analogRead(A2)>980){
      lenken(1, 0);
```

```
  }
  else if
  (analogRead(A0)>980 &&
  analogRead(A1)>980 &&
  analogRead(A2)<980){
      lenken(0, 0);

  }

  else{geradeaus(0);}
}
```

6. Originalitätserklärung

Wir erklären, dass wir die Forscherarbeit ohne fremde Hilfe angefertigt und nur die im Literaturverzeichnis angeführten Quellen und Hilfsmittel benutzt haben.

Paul Maier

Jakob Hollborn