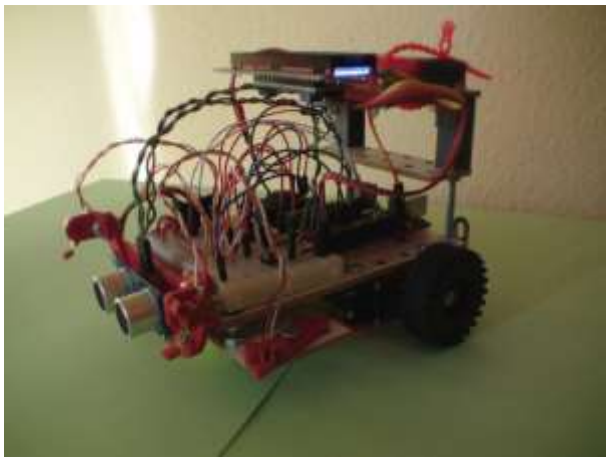
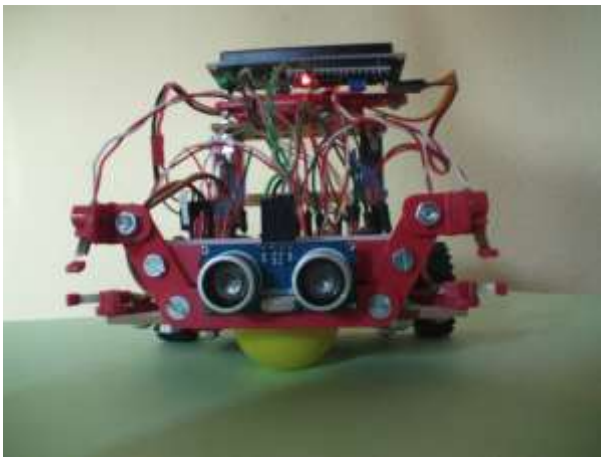
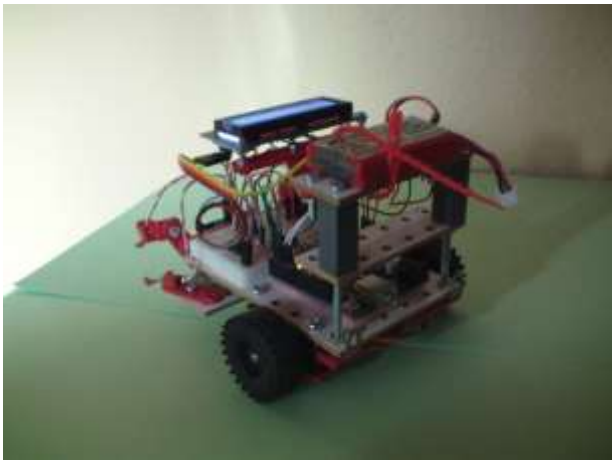
EBW Hannover

Arduino Roboter







im Eigenbau

Enno Klatt

Inhaltsverzeichnis

1.	Einführung	3
2.	Problem	3
2.1.	Was soll der Roboter machen?	3
3.	Analyse	3
3.1.	Hardware (Was ist bekannt?)	3
3.2.	Software (Was ist bekannt?)	3
3.3.	Hardware (Was ist unbekannt?)	3
4.	Entwurf	4
4.1.	Manöver, Strategien (Algorithmus)	4
5.	Test der Komponenten	5
5.1.	Taster	5
5.2.	Ultraschallmesser	5
5.3.	Servo	6
6.	Programmierung	7
7.	Zusammenfassung	9
8.	Anlage	10
8.1.	Versuche Taster	10
8.2.	Versuche Ultraschallmesser	14
8.3.	Versuche Servos	16
9.	Technische Daten	18

1. Einführung

Mein erster Versuch war der ASURO (Conrad, Voelkner, ca. 50 €). Der Spaß besteht hier im Löten, Testen und Programmieren.
In der Schule habe ich das SoccerBot (Roboterbausatz "SoccerBot" ca. 300 €) in die Finger bekommen (3 Stück).
Hieraus entstand die Idee, es mal selbst von Anfang an zu versuchen.

2. Problem

2.1. Was soll der Roboter machen?

- Nach dem Start automatisch agieren.
- Einfache Hindernisse erkennen und diesen ausweichen.
- Den Ausgang aus einem leeren rechteckigen Raum finden.
- Den Ausgang aus einem beliebigen Raum finden.

3. Analyse

3.1. Hardware (Was ist bekannt?)

- Arduino UNO
- Plattform aus LPE-Technik (http://umt-in-der-schule.de/umt-halbzeuge.html).
- Preiswerte Antriebe, z.B. umgebaute Servos (Standard-Servo IQ-400BB).
- Entfernungsmessung mit Ultraschall (HC-SR04).
- Berührungssensoren (Taster).

