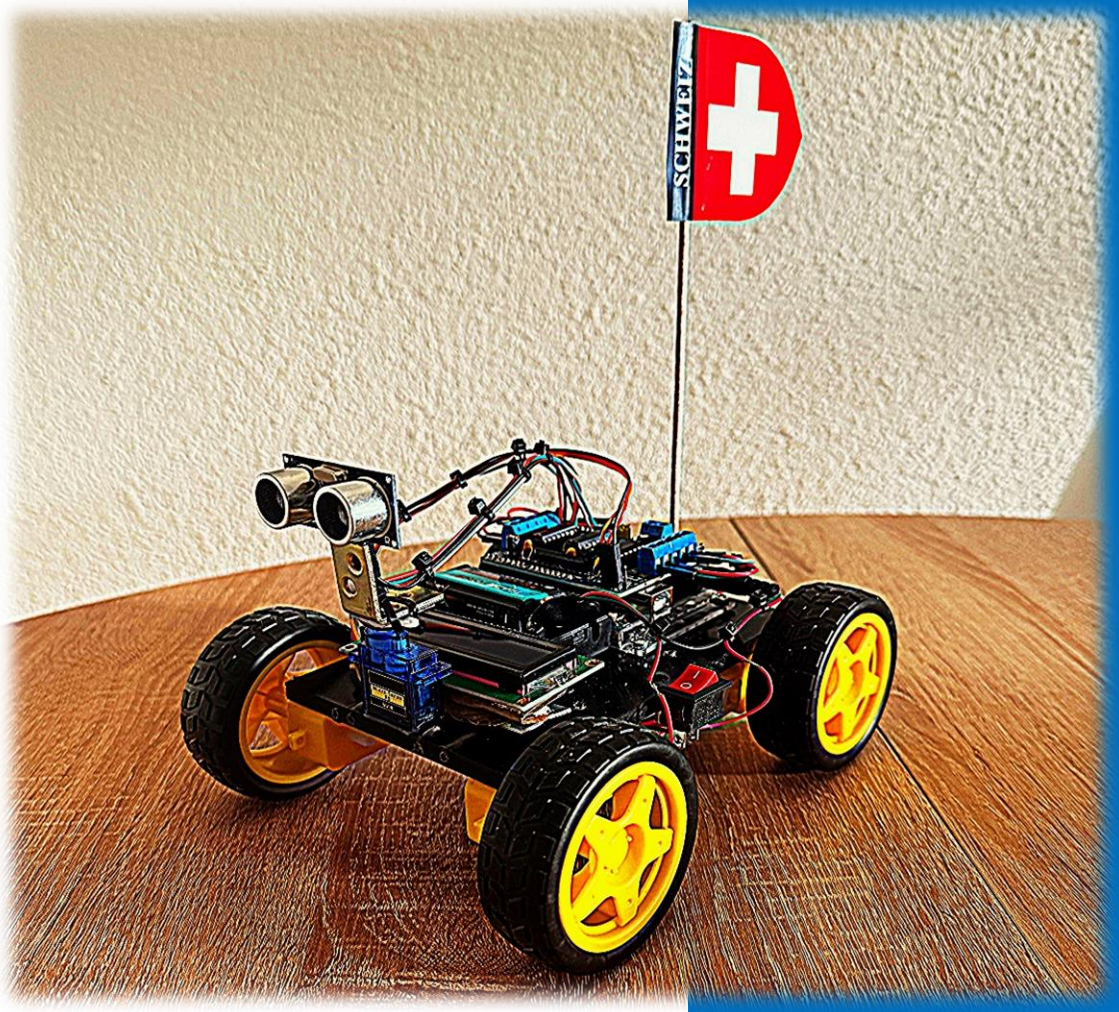
Arduino Obstacle detection Car



Ismael Bachmann-Morales

Karim Khachia

15.12.2024

1 Inhaltsverzeichnis

2 Einleitung

Die vorliegende Arbeit beschreibt die Entwicklung eines autonomen Hindernisvermeidungsfahrzeugs basierend auf der Arduino-Plattform. Ziel des Projekts war es, ein energieeffizientes und zuverlässiges Fahrzeug zu entwerfen, das Hindernisse erkennt, selbstständig ausweicht und seinen Status auf einem LCD-Display anzeigt.

Diese Dokumentation behandelt alle Aspekte des Projekts, von der Planung über die Inbetriebnahme bis hin zur Reflexion.

---

3 Projektbeschreibung

Das entwickelte Fahrzeug basiert auf einem Arduino UNO Mikrocontroller, einem Ultraschallsensor für die Hinderniserkennung und einem L293D-Motor-Shield zur Motorsteuerung. Es ist in der Lage, autonom zu navigieren, Hindernisse zu erkennen und auszuweichen, sowie den Status "Fährt", "Stopp", "Links" und "Rechts" auf einem LCD-Display anzuzeigen.

Die Implementierung orientiert sich an den Prinzipien von Clean Code und modularer Softwareentwicklung, was die Erweiterbarkeit und Wartbarkeit des Systems gewährleistet. Zusätzliche Tests und Optimierungen garantieren die Zuverlässigkeit und Energieeffizienz des Fahrzeugs.

4 Ziele des Projekts

1. Hinderniserkennung: Präzises Erkennen von Hindernissen in einem Radius von 30 cm mithilfe eines Ultraschallsensors.
2. Autonome Navigation: Stabiles Fahren auf unterschiedlichen Oberflächen und präzises Ausweichen.
3. Statusanzeige: Darstellung der Betriebszustände auf einem LCD-Display für eine verbesserte Benutzertransparenz.
4. Clean Code-Prinzipien: Modularer, gut strukturierter und dokumentierter Code.
5. Erweiterbarkeit: Möglichkeit der Integration zusätzlicher Sensoren und Funktionen.

5 Organisation

5.1 Terminplan

Bereich	Aufgabe	Verantwortliche Person(en)	Deadline	Status	Bemerkungen	Zeit
Organisation	Terminplan	Ismael & Karim	17.11.2024	Abgeschlossen	Erstellung des Terminplans	1h
	Pflichtenheft-Aufgabenerstellung	Ismael & Karim	17.11.2024	Abgeschlossen	Abgleich mit Projektanforderungen	1h
	Team-Organisation	Ismael & Karim	17.11.2024	Abgeschlossen	-	1h
SW- & HW-Engineering	Elektronik-Schema	Ismael & Karim	23.11.2024	Abgeschlossen	Verbindung Motorsteuerungen, Arduino UNO	2h
	1. Bestellung der Bauteile	Ismael & Karim	23.11.2024	Abgeschlossen	-	30min
	Aufbau des Fahrzeugs	Ismael & Karim	30.11.2024	Abgeschlossen	Komplettaufbau von Grund auf	5h
	Flussdiagramm sämtlicher SW-Module	Ismael & Karim	30.11.2024	Abgeschlossen	Erstellung des Flussdiagramms	3h
	SW-Regeln	Ismael & Karim	30.11.2024	Abgeschlossen	Clean-Code-Richtlinien umsetzen	2h
	2. Bestellung der Bauteile	Ismael & Karim	30.11.2024	Abgeschlossen	Motor-Shield L293D	30min
	Umbau des Fahrzeugs	Ismael & Karim	07.12.2024	Abgeschlossen	vollständiger Abbau und Neuaufbau des Fahrzeugs	5h
	Umsetzung "Clean Code"	Ismael & Karim	01.12.2024	Abgeschlossen	Optimierung während der Testphase	5h
	3rd-Party-Code-Fragmente	Ismael & Karim	01.12.2024	Abgeschlossen	Arduino-Libraries dokumentieren	3h
	SW-Modularisierung	Ismael & Karim	01.12.2024	Abgeschlossen	Module klar definieren	2h
Inbetriebnahme / Features	Inbetriebnahme & Test-Protokoll	Ismael & Karim	05.12.2024	Abgeschlossen	Tests fortlaufend dokumentieren	4h

	Soll-/Ist-Abgleich mit Pflichtenheft	**Ismael & Karim**	**14.12.2024**	**Abgeschlossen**	**Kontrolle**	**1h**
Dokumentation	**Dokumentation abschließen**	**Ismael und Karim**	**15.12.2024**	**Abgeschlossen**	**Letzte Überarbeitung und Kontrolle**	**4h**

Gesamte Zeit: 40 Stunden

5.2 Phasenplanung

Phase 1: Vorbereitung (Woche 1)

- Ziel: Sicherstellen, dass alle Voraussetzungen für den Projektstart erfüllt sind.
    - Recherche zu den erforderlichen Komponenten.
    - Erstellung eines detaillierten Pflichtenhefts.
    - Bestellung und Beschaffung der Hardwarekomponenten.
    - Festlegen eines Kommunikationsplans für das Team.

Phase 2: Hardwareaufbau (Woche 2)

- **Ziel: Die physische Basis des Roboters fertigstellen.**
    - Montage des Fahrzeugchassis.
    - Installation und Verkabelung der Sensoren, Motoren und des LCD-Bildschirms.
    - Überprüfung der Hardware auf Fehler oder Defekte.

Phase 3: Softwareentwicklung (Woche 3)

- Ziel: Den Robotercode erstellen und schrittweise testen.
    - Implementierung der Grundfunktionen (Motorsteuerung und Hinderniserkennung).
    - Integration der LCD-Funktionalität zur Statusanzeige.
    - Aufbau einer stabilen Kommunikationsschnittstelle für Debugging.

Phase 4: Integration und Tests (Woche 4)

- Ziel: Sicherstellen, dass alle Komponenten nahtlos zusammenarbeiten.
    - Durchführung von Einzeltests für Sensoren, Motoren und LCD.
    - Integration der Softwaremodule und erster Systemtest.
    - Identifikation und Behebung von Fehlern in der Hardware oder Software.

Phase 5: Optimierung und Dokumentation (Woche 5)

- Ziel: Abschlussarbeiten durchführen und das Projekt abschließen.
    - Optimierung der Bewegungslogik sowie des Entscheidungsprozesses.
    - Finalisierung der Projektunterlagen, inklusive Testprotokolle.
    - Erstellung einer Präsentation oder Demonstration für die Ergebnisse.
    - Reflexion und Diskussion von Erweiterungsmöglichkeiten.

6 Pflichtenheft

Kategorie	Beschreibung
Ziel des Projekts	Entwicklung eines autonomen Fahrzeugs, das Hindernisse erkennt, eigenständig ausweicht und den Status auf einem LCD anzeigt.
Funktionalität	- Das Fahrzeug soll Hindernisse mithilfe eines Ultraschallsensors erkennen.
	- Bei der Erkennung eines Hindernisses soll das Fahrzeug die Richtung automatisch ändern.
Anzeigefunktion	- Statusanzeigen wie "Fährt", "Stopp", "Links", "Rechts" werden auf einem LCD-Display dargestellt, um Transparenz für den Benutzer zu schaffen.
Softwarequalität	- Der Code soll klar strukturiert und kommentiert sein, sodass er leicht erweiterbar und wartbar ist (Clean-Code-Prinzipien).
Leistungsziele	- Das Fahrzeug soll in der Lage sein, schnell auf Hindernisse zu reagieren und stabil auf unebenen Oberflächen zu fahren.
Energieeffizienz	- Die Stromversorgung soll für mindestens 2 Stunden Betriebszeit ausgelegt sein, wobei die Leistung über eine Powerbank sichergestellt wird.
Erweiterbarkeit	- Das System soll modular aufgebaut sein, sodass Sensoren oder neue Funktionen leicht hinzugefügt werden können.