3.2. Software (Was ist bekannt?)

- Arduino IDE.
- Programmieren in „C++", angepasst an die IDE!
- Auswertung von Sensoren (Taster) und steuern von Aktoren (Servos).

3.3. Hardware (Was ist unbekannt?)

- Welche Typen von Servos sind geeignet?
- Überschreiten die Lastströme die zulässigen Arduino-Ströme? (max. 200 mA).
- Art und Aufbau der Spannungsversorgung für Arduino und Servos?
- Ist ein Ultraschallsensor als Entfernungsmesser geeignet. Sind die Messwerte stets brauchbar und können sie zur Steuerung dienen?
- Reagieren die Berührungssensoren (Taster) auf kleinste Kräfte?

3.4. Beziehungen (bekannt)

Für den Arduino UNO können mit Hilfe der Arduino-IDE Programme erstellt werden.
Für Taster, Ultraschallsensor und Servo stehen Software-Bibliotheken zur Verfügung. Die darin enthaltenen Funktionen machen das Abfragen der Sensoren und das steuern der Aktoren möglich.

4. Entwurf

4.1. Manöver, Strategien (Algorithmus)

Die auszuführenden Manöver, z.B. was soll nach Erkennen eines Hindernisses geschehen, müssen erkundet und getestet werden.
Mehr noch, welche Strategien müssen verfolgt werden, um z.B. den Ausgang aus einem Raum zu finden?

Strategien

1. Da die Lage der Öffnung im Raum unbekannt ist, kann man zu Beginn nicht entscheiden, ob man stets im Uhrzeigersinn oder gegen den Uhrzeigersinn drehen sollte. Willkürlich dreht sich der Roboter daher im Uhrzeigersinn.
2. Weiterhin sei die Fahrt vorwärts, wenn kein Hindernis auftritt, unbegrenzt.
3. Wird ein Hindernis erkannt, soll eine kurze Fahrt rückwärts (wenige Zentimeter) mit anschließender Drehung nach rechts ausgeführt werden.
4. Da der Ultraschallsensor nur einen Winkel von +/- 30 Grad abdeckt, können Hindernisse übersehen werden. Taster an der rechten und linken Außenseite sollen bei Kontakt eine Rückwärtsfahrt mit Drehung auslösen.

Manöver

Aktion		
1	Roboter startet mit Hinderniserkennung	Bevor eine Fahrt beginnen kann, wird in Vorwärts-Richtung mit dem Ultraschallmesser der Abstand zu einem Hindernis gemessen. Ist der Abstand < 10 cm, dann Aktion „4“ sonst Aktion „3“ Fahrt vorwärts.
2	Fahrt vorwärts	Kein Hindernis, dann Fahrt vorwärts.
3	Fahrt rückwärts	Hindernis nach Aktion „2“, dann einige Zentimeter rückwärtsfahren, dann 20 Grad nach rechts drehen.
4	Taster rechts: Fahrt rückwärts, Drehung nach links	Bei Kontakt einige Zentimeter rückwärtsfahren, dann 20 Grad nach links drehen.
5	Taster links: Fahrt rückwärts, Drehung nach rechts	Bei Kontakt einige Zentimeter rückwärtsfahren, dann 20 Grad nach rechts drehen.
6	Rundumblick (noch nicht fertig)	Ein Ultraschallsensor, über einen Servo um 180 Grad gedreht, sucht nach einer Öffnung im Raum. Ist eine vorhanden, dann Fahrt vorwärts. Nicht ideal, da Öffnungen nach hinten nicht gefunden werden.

5. Test der Komponenten

5.1. Taster

Aufgabe	Typ	
Roboterstart Roboterstopp Bumper links und rechts	Taster Minitaster	Funktionen: pinMode(intPin, INPUT), intPin = 7 digitalRead(intPin) Konstanten: true oder HIGH oder „1“ oder alle Werte außer „0“ sind true false oder LOW oder „0“ Schaltplan: Pullup-Widerstand / Pulldown-Widerstand? Welche Spannung am PIN ergibt HIGH / LOW? true/false sind Boolean Konstanten HIGH/FALSE sind Pin-Niveaus: 5 Volt/0 Volt
Versuche Taster Versuche Potentiometer		Siehe Anlage 8.1

5.2. Ultraschallmesser

Aufgabe	Typ	
Entfernung messen	Ultaschall HC-SR04	Funktionen: pinMode(intTrigPin, OUTPUT), intTrigPin = 11 pinMode(intEchoPin, INPUT), intEchoPin = 10 digitalWrite(intTrigPin, HIGH) delayMicroseconds() pulseIn(intEchoPin, HIGH) Berechnung: s = ½ * v * t t in µs, v = 340 m/s, s in cm daher: s = t * 0.017 (Achtung Punkt = Komma) Die Funktion pulseIn() liefert t in µs
Versuche Ultraschallmesser		Siehe Anlage 8.2