6.1 Detaillierte Anforderungen

1. Hinderniserkennung: Das Fahrzeug muss Hindernisse in einem Radius von 30 cm erkennen und präzise reagieren.
2. Autonome Navigation: Automatische Richtungsänderung basierend auf den Messdaten.
3. Statusanzeige: Der aktuelle Betriebsmodus wird klar und verständlich auf einem LCD angezeigt.
4. Reaktionszeit: Sensordaten sollen innerhalb von 100 ms verarbeitet werden, um schnelle Anpassungen zu ermöglichen.
5. Sicherheitsmaßnahmen: Implementierung eines Not-Stopp-System und sicherer Spannungswerte, um Fehlfunktionen zu vermeiden.

7 Soll / Ist Abgleich

7.1 Zielsetzung

SOLL	IST	Abweichung
Das Fahrzeug soll Hindernisse erkennen und autonom ausweichen.	Hindernisse werden durch den Ultraschallsensor erkannt. Das Fahrzeug stoppt und weicht aus.	Keine
Hindernisse sollen in Echtzeit erkannt werden (Entfernung ≤ 30 cm).	Echtzeit-Hinderniserkennung durch den HC-SR04-Sensor, reagiert zuverlässig bei ≤ 30 cm.	Keine
Der aktuelle Status soll auf einem LCD angezeigt werden.	Das LCD zeigt dynamisch die Statusmeldungen "Start", "Fährt", "Sucht", "Links", "Rechts" an.	Keine
Fahrzeug fährt selbstständig vorwärts und führt Ausweichmanöver durch.	Fahrzeug bewegt sich vorwärts, stoppt bei Hindernissen, scannt links/rechts und weicht aus.	Keine
Steuerung über Arduino Uno und AFMotor-Shield.	Arduino Uno und AFMotor-Shield wurden erfolgreich integriert.	Keine

7.2 Anforderungen

Hardware

SOLL	IST	Abweichung
4 Gleichstrommotoren: Bewegung des Fahrzeugs.	4 Gleichstrommotoren wurden angeschlossen und korrekt gesteuert.	Keine
HC-SR04 Ultraschallsensor: Hinderniserkennung.	Sensor arbeitet zuverlässig und misst Entfernungen präzise.	Keine
Servo-Motor: Drehung des Ultraschallsensors.	Der Servo dreht den Sensor präzise nach links und rechts zur Umgebungserfassung.	Keine
LCD (I2C): Statusanzeige.	LCD zeigt Statusmeldungen entsprechend der Fahrzeugaktionen an.	Keine

7.3 Software

SOLL	IST	Abweichung
Statusmeldungen: "Fährt", "Stopp", "Sucht", "Links", "Rechts".	Statusmeldungen werden dynamisch und korrekt angezeigt.	Keine
Saubere Code-Struktur: Modularisierung, Kommentierung.	Code ist modular aufgebaut und gut kommentiert.	Keine
Dynamische Hindernisvermeidung: Richtung mit mehr Platz wird gewählt.	Fahrzeug weicht der Richtung mit mehr Platz zuverlässig aus.	Keine

7.4 Performance

SOLL	IST	Abweichung
Reaktionszeit: ≤ 1 Sekunde.	Hindernisvermeidung erfolgt innerhalb von 0,5–1 Sekunde.	Keine
Hinderniserkennung: ≤ 30 cm.	Hindernisse werden korrekt bei ≤ 30 cm erkannt.	Keine

8 Überarbeiteter Projektbericht Troubleshooting

8.1 Funktionalität

Das ursprüngliche Ziel bestand darin, ein Fahrzeug zu entwickeln, das einer schwarzen Linie folgt. Aufgrund technischer Probleme wurde das Konzept angepasst: Jetzt erkennt das Fahrzeug Hindernisse und weicht diesen aus.

8.2 Hardware

- Ursprüngliche Komponenten: Zwei IR-Sensoren, L298N-Motor-Shield, Arduino UNO, vier Gleichstrommotoren.
- Anpassung: Ein L293D-Motor-Shield wurde integriert, um die Steuerung der Motoren zu verbessern, und ein Ultraschallsensor wurde hinzugefügt.
- Sowie ein LCD-Display für die Anzeige der Richtung und Bewegungen.
- Die Powerbank wurde durch eine zuverlässige 9-Volt-Batterie ersetzt, die eine höhere Spannung als die zuvor genutzte 7,2-Volt-Batterie liefert. Diese Änderung war notwendig, da wir festgestellt haben, dass das Fahrzeug mit 7,2 Volt zu wenig Leistung hatte. Dies führte zu Störungen im Fahrverhalten und beeinträchtigte die Sensorerkennung erheblich. Mit der 9-Volt-Batterie wurde die Leistungsfähigkeit verbessert und ein stabileres Betriebsverhalten gewährleistet.

8.3 Software

- Ursprüngliche Steuerung basierte auf IR-Sensoren, die durch die Ultraschallsensoren ersetzt wurden.
- Die Software wurde überarbeitet, um die Hinderniserkennung und Bewegungslogik zu optimieren.

8.4 Probleme mit L298N-Motor-Shield

- Das L298N-Motor-Shield war defekt und konnte nicht alle Motoren separat korrekt ansteuern. Zusätzlich waren die Ausgänge 3 und 4 defekt.
- Lösung: Austausch durch das L293D-Motor-Shield, dass eine präzisere Steuerung ermöglicht und alle Motoren einzeln ansteuert.

8.5 Performance

- Das Fahrzeug weicht Hindernissen präzise aus.
- Durch das LCD-Display werden die Kommandos angezeigt.

9 Komponenten

9.1 Hardware

1. **Arduino Uno**: Hauptcontroller.
2. **Motor Shield (AFMotor oder kompatibel)**: Steuerung der Motoren.
3. **Ultraschallsensor HC-SR04**: Hinderniserkennung.
4. **Vier Gleichstrommotoren**: Antrieb des Fahrzeugs.
5. **Servo-Motor**: Steuerung des Ultraschallsensors für seitliche Scans.
6. **LCD-Display (I2C, 16x2)**: Statusanzeige.
7. **9V Batterie**: Stromversorgung.
8. **Fahrzeugchassis**: Träger der Komponenten.

9.2 Software / Modularisierung

Hauptfunktionen:

1. **moveForward()**: Aktiviert alle Motoren für Vorwärtsbewegung.
2. **moveStop()**: Stoppt alle Motoren.
3. **turnRight()**: Dreht das Fahrzeug nach rechts.
4. **turnLeft()**: Dreht das Fahrzeug nach links.
5. **scanForward(), scanRight(), scanLeft()**: Bewegt den Servo-Motor zur Umgebungserkennung.

Clean-Code-Prinzipien:

- Funktionen sind kurz und gut kommentiert.
- Verwendung von Konstanten anstelle von Magic Numbers.
- Modularisierung der Software in klar definierte Module.

Software (3th Party-Code)-Bibliotheken:

- **AFMotor-Library**: Steuerung der Motoren.
- **NewPing-Library**: Steuerung des Ultraschallsensors.
- **Servo-Library:** Steuerung des Ultraschallsensors für seitliche Scans.
- **LiquidCrystal_I2C**: Steuerung des LCD-Displays.

9.3 Inbetriebnahme & Testprotokoll

Test-ID	Testbeschreibung	Erwartetes Ergebnis	IST-Ergebnis	Status	Bemerkungen
T-001	LCD-Anzeige: Status "Fährt, Links, Recht, Stopp"	Der Status wird korrekt angezeigt	OK	Erfüllt	Keine
T-002	Vorwärtsbewegung	Fahrzeug fährt geradeaus	OK	Erfüllt	Keine
T-003	Hinderniserkennung	Fahrzeug stoppt bei einem Hindernis ≤ 30 cm Abstand	Stopp und Scan aktiviert	Erfüllt	Empfindlichkeit angepasst
T-004	Richtungsänderung: Links	Fahrzeug dreht nach links, wenn nötig	Korrekt links gedreht	Erfüllt	Keine
T-005	Richtungsänderung: Rechts	Fahrzeug dreht nach rechts, wenn nötig	Korrekt rechts gedreht	Erfüllt	Keine
T-006	Fährt dorthin, wo mehr Platz erkannt wird	Fährt in die Richtige Richtung und reagiert.	OK	Erfüllt	Keine

10 Visuelle Darstellung der Komponenten

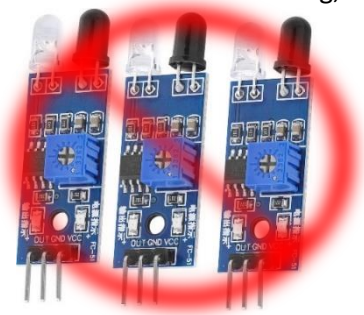
10.1 Symbolbedeutung

Nicht in der finalen Version implementierte Bauteile.

10.2 Übersicht der Bauteile

- Arduino Uno Das Herzstück des Systems, zuständig für die Steuerung und Verarbeitung der Sensordaten.

- IR- Sensor zur Linienerkennung, erkennt reflektiertes Infrarotlicht.



KCD1-101 Kippschalter On/Off Schalter.

- K LCD1602 LCD-Display Blau mit I2C Modul.

- L298N Modul zur Steuerung der beiden Gleichstrommotoren.

- Räder & Dc Motoren Vier Motoren, die das Fahrzeug antreiben mit Rädern.

- Batteriehalter 2xAA mit Anschlusskabel und Schalter.

- GLK-Technologies® G2PB18650 Powerbank mit x2 3200 mAh 3,7V High Power Akku.

- L293D Motor Control Shield.