5.3. Servo

Aufgabe	Typ	
Antrieb	Tower Pro SG 90 Reely RS-303	Bibliothek einbinden: #include <Servo.h> Objekt einer Klasse erzeugen: Servo objServo Funktionen: objServo.attach(), Servoobjekt mit PWM-Pin verbinden objServo.write(), Drehbefehl an Servo im Winkelmass
Versuche Servos		Siehe Anlage 8.3

6. Programmierung

Hier geht es darum, die Strategien und die Manöver mithilfe der Arduino Software (IDE basierend auf C++) umzusetzen.
Neben den Kontrollstrukturen (Sequenz, Auswahl, Schleife) ist es hilfreich, das Programm in selbst definierte Funktionen (auch Unterprogramm, Methoden genannt) zu gliedern.

6.1. Neue Funktionen

Neue Funktionen	**aufgerufen in**	**aufgerufen in Manöver! „switch“**	
driveLogic()	loop()		
isLeftaHole()	driveLogic()		
driveForward()		**F**	Vorwärtsfahrt
driveBackward()		**B**	Rückwärtsfahrt
leftTurn()		**L**	Rückwärtsfahrt links drehen
rightTurn		**R**	Rückwärtsfahrt rechts drehen
StopServoLeft()	loop() leftTurn() rightTurn()		
StopServoRight()	loop() leftTurn() rightTurn()		
UltrasonicSensor()	driveLogic() isLeftaHole		
FrontBuffer()	driveLogic()		
Lcd_Write()	setup() loop() isLeftaHole() UltrasonicSensor()	Bei Manöver: F, B. L, R	

6.2. Schaltzentrale

Stets werden die Schritte

- Sensoren erfassen, auswerten
- Manöver auswählen
- Manöver ausführen

innerhalb der vorgegebenen loop()-Funktion endlos (Unterbrechung durch Taster) durch die neue Funktion driveLogic() ausgeführt. Der Status der Sensoren wird umgesetzt in Manöver. Jedes Manöver ist durch einen Buchstaben gekennzeichnet.

6.3. Sensor -> Manöver => Strategie

Sensor	**Manöver**	**Buchstabe**
Ultraschallmesser	Fahrt vorwärts	**F**
Ultraschallmesser	Fahrt rückwärts	**B**
Taster rechts	Fahrt rückwärts, Drehung nach links	**L**
Taster links	Fahrt rückwärts, Drehung nach rechts	**R**

In der Reaktion auf die Sensoren, umgesetzt in Manöver, ist die **Strategie** des Roboters verborgen, die ihm gestellten Probleme zu lösen.

Es stellt sich heraus, dass in diesem Progammabschnitt der zeitaufwendigste Teil des Projektes verborgen ist.

6.4. Programm

Eine erste Version des Programms findet sich im Anhang.

7. Zusammenfassung

Die Sensoren, Ultraschallmesser und Taster, erfüllen die Erwartungen.
Die Aktoren, also die Servos, bewegen den Roboter mit der nötigen Genauigkeit.
Die Manöver, wie „unbegrenzte Fahrt vorwärts“, sind schnell zu entwickeln.
Der „Teufel im Detail“ steckt in den Strategien! In nicht reckeckigen Räumen fährt sich der Roboter häufig in eine Falle, d.h. alle Manöver führen in eine endlose Wiederholung.

8. Anlage

8.1. Versuche Taster

zu Kapitel 5.1 Taster

Problem Zeigen, dass sich bei 5V „HIGH / true / 1“ als Vergleichswerte ergeben.
Bei 0V sollten sich „LOW / false / 0“ ergeben.

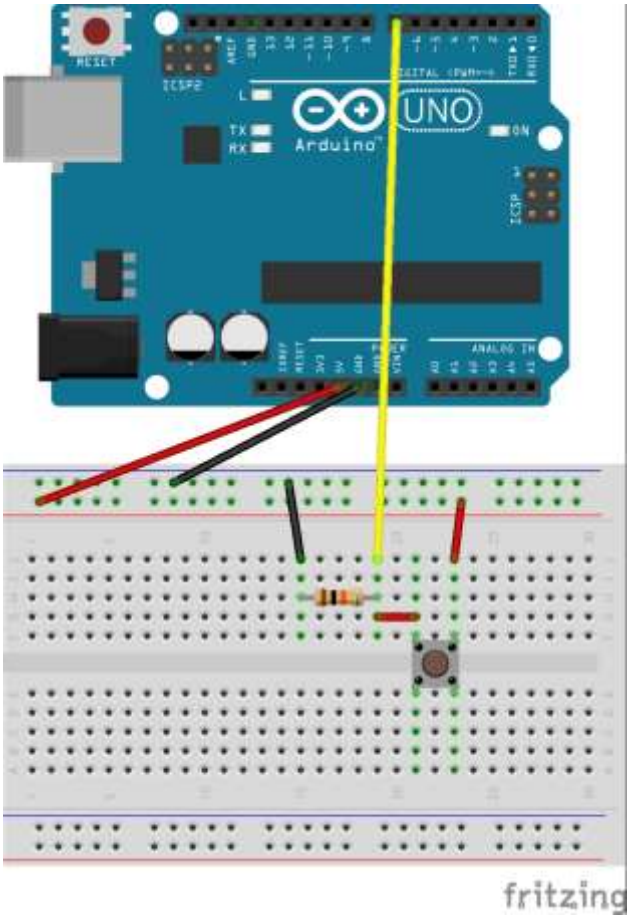
Arduino Programm

```
// EBW (All_digitalRead)
// Taster, pinMode(inPin, INPUT)
int intInPin = 7;            // Taster verbunden mit Pin 7
int intSensorVal = 0;        // Variable, die den gelesenen Wert speichert

void setup()
{
  pinMode(intInPin, INPUT);      // Setzt den digitalen Pin 7 auf INPUT
  Serial.begin(9600);            // Vorbereiten der Anzeigen im Monitor
(Strg + Umschalt + M)
}

void loop()
{
  //LOW entspricht "false" entspricht "0" entspricht < 3 Volt
  //IGH entspricht "true" entspricht "1" entspricht > 3 Volt
  intSensorVal = digitalRead(intInPin);          // Lese von Pin inPin
  if (intSensorVal == LOW) Serial.println("LOW");
  if (intSensorVal == HIGH) Serial.println("HIGH");
  if (intSensorVal == false) Serial.println("false");
  if (intSensorVal == true) Serial.println("true");
  if (intSensorVal == 0) Serial.println("0");
  if (intSensorVal == 1) Serial.println("1");
  delay(2000);
}
```

Versuche

1. Programm starten.
   Pin 7 abwechselnd mit GND und 5V verbinden.
2. Programm starten.
   Pin 7 über Widerstand 10 kOhm abwechselnd mit GND und 5V verbinden.
   Der Widerstand wirkt dann als Pulldown- oder Pullup-Widerstand.
   Dabei delay() klein wählen.
3. Pin 7 über Widerstand 10 kOhm mit GND verbinden (Pulldown).
   Taster mit 5V verbinden (siehe Schaltplan 1).
   Programm starten.
   Taste drücken sollte dann HIGH ergeben.

Schaltplan 1



Versuche zur Funktion eines „Digital-Pin“ gesetzt als INPUT.

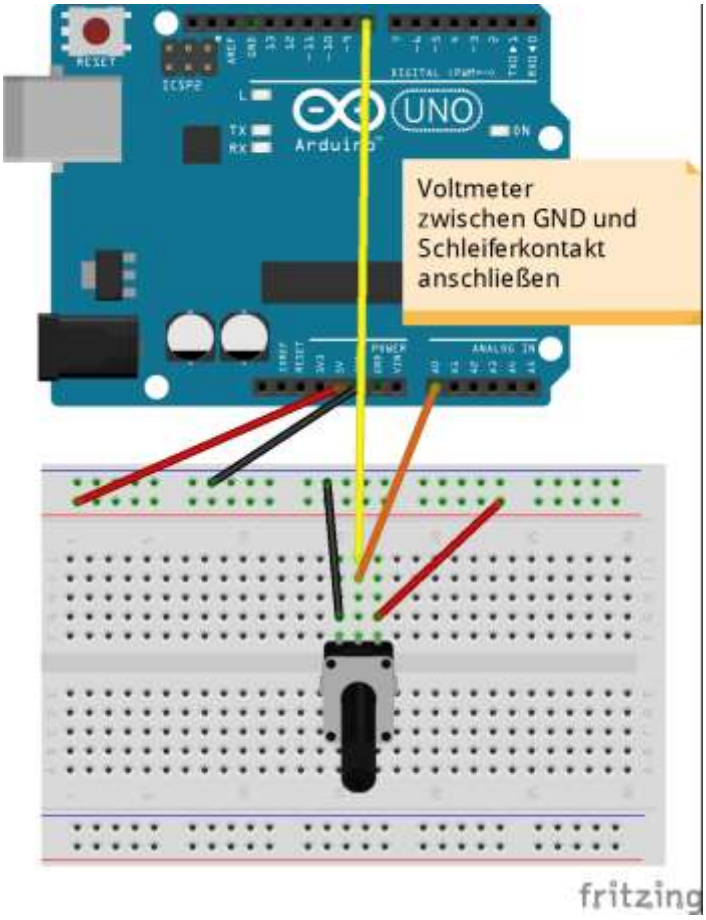
Problem Bei welchen Spannung geht der Pin auf LOW oder HIGH?

Arduino Programm

```
// EBW (All_Poti)
// Potentiometer 10kOhm, bei welcher Spannung kommt HIGH / LOW
int intPotPinA0 = A0;     // Schleiferkontakt an Analog-Pin A0
int intPotPin = 8;       // Schleiferkontakt verbunden mit Pin 8
int intSensorVal = 0;     // Variable, die den digitalen Wert speichert
int intPotVal = 0;        // Variable, die den analogen Wert speichert
float flVolt = 0;         // Umrechnung in Volt
float flVoltmeter = 4.83; // Mit Voltmeter gemessene max. Spannung

void setup()
{
  pinMode(intPotPin, INPUT);  // Pin 8 auf INPUT
  Serial.begin(9600);
}

void loop()
{
  intSensorVal = digitalRead(intPotPin);  // Lese von intPotPin
  intPotVal = analogRead(intPotPinA0);    // Lese von intPotPinA1
  flVolt = map(intPotVal, 0, 1023, 0, 255);
  flVolt = flVolt/255 * flVoltmeter;    // Umrechnung in Volt

  if (intSensorVal == LOW) Serial.println("LOW");
  if (intSensorVal == HIGH) Serial.println("HIGH");

  Serial.print("flVolt");
  Serial.print("\t");
  Serial.println(flVolt);
  delay(1000);
}
```

Versuch 4. Programm starten.
Schleiferkontakt zwischen den Endstellungen drehen.
Grenzspannung für HIGH und LOW ermitteln.

Schaltplan 2



8.2. Versuche Ultraschallmesser

zu Kapitel 5.2 Ultraschallmesser

Problem
Welche Hindernisse werden erkannt?
Ermittlung des Messbereichs!
Zuverlässigkeit der Messungen!
Verschiedene Längen des Trigger-Signals testen!

Arduino Programm

```
// EBW (All_HCSR04)
// Ultraschall-Entfernungsmesser, HC-SR04
// Der Trigger-Pin wird für minimal 10 µs auf HIGH gesetzt
// Der Ultraschallsensor erzeugt dann 8 40kHz Rechtecksignale
// Der Ultraschallsensor erkennt die Laufzeit eines Signales
// Der ECHO-Pin gibt einen Impuls aus, das der Signallaufzeit entspricht
// Die Funktion pulseIn() gibt die Laufzeit in µs aus
// s = 0.5 * t * v = 0.5 * 340 m/s * t = 0.017 * t [s in cm, t in µs]
// HC-SR04 Kontakte:
// Vcc to Arduino 5V
// Gnd to Arduino GND
// Trig to Arduino Pin 11
// Echo to Arduino Pin 10

int intTrigPin = 11;                // TRIGGER verbunden mit Pin 11
int intEchoPin = 10;                // ECHO verbunden mit Pin 10
double dblDuration;                 // Laufzeit in µs am Pin 10
double dblDistance;                 // Laufzeit in cm
int intTrigTime = 100;

void setup() {
  pinMode(intTrigPin, OUTPUT);      // Setzt den digitalen Pin 7 auf OUTPUT
  pinMode(intEchoPin, INPUT);
  Serial.begin(9600);
}

void loop() {
  digitalWrite(intTrigPin, HIGH);           // Trigger-Signal ein
  delayMicroseconds(intTrigTime);           // Dauer des Trigger-Signals
  digitalWrite(intTrigPin, LOW);            // Trigger-Signal aus
  dblDuration = pulseIn(intEchoPin, HIGH);  // Laufzeit hin und zurück in
                                            // µs (Mikrosekunden
  Serial.println(dblDuration);
  dblDistance = dblDuration * 0.017;     // Berechnung der Entfernung in cm
  Serial.print("Entfernung:");
  Serial.println(dblDistance);
  delay(2000);
}
```

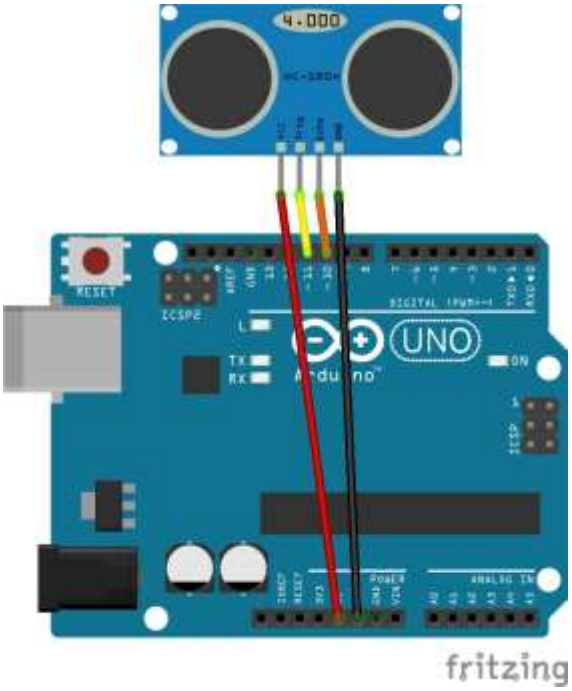
Versuche

5. Programm starten.
Verschiedene Gegenstände vor den Ultraschallsensor halten.
(Bleistift, Lineal, DIN A4 Blatt)

6. Programm starten.
Messbereich von-bis ermitteln.

7. Programm starten.
Einfluss der Länge des Trigger-Signals testen (10 µs, 100 µs, 1000 µs)

Schaltplan 3



8.3. Versuche Servos

zu Kapitel 5.3 Servo

Probleme
Externe Spannungsversorgung erforderlich?
Stromstärken im Servo-Betrieb?
Servo für kontinuierliche Drehung umbauen.

Arduino Programm

```
// EBW (All_Servo_H_2)
// Steuerung über serielle Eingabe
// Gehackter Servo, Drehwinkel 360 Grad
// REELY Servo RS-303
// Separate Spannungsversorgung erforderlich

#include <Servo.h>           // Bibliothek Servo einbinden

int intServoPin = 5;         // Digital-Pin 5 für Servo
int intServoVal;             // Winkel, daraus folgt Drehrichtung
                             // Geschwindigkeit Servo
int intServoCenter = 88;     // Ermittelte Mittelstellung = Stopp

Servo objServo;              // Objekt (objServo) der Klass Servo erzeugen

void setup()
{
  objServo.attach(intServoPin);  // Das Servoobjekt (objServo)
                                 // verbinden mit Digital-Pin 5
  Serial.begin(9600);
  Serial.print("Centerwert: ");
  Serial.println(intServoCenter);
  Serial.println("Moegliche Werte -20 H und +20 L");
                                   //High => im Uhrzeigersinn
                                   //Low  => entgegen Uhrzeigersinn
  intServoVal = intServoCenter;
}

 void loop()
{
   if (Serial.available() > 0) {
    // lese den Integerwert
    intServoVal = Serial.parseInt();
    intServoVal = intServoCenter + intServoVal;
    Serial.print("Drehwinkel: ");
    Serial.println(intServoVal, DEC);
  }
  objServo.write(intServoVal);    // Servo auf den einglesenen Wert setzen
  delay(500);
}
```

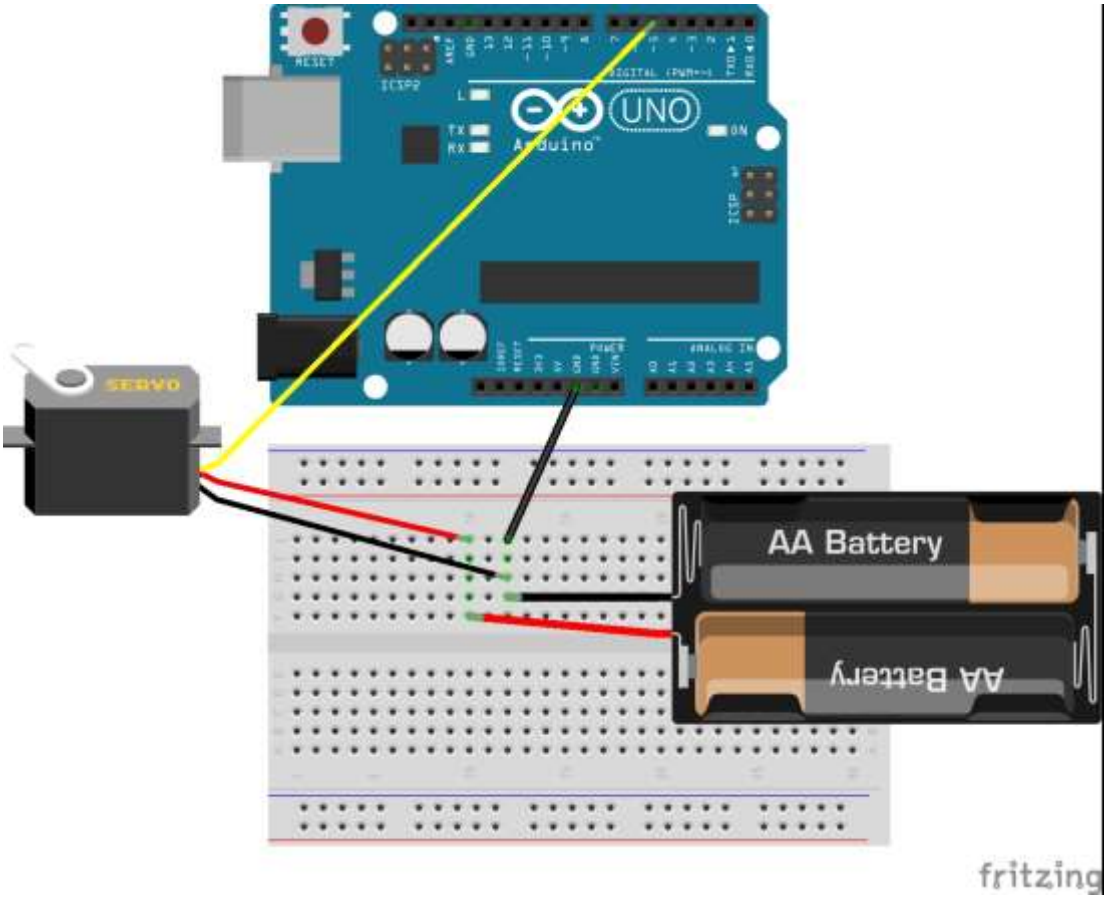
Versuche

8. Servo-Modelle auf Stromaufnahme prüfen!

9. Servo für kontinuierliche Drehung umbauen und kalibrieren!

10. Steuerung eines gehackten Servos.

Schaltplan 4


UNO
SERVO
AA Battery
AA Battery
fritzing

8.4. Technische Daten

Servo

Servo#IQ-400BB
- Kugelgelagert
- Staubdichtes Kunststoffgehäuse
- Kunststoffgetriebe aus schlagzähem Nylon
- Spezial-Poti mit Anti-Fading-Beschichtung
- Hohe Wiederkehrgenauigkeit
- Hohes Haltemoment
- JR/hitec-Steckersystem

Einsatzgebiet:
Motorflugmodelle bis 5 kg, RC-Cars 1:10 On Road

Technische Daten
Betriebsspannung: 4,8-6,0V;
Stellzeit (60°): 0,21 / 0,17s;
Stellmoment: 50 / 62 Ncm;
Gewicht: 46g;
Abmessungen:39,5x20,0x39,6mm
(Messwerte 4,8 / 6,0V)

8.5. Programm

```
//Servo10 ist die letzte Version!

#include <Servo.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

#define intButtonPin 2
#define intLBufferPin 3
#define intRBufferPin 4
#define intServoLeftPin 5
#define intServoRightPin 6
#define intServoLeftCenter 85               // Mittelstellung des linken Servos
#define intServoRightCenter 93              // Mittelstellung des rechten Servos
#define intFTrigPin 11                      // Ultraschall Trigger-Pin
#define intFEchoPin 10                      // Ultraschall Echo-Pin

LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
Servo objServoLeft;
Servo objServoRight;
boolean bolOnState = false;                 // Speichert das Drücken der Starttaste
double dblFDuration, dblFDistance;
char charWhatToDo, charWhereIsHole = 'X';    // X  = nicht bestimmt

void setup() {
  pinMode(intButtonPin, INPUT);
  pinMode(intRBufferPin, INPUT);
  pinMode(intLBufferPin, INPUT);
  pinMode(intFTrigPin, OUTPUT);
  pinMode(intFEchoPin, INPUT);

  lcd.begin(16, 2);                         // 16 Zeichen, 2 Reihen
  Lcd_Write(0, 0, "Starttaste?");           // Eigene Funktion zum Schreiben auf LCD
} // Ende setup

void loop() {
  if (digitalRead(intButtonPin) == HIGH && !bolOnState) {
    bolOnState = true;
    objServoLeft.attach(intServoLeftPin);
    objServoRight.attach(intServoRightPin);
    delay(500);
    StopServoLeft();
    StopServoRight();
    Lcd_Write(0, 0, "Servos gestartet");
  }
  if (digitalRead(2) == HIGH && bolOnState) {
    bolOnState = false;
    objServoLeft.detach();
    objServoRight.detach();
    delay(500);
    StopServoLeft();
    StopServoRight();
    Lcd_Write(0, 0, "Servos gestoppt");
    Lcd_Write(0, 1, "");
  }
  if (bolOnState) driveLogic();               // Eigene Funktion für Fahraktionen
} // Ende loop

void driveLogic() {
  // Wird bei jedem Fahrt-Schritt einmal durchlaufen

  // Funktion isLeftaHole() ist ein Platzhalter für zukünfitige Weiterentwicklungen
  if (charWhereIsHole == 'X' ) isLeftaHole(); // Mögliche Werte: X, H, L
  // Aktuell dreht der Roboter immer im Uhrzeigersinn
```

```
  // Bisher prüft isLeftaHole() nur, ob im Winkel von 45 Grad
  // nach links eine Öffnung ist

  // Schaltzentrale: Alle Sensoren abfragen und Manöver bestimmen
  UltrasonicSensor();                          // Distanz zum Hindernis ermitteln
  if (dblFDistance >= 10)                      // Entfernung "größer gleich" 10 cm
  {
    charWhatToDo = 'F';  // Kein Hindernis, vorwärts fahren
  }
  else
  {
    charWhatToDo = 'B';  // Ein Hindernis, rückwärts fahren
  }
  if (FrontBuffer() == 1) charWhatToDo = 'R'; // linke Buffer, d.h. RechtsDrehen
  if (FrontBuffer() == 2) charWhatToDo = 'L'; // rechte Buffer, d.h. LinksDrehen

  // Aktion: Manöver erledigen
  switch (charWhatToDo) {
    case 'F':
      Lcd_Write(0, 1, "Schall: Vor");
      driveForward(10);                        // Vorwaertsfahrt, Parameter
Geschwindigkeit "10"
      // steht für Drehgeschwindigkeit
      break;
    case 'B':
      Lcd_Write(0, 1, "Schall: Zurueck");
      driveBackward(10);                       // Rueckwaertsfahrt,
      delay(500);                              // Parameter "500", Dauer der Aktion
      rightTurn(20);                           // Rechtsdrehen, Parameter "20" steht
für Winkel

      break;
    case 'R':
      Lcd_Write(0, 1, "Buffer: Links");
      driveBackward(10);                       // Rueckwaertsfahrt, Parameter
Geschwindigkeit "10"
      delay(500);                              // Parameter "500", Dauer der Aktion
      rightTurn(20);                           // Rechtsdrehen um 20 Grad
      break;
    case 'L':
      Lcd_Write(0, 1, "Buffer: Rechts");
      driveBackward(10);                       // Rueckwaertsfahrt
      delay(500);                              //Parameter "500", Dauer der Aktion
      leftTurn(20);                            // Linksdrehen um 20 Grad
      break;
  }
} // Ende driveLogic

// Drehe nach links und schaue nach einer Öffnung, sonst drehe zurück
void isLeftaHole() {
  // Ausgangswert charWhereIsHole = 'X', d.h. aktuell nur einmalige Prüfung am Anfang
  // Ist links eine Öffnung, dann "L" für Low Pressure (Tief = linksdrehend)
  // sonst                        "H" für High Pressure (Hoch = rechtsdrehend)
  leftTurn(45);
  UltrasonicSensor();
  if (dblFDistance >= 10)                      // Entfernung "größer gleich" 10 cm
  { Lcd_Write(0, 1, "Links ein Loch");
    charWhereIsHole = 'L';
  }
  else
  { Lcd_Write(0, 1, "Links kein Loch");
    rightTurn(45);
    charWhereIsHole = 'H';
  }
}
```

```
// Vorwärtsfahrt, unbegrenzte Zeit
void driveForward(int intSpeed) {
  int intCorrectLeft = 0;
  int intCorrectRight = 0;
  objServoLeft.write(intServoLeftCenter + intSpeed + intCorrectLeft);
  objServoRight.write(intServoRightCenter - intSpeed + intCorrectRight);
}

// Rückwärtsfahrt, unbegrenzte Zeit
void driveBackward(int intSpeed) {
  int intCorrectLeft = +1;
  int intCorrectRight = 0;
  objServoLeft.write(intServoLeftCenter - intSpeed + intCorrectLeft);
  objServoRight.write(intServoRightCenter + intSpeed + intCorrectRight);
}

// Linksdrehen, begrenzte Zeit
void leftTurn(int intTime) {
  intTime = intTime * 40;                    // 90 Grad entspricht 3600 ms
  objServoLeft.write(intServoLeftCenter - 7);
  objServoRight.write(intServoRightCenter - 7);
  delay(intTime);
  StopServoRight();
  StopServoLeft();
}

// Rechtsdrehen, begrenzte Zeit
void rightTurn(int intTime) {
  intTime = intTime * 40;                    // 90 Grad entspricht 3600 ms
  objServoLeft.write(intServoLeftCenter + 7);
  objServoRight.write(intServoRightCenter + 7);
  delay(intTime);
  StopServoRight();
  StopServoLeft();
}

// Halt ServoLeft
void StopServoLeft() {
  objServoLeft.write(intServoLeftCenter);     // Mittelstellung gleich Server stopp
}
// Halt ServoRight
void StopServoRight() {
  objServoRight.write(intServoRightCenter);
}

// Ultraschall-Entfernungsmesser, HC-SR04 abfragen
void UltrasonicSensor() {
  digitalWrite(intFTrigPin, HIGH);
  delayMicroseconds(1000);
  digitalWrite(intFTrigPin, LOW);
  dblFDuration = pulseIn(intFEchoPin, HIGH);
  dblFDistance = dblFDuration * 0.017;
  Lcd_Write(0, 0, "Entfernung:");
  Lcd_Write(12, 0, String(dblFDistance));
}

//Bumper abfragen
int FrontBuffer() {
  int intStatusBuffer = 0;
  //intStatus 0: kein Buffer
  //intStatus 1: left Buffer
  //intStatus 2: right Buffer
  if (digitalRead(3) == HIGH) intStatusBuffer = 1;
  if (digitalRead(4) == HIGH) intStatusBuffer = 2;
```

```
  return  intStatusBuffer;
}

void Lcd_Write(int intColumn, int intRow, String strText)
{
  lcd.setCursor(intColumn, intRow);
  strText = strText + "               ";
  lcd.print(strText);
}
```