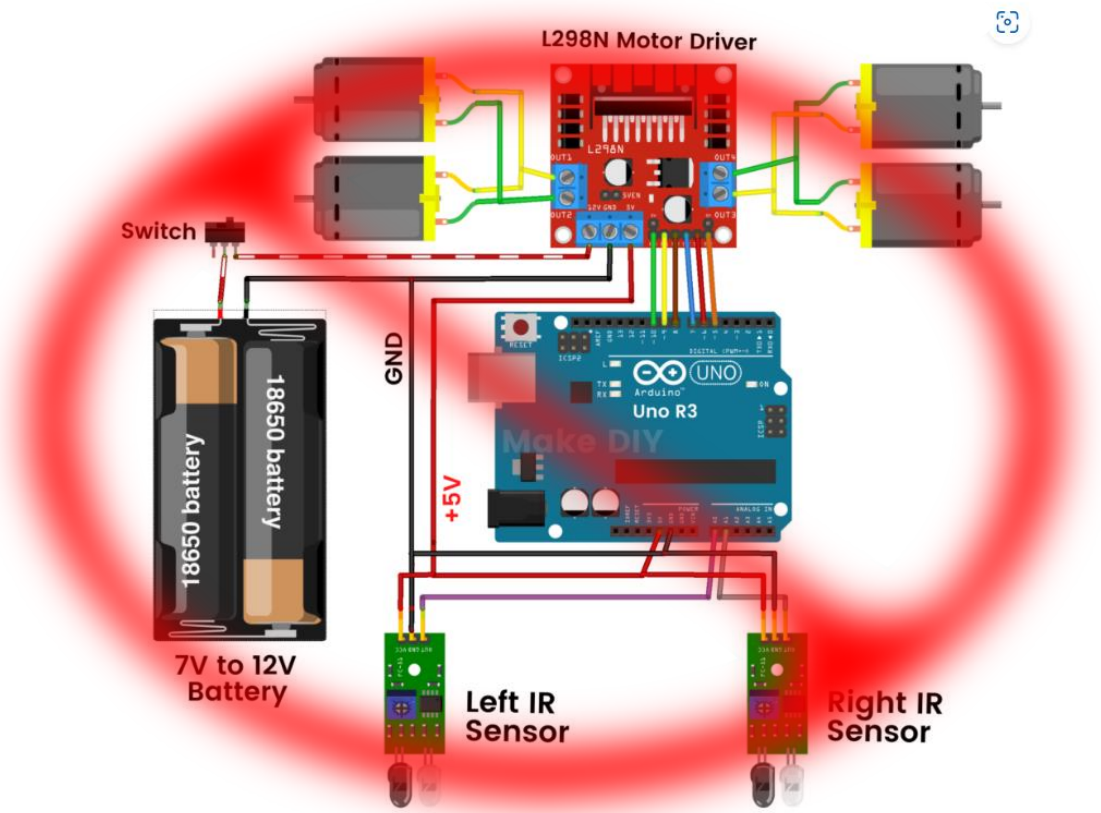
- Tower Pro Micro Servo 9G / SG90.

- B&T Metall Hart PVC schwarz Platten 6,0 mm stark im Zuschnitt Größe 100 x 150 mm.

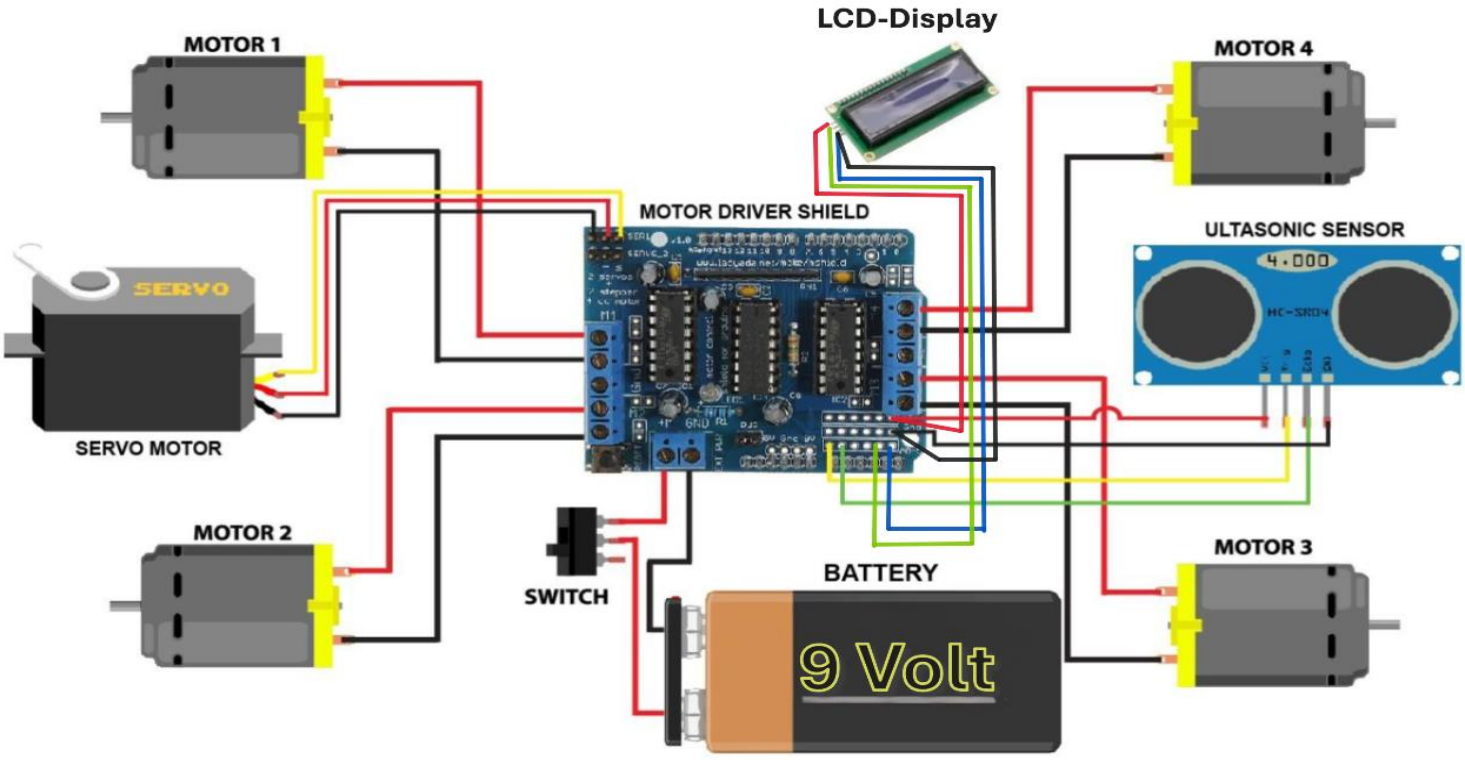
- 9 Volt Batterie Stromversorgung.

10.3 Elektronik-Schema

10.3.1 Follow Line Car

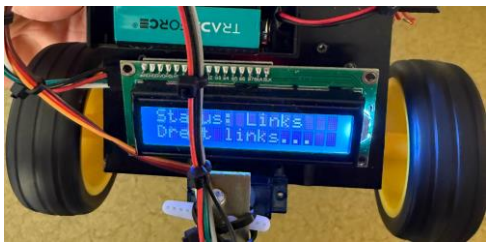


10.3.2 Obstacle detection Car



10.3.3Verbindungen:

- **Motoren:**
    - Motor 1 (rechts vorne): Motoranschluss 1 auf dem Shield.
    - Motor 2 (rechts hinten): Motoranschluss 2 auf dem Shield.
    - Motor 3 (links vorne): Motoranschluss 3 auf dem Shield.
    - Motor 4 (links hinten): Motoranschluss 4 auf dem Shield.
- **Ultraschallsensor HC-SR04:**
    - **Trig-Pin**: A0
    - **Echo-Pin**: A1
- **Servo-Motor:**
    - PWM-Signal: Pin 10.
- **LCD (I2C):**
    - SCL: Verbunden mit A5.
    - SDA: Verbunden mit A4

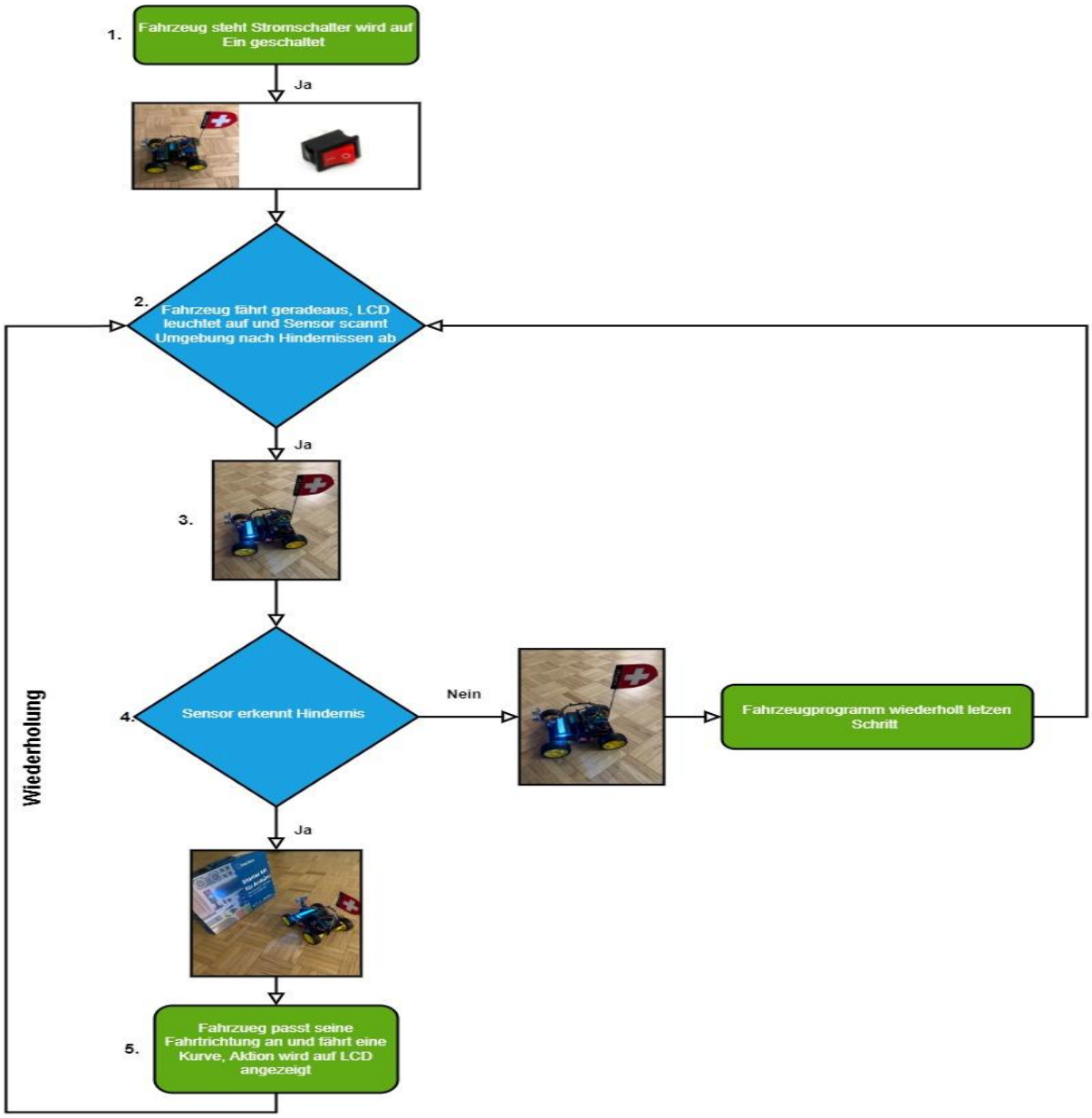


10.4 Flussdiagramm

Flussdiagramm der Software

Hauptprozesse:

1. Start (Initialisierung der Komponenten).
2. Bewegung: Vorwärtsfahren.
3. Hinderniserkennung: Ultraschallsensor misst Abstand.
4. Entscheidung: Hindernis erkannt (≤ 30 cm)?
    - Ja: Fahrzeug stoppt, Umgebungsscan.
    - Nein: Fahrzeug fährt weiter.
5. Richtungsänderung: Drehung nach links oder rechts basierend auf Sensorwerten.
6. Wiederholung.



10.5 Clean Code

Clean Code:

- Funktionen sind kurz und gut kommentiert.
- Nutzung von Konstanten statt Magic Numbers.
- Modularisierung der Software in klar definierte Module.

```
#include <AFMotor.h>
#include <Servo.h>
#include <NewPing.h>
#include <LiquidCrystal_I2C.h>

// Pins für den Ultraschallsensor
#define Trig_Pin A0
#define Echo_Pin A1
#define Max_Dist 250
#define STOP_DISTANCE 30 // Abstand in cm, bei dem das Fahrzeug stoppt

// Maximalgeschwindigkeit der Motoren
const int Max_Speed = 200;

// Motoren und Servo initialisieren
AF_DCMotor motor1(1, MOTOR12_1KHZ);
AF_DCMotor motor2(2, MOTOR12_1KHZ);
AF_DCMotor motor3(3, MOTOR34_1KHZ);
AF_DCMotor motor4(4, MOTOR34_1KHZ);
Servo myservo;
NewPing ultra_sonic(Trig_Pin, Echo_Pin, Max_Dist);

// LCD initialisieren
LiquidCrystal_I2C lcd(0x27, 16, 2); // Adresse 0x27, 16 Zeichen, 2 Zeilen

void setup() {
  Serial.begin(9600);

  // LCD initialisieren
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Status: Start    ");
  lcd.setCursor(0, 1);
  lcd.print("Bereit...        ");
  delay(1000);

  // Servo initialisieren
  myservo.attach(10);
  myservo.write(90);  // Startposition (Mitte)
  delay(500);

  moveForward(); // Fahrzeug startet direkt in der Vorwärtsbewegung
}

void loop() {
  int distance = scanForward(); // Umgebung vor dem Fahrzeug scannen

  if (distance > 0 && distance <= STOP_DISTANCE) {
    moveStop(); // Stoppen bei Hindernis
    delay(500);
```

```
    lcd.setCursor(0, 0);
    lcd.print("Status: Stop     "); // Stop anzeigen
    lcd.setCursor(0, 1);
    lcd.print("Scanning...      ");

    int distRight = scanRight(); // Nach rechts scannen
    delay(500);
    int distLeft = scanLeft();   // Nach links scannen
    delay(500);

    // Drehen in Richtung mit mehr Platz
    if (distRight > distLeft) {
      turnRight();
    } else {
      turnLeft();
    }
  } else {
    moveForward(); // Weiter vorwärts fahren, wenn kein Hindernis erkannt wird
  }
}

// Funktion: Umgebung vor dem Fahrzeug scannen
int scanForward() {
  myservo.write(90);  // Servo in Mittelposition
  delay(300);
  lcd.setCursor(0, 0);
  lcd.print("Status: Faehrt   "); // Fährt anzeigen
  lcd.setCursor(0, 1);
  lcd.print("                "); // Zweite Zeile löschen
  return readDistance();
}

// Funktion: Umgebung nach rechts scannen
int scanRight() {
  lcd.setCursor(0, 0);
  lcd.print("Status: Sucht    "); // Sucht anzeigen
  lcd.setCursor(0, 1);
  lcd.print("Nach rechts...   ");
  myservo.write(10);  // Servo nach rechts drehen
  delay(500);
  int dist = readDistance();
  myservo.write(90);  // Servo zurück zur Mitte
  return dist;
}

// Funktion: Umgebung nach links scannen
int scanLeft() {
  lcd.setCursor(0, 0);
  lcd.print("Status: Sucht    "); // Sucht anzeigen
  lcd.setCursor(0, 1);
  lcd.print("Nach links...    ");
  myservo.write(170);  // Servo nach links drehen
  delay(500);
  int dist = readDistance();
  myservo.write(90);  // Servo zurück zur Mitte
  return dist;
}

// Funktion: Entfernung messen
int readDistance() {
  int cm = ultra_sonic.ping_cm();
  if (cm <= 0) {  // Wenn Sensor keine gültige Messung liefert
    cm = Max_Dist;
  }
  return cm;
```

```
}

// Funktion: Vorwärtsbewegung
void moveForward() {
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);

  motor1.setSpeed(Max_Speed);
  motor2.setSpeed(Max_Speed);
  motor3.setSpeed(Max_Speed);
  motor4.setSpeed(Max_Speed);

  lcd.setCursor(0, 0);
  lcd.print("Status: Faehrt   "); // Fährt anzeigen
  lcd.setCursor(0, 1);
  lcd.print("                "); // Zweite Zeile leeren
  Serial.println("Fahrzeug fährt vorwärts...");
}

// Funktion: Stoppen
void moveStop() {
  motor1.run(RELEASE);
  motor2.run(RELEASE);
  motor3.run(RELEASE);
  motor4.run(RELEASE);

  lcd.setCursor(0, 0);
  lcd.print("Status: Stop    "); // Stop anzeigen
  lcd.setCursor(0, 1);
  lcd.print("                "); // Zweite Zeile löschen
  Serial.println("Fahrzeug gestoppt...");
}

// Funktion: Rechtsdrehung
void turnRight() {
  lcd.setCursor(0, 0);
  lcd.print("Status: Rechts   "); // Rechtsdrehung anzeigen
  lcd.setCursor(0, 1);
  lcd.print("Dreht rechts...  ");
  Serial.println("Fahrzeug dreht nach rechts...");
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);
  delay(800);  // Längere Drehzeit, um die Richtung deutlich zu ändern
  moveForward(); // Danach wieder vorwärts fahren
}

// Funktion: Linksdrehung
void turnLeft() {
  lcd.setCursor(0, 0);
  lcd.print("Status: Links    "); // Linksdrehung anzeigen
  lcd.setCursor(0, 1);
  lcd.print("Dreht links...   ");
  Serial.println("Fahrzeug dreht nach links...");
  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
  delay(800);  // Längere Drehzeit, um die Richtung deutlich zu ändern
  moveForward(); // Danach wieder vorwärts fahren
}
```

11 Fazit

Wir blicken mit Stolz auf unser gemeinsames Projekt zurück und sind beeindruckt von den Fortschritten, die wir in kurzer Zeit erzielt haben. Die Dokumentation unseres autonomen Hindernisvermeidungsfahrzeugs zeigt die gesamte Reise von der ersten Idee bis hin zum fertigen Produkt. Unser Ziel, ein Fahrzeug zu entwickeln, das nicht nur Hindernisse erkennt und autonom ausweicht, sondern auch seinen Status auf einem LCD Display anzeigt, haben wir vollständig erreicht. Dabei konnten wir die Prinzipien von Clean Code konsequent umsetzen, wodurch unser Code strukturiert, modular und zukunftssicher gestaltet wurde.

Im Laufe des Projekts haben wir viele wertvolle Erfahrungen gesammelt, insbesondere im Umgang mit elektronischen Bauteilen, der Planung eines komplexen Systems und der Fehlersuche. Herausforderungen, wie der Defekt des L298N-Motorshields, konnten wir gemeinsam meistern, indem wir eine alternative Lösung mit dem L293D-Motorshield erfolgreich integriert haben. Auch die Überarbeitung der Software zur Optimierung der Hinderniserkennung und Bewegungslogik hat uns gezeigt, wie wichtig Tests und iterative Verbesserungen sind.

Unsere Inspiration, ein autonomes Fahrzeug zu entwickeln, das ausschließlich mit Batterien betrieben wird, stammt aus unserem Interesse an nachhaltiger Energie und moderner Robotik. Unsere Vision war es, ein energieeffizientes, zuverlässiges und zugleich erweiterbares System zu bauen, das auch in der Praxis überzeugt. Mit diesem Projekt haben wir nicht nur unsere technischen Fähigkeiten verbessert, sondern auch gelernt, als Team effektiv zusammenzuarbeiten.

Abschließend können wir sagen, dass uns die Arbeit an diesem Projekt nicht nur gefordert, sondern auch unglaublich motiviert hat. Es hat uns gezeigt, was mit Engagement, Kreativität und technischer Präzision möglich ist. Wir freuen uns darauf, dieses Wissen in zukünftigen Projekten einzusetzen und weiterzuentwickeln.

12 Erweiterungsmöglichkeiten

12.1 Kamera

Eine Kamera ermöglicht visuelle Erkennung, z. B. von Objekten oder Verkehrszeichen. Dadurch kann das Fahrzeug präziser navigieren oder Objekte priorisieren. In Kombination mit Bilderkennung wird es ideal für Sicherheits- und Überwachungsaufgaben.

12.2 Remote-Controller

Mit Infrarot oder WLAN kann das Fahrzeug per App oder Fernbedienung gesteuert werden. Dies erlaubt manuelle Eingriffe, Live-Überwachung und die flexible Steuerung zwischen automatischem und manuellem Betrieb.

13 Quellen

1. **NewPing-Library**: Offizielle Bibliothek für Ultraschallsensoren: https://bitbucket.org/teckel12/arduino-new-ping/wiki/Home
2. **AFMotor-Library**: Steuerung der Motoren: https://learn.adafruit.com/adafruit-motor-shield/library
3. **LiquidCrystal_I2C**: LCD-Bibliothek: https://github.com/johnrickman/LiquidCrystal_I2C
4. **Projektidee :** https://www.youtube.com/watch?v=bRZNyQtwZVE
5. **Beihilfe:** ChatGPT (Code analysieren)

13.1 Eingekaufte Teile:

1. **Arduino Uno Herstellerwebsite:** https://www.bastelgarage.ch/arduino-uno-rev-3-smd-board-atmega328?search=arduino%20uno
2. **IR-Sensor (TCRT5000)**
   URL: https://www.amazon.de/dp/B07D924JHT?ref=ppx_yo2ov_dt_b_fed_asin_title&th=1
3. **L298N Motor Driver Modul**
   Produktdaten: https://www.bastelgarage.ch/l298n-schrittmotorendstufe-h-brucke-dc-motor-treiber?search=l298
4. **Räder Dc Motoren**
   URL: https://www.amazon.de/Gebildet-Getriebemotor-Allradantrieb-Spielzeugauto-Flugzeugspielzeug
5. **GLK-Technologies® G2PB18650 Powerbank mit x2 3200 mAh 3,7V High Power Akku**
   URL: https://www.amazon.de/GLK-Technologies%C2%AE-G2PB18650-Powerbank-3200-Power
6. **B&T Metall Hart PVC schwarz Platten 6,0 mm stark im Zuschnitt Größe 100 x 150 mm**
   URL: https://www.amazon.de/Metall-schwarz-Platten-stark-Zuschnitt
7. **Tower Pro Micro Servo 9G / SG90**
   URL: https://www.bastelgarage.ch/tower-pro-micro-servo-9g-sg90
8. **L293D Motor Control Shield**
   URL: https://www.bastelgarage.ch/motor-control-shield-fur-arduino-v1-0-mit-l293d?search=l293d

13.2 Anhang:

1. **GitHub URL:** https://github.com/isbamo/Arduino-Car
   QR-Code: