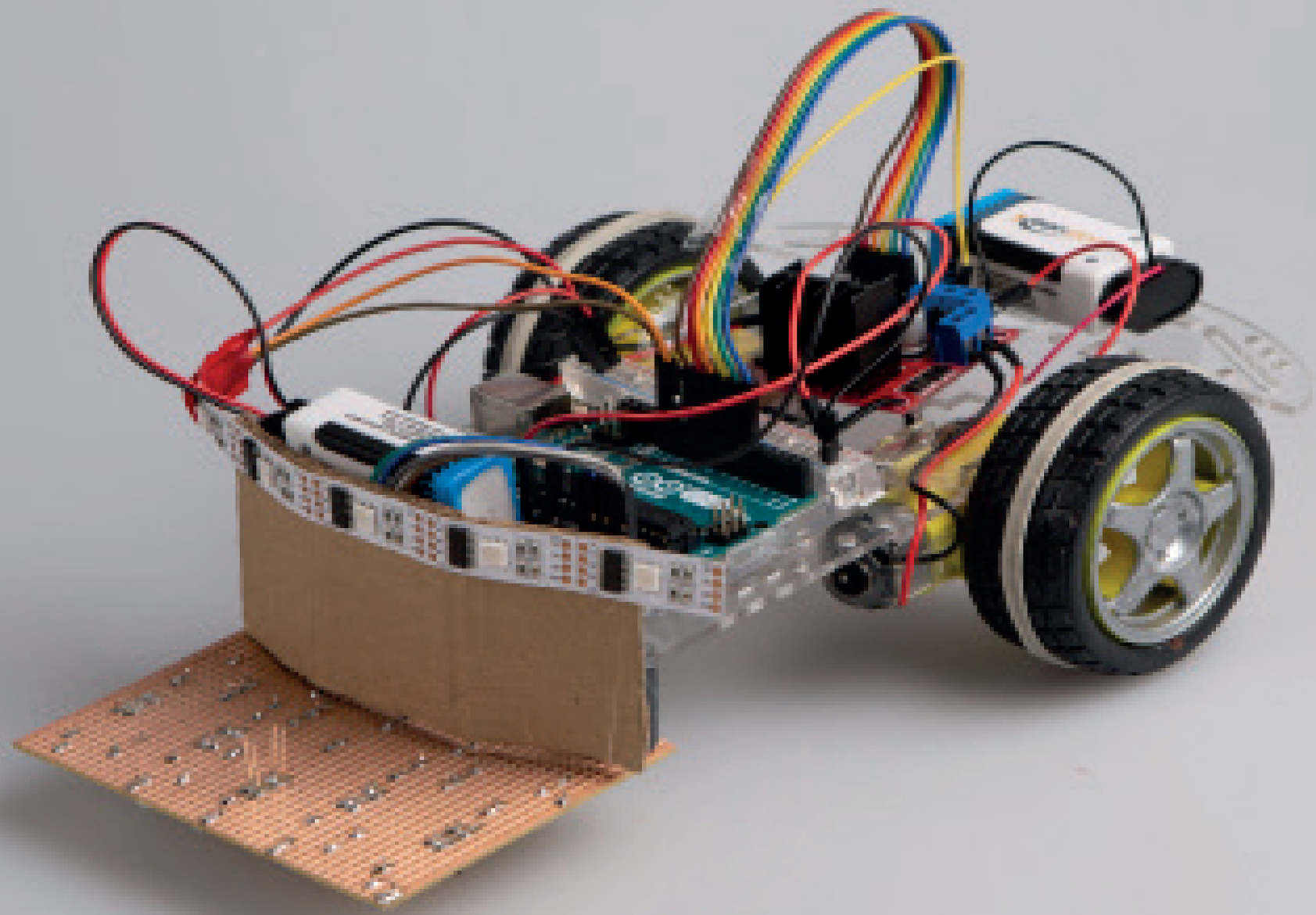
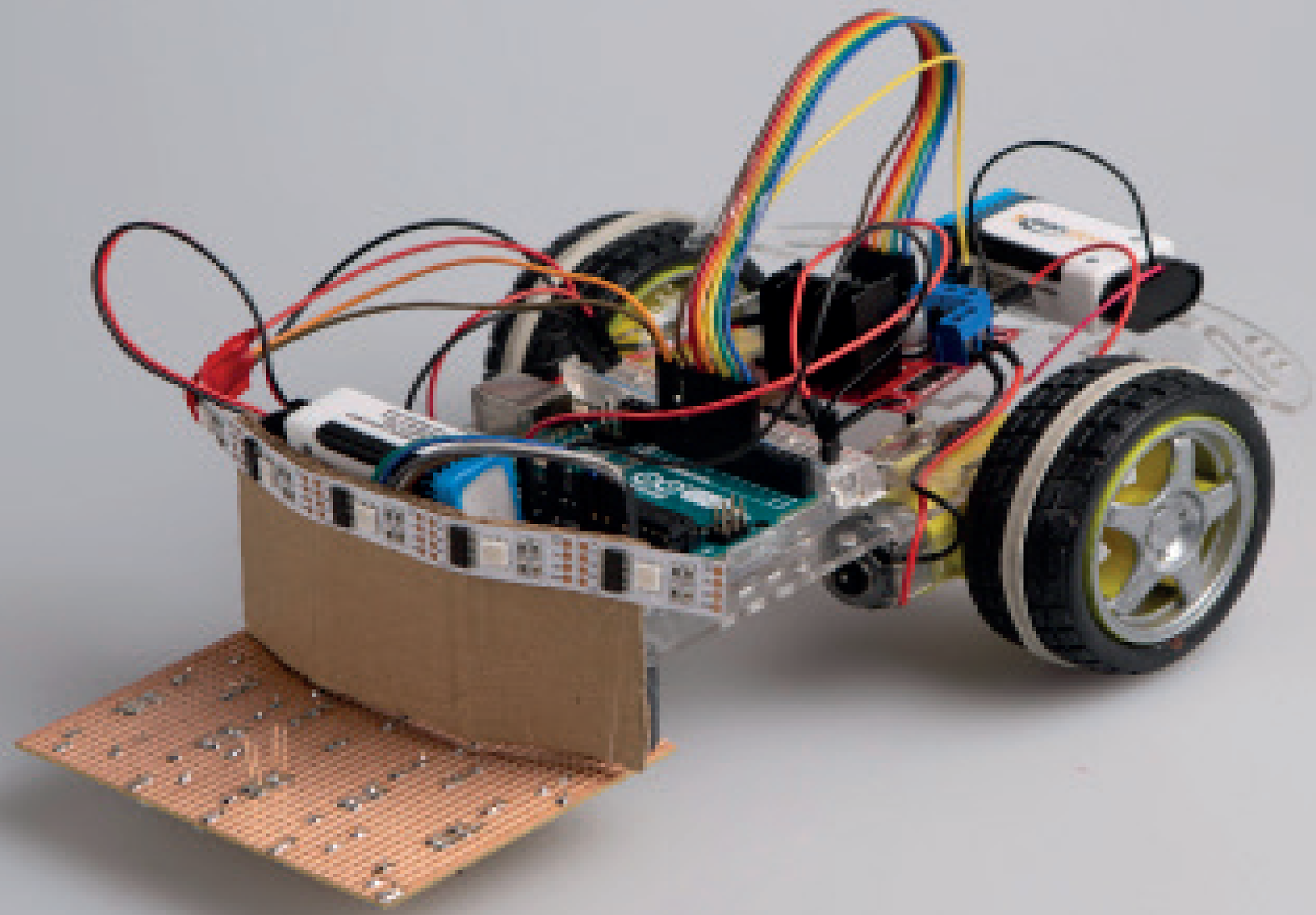
Einführung Technischs-Grundlagenprojekt

Dokumentation

Christian Kolb & Hartmut Idler



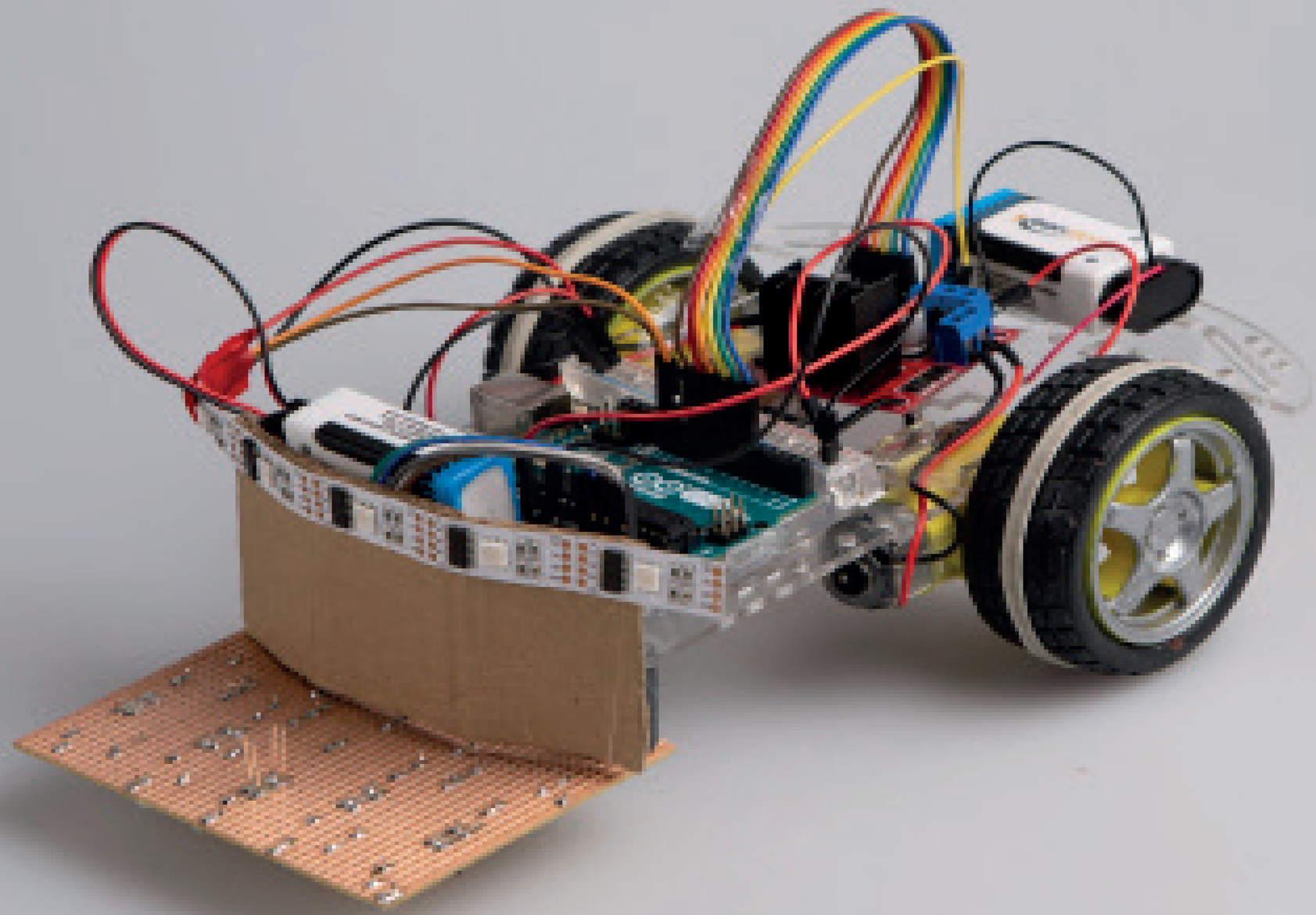
Gliederung

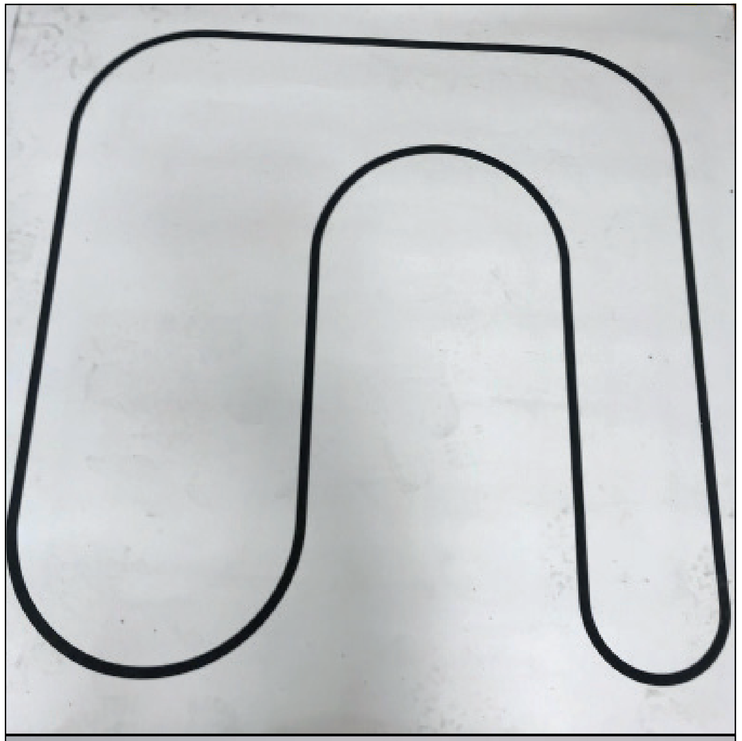


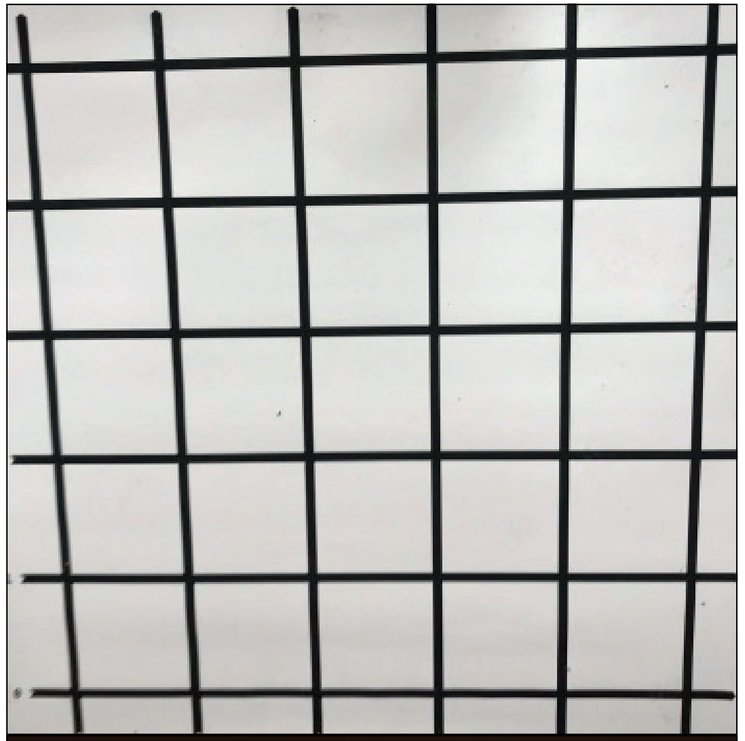
Einleitung

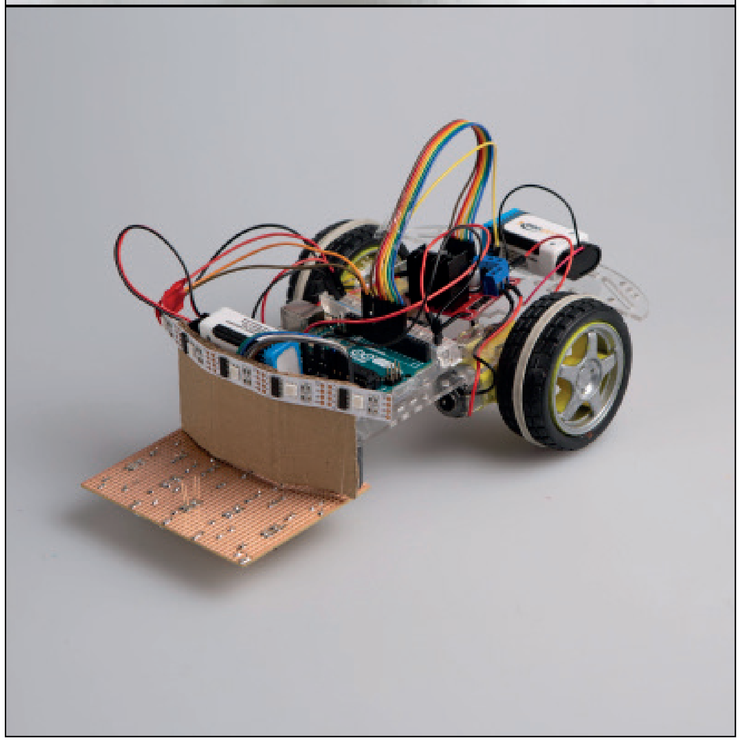
Dokumentation

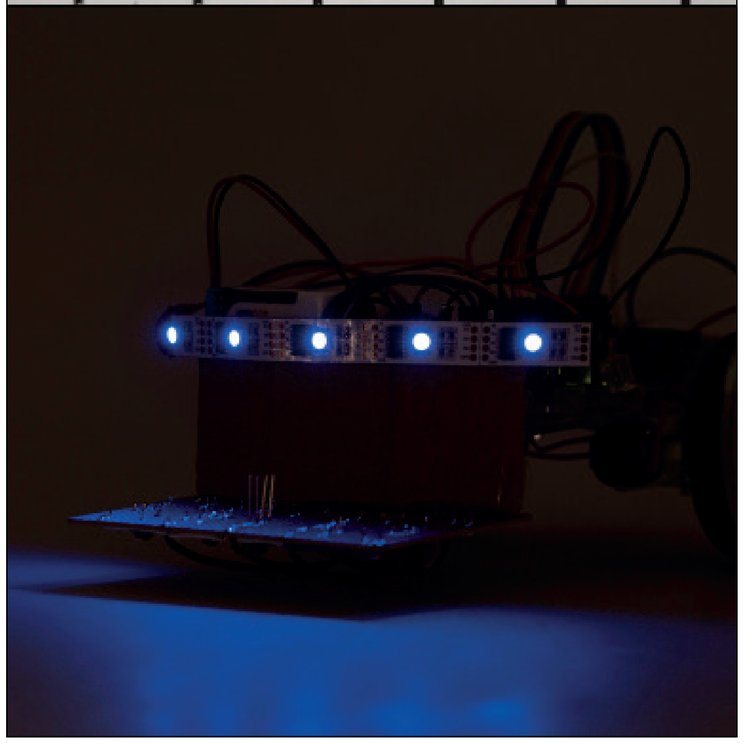
Christian Kolb & Hartmut Idler











Aufgabenstellung

Ziel der Vorlesung Technisches Grundlagenprojekt war es einen Line-Follower und einen Grid-Follower zu entwickeln, der sich dann auf dem entsprechenden Grid fortbewegen kann. Die Hardware und Software komponenten werden selbstständig miteinander verbunden.

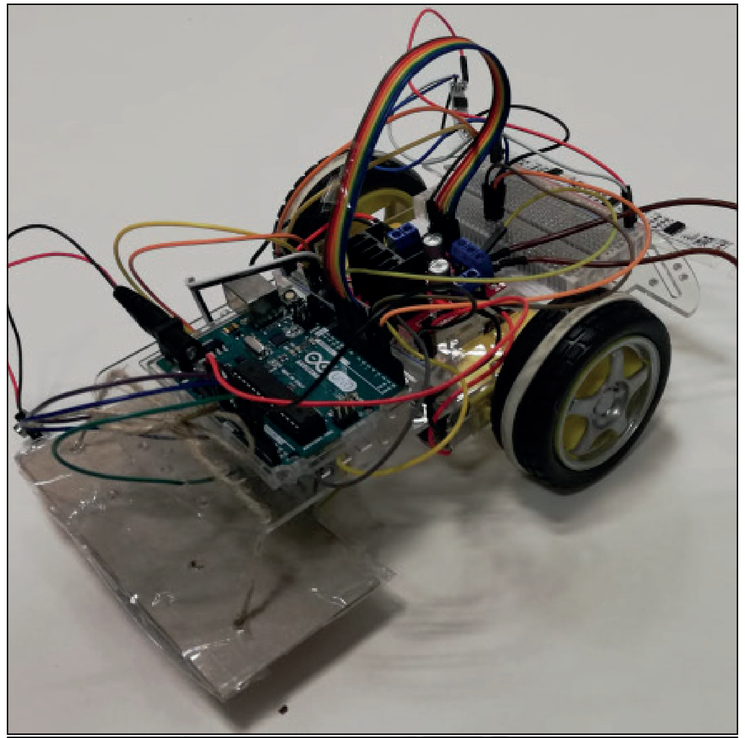
Die Umsetzung des ganzen geschah durch einen Arduino Uno sowie durch eine Platine mit fünf IR Sensoren, die zwei Gleichstrommotoren wurden über ein L298N Motor Shield angesteuert, über ein LED-Strip wurden die Sensor Aktivitäten visualisiert.

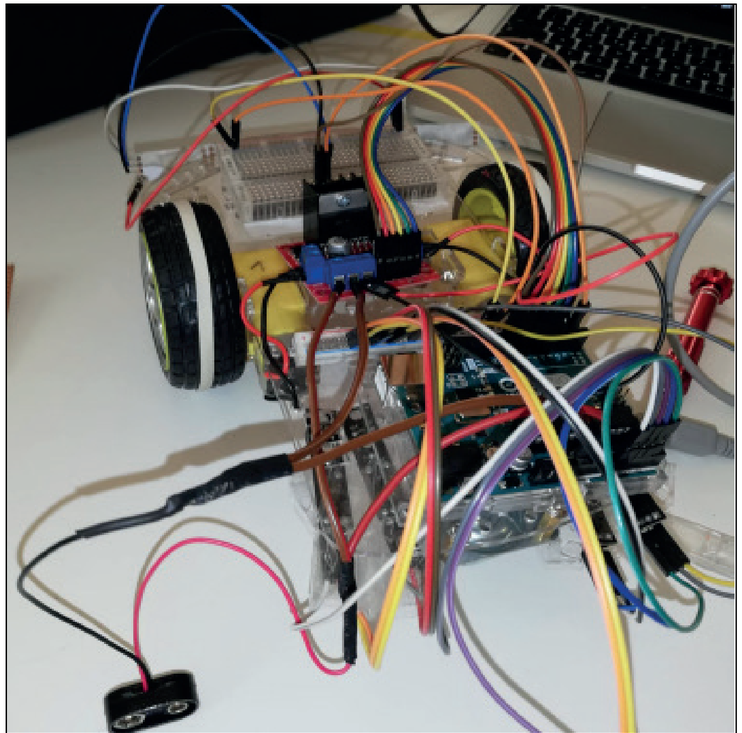
Projekt Prozess

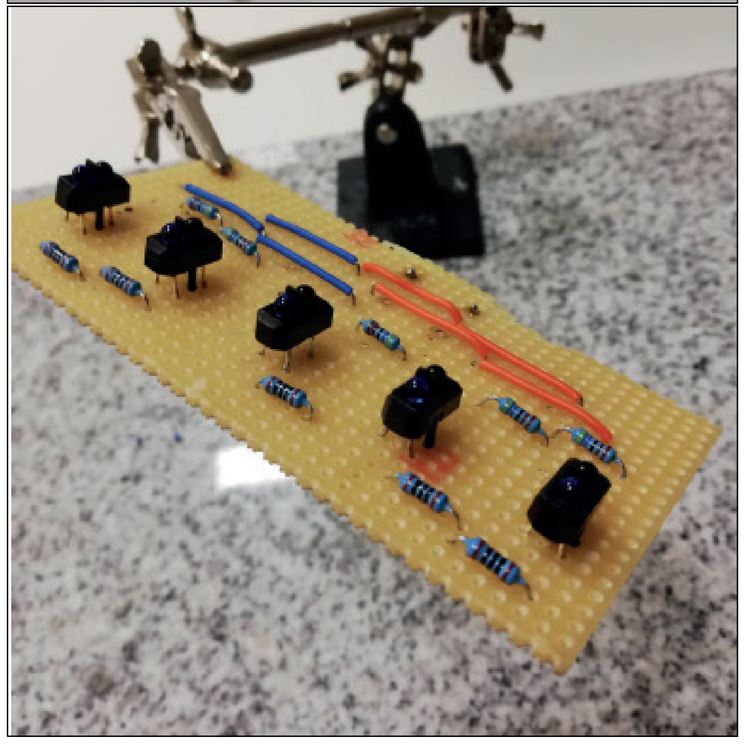
Zu Beginn musste man das Arduino-Kit zusammenbauen, die Motoren wurden an entsprechende stelle montiert und die Hardware wie das Arduino Uno Board, das L298N Motor Shield und ein Steckbrett wurden auf der Oberseite des Line/Grid Follower montiert.

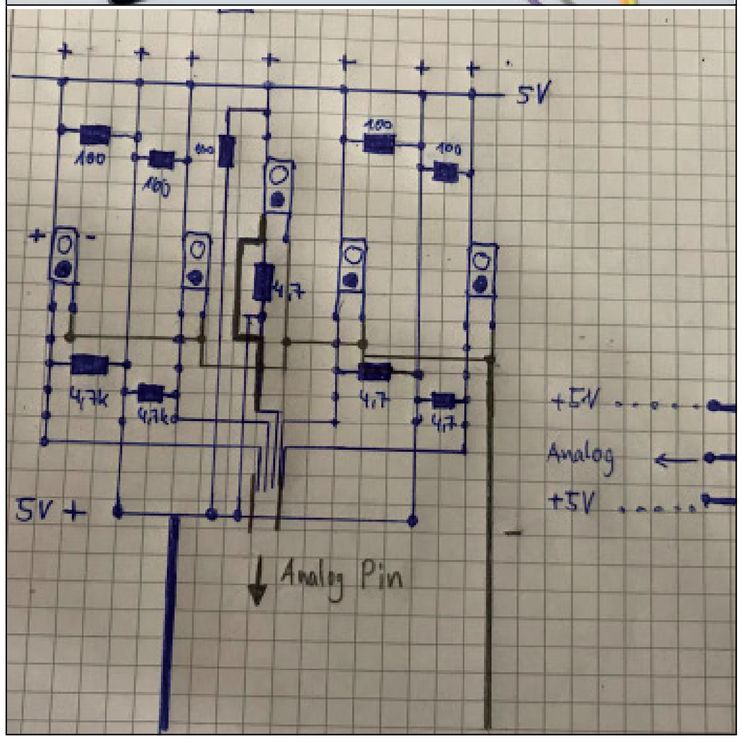
Die Platine mit den TCRT5000 Sensoren musste erst noch gesteckt und verlötet werden, dazu fertigten wir eine Skizze an wie die Sensoren und Widerstände auf die Platine kommen. Als Widerstand wurde einmal ein 4700 Ohm und ein 100 Ohm Widerstand gewählt. Das löten der Platine war schwerer als gedacht da wir bei dem ersten Versuch eine Platine mit einzel Loch hatten dort mussten die Leiterbahnen noch verbunden werden, leider funktionierte dann unsere Platine nicht, da zum einen die Masse nicht überall verbunden war und zwei TCRT Sensoren kaputt waren. Die Finale Sensor Platine wurde dann mit einer mehrloch Platine, einem sauberen verbinden zwischen Sensor, Widerstand und dem Plus und Minus Pol realisiert. Sie wurde dann an der vorderen Unterseite des Line/Grid Follower mit einem möglichst geringen Abstand zum Boden montiert

Der LED Strip wurde an der Fahrzeugvorderseite angebracht, um gut sichtbar zu sein.





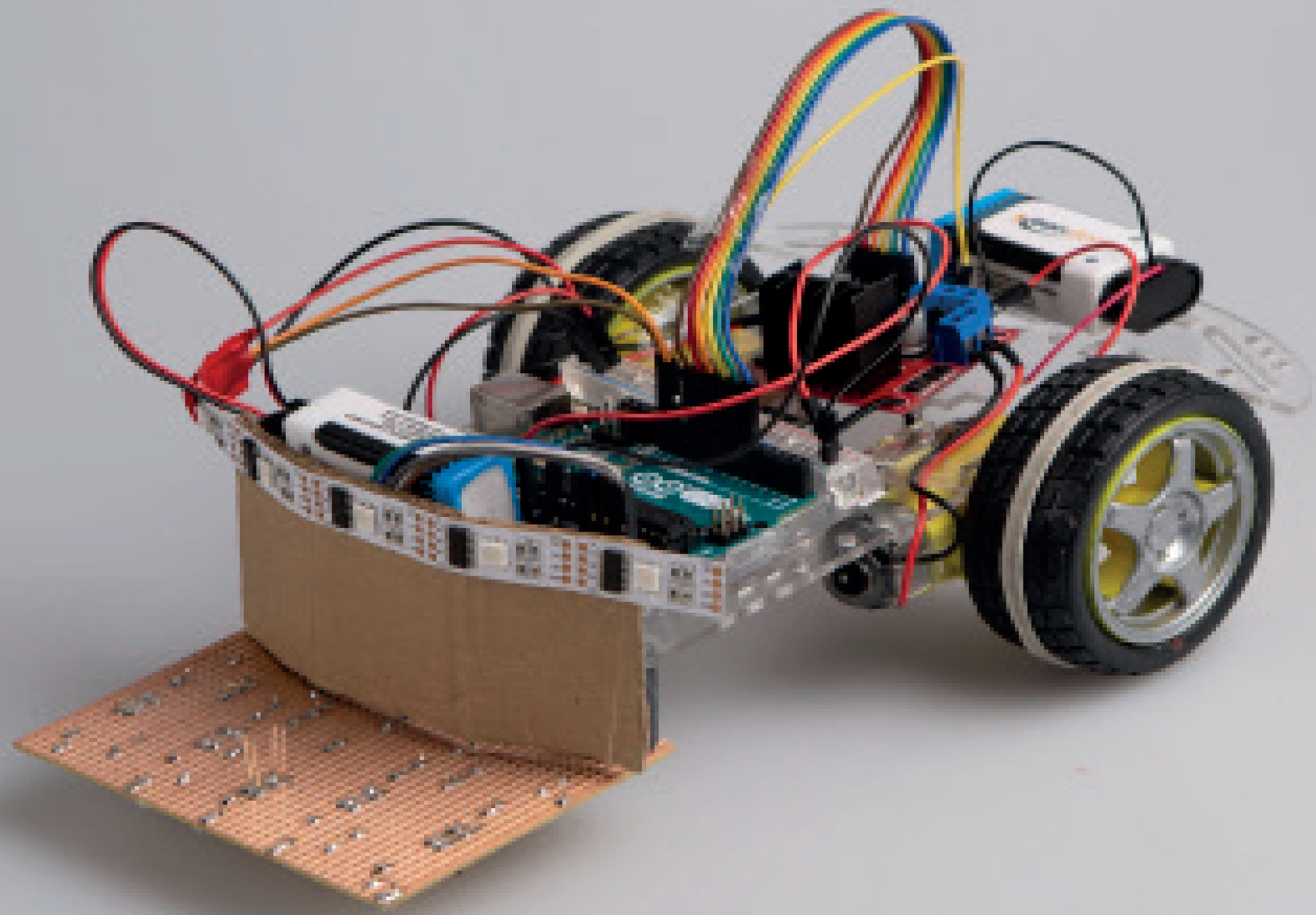




Hardware - Komponenten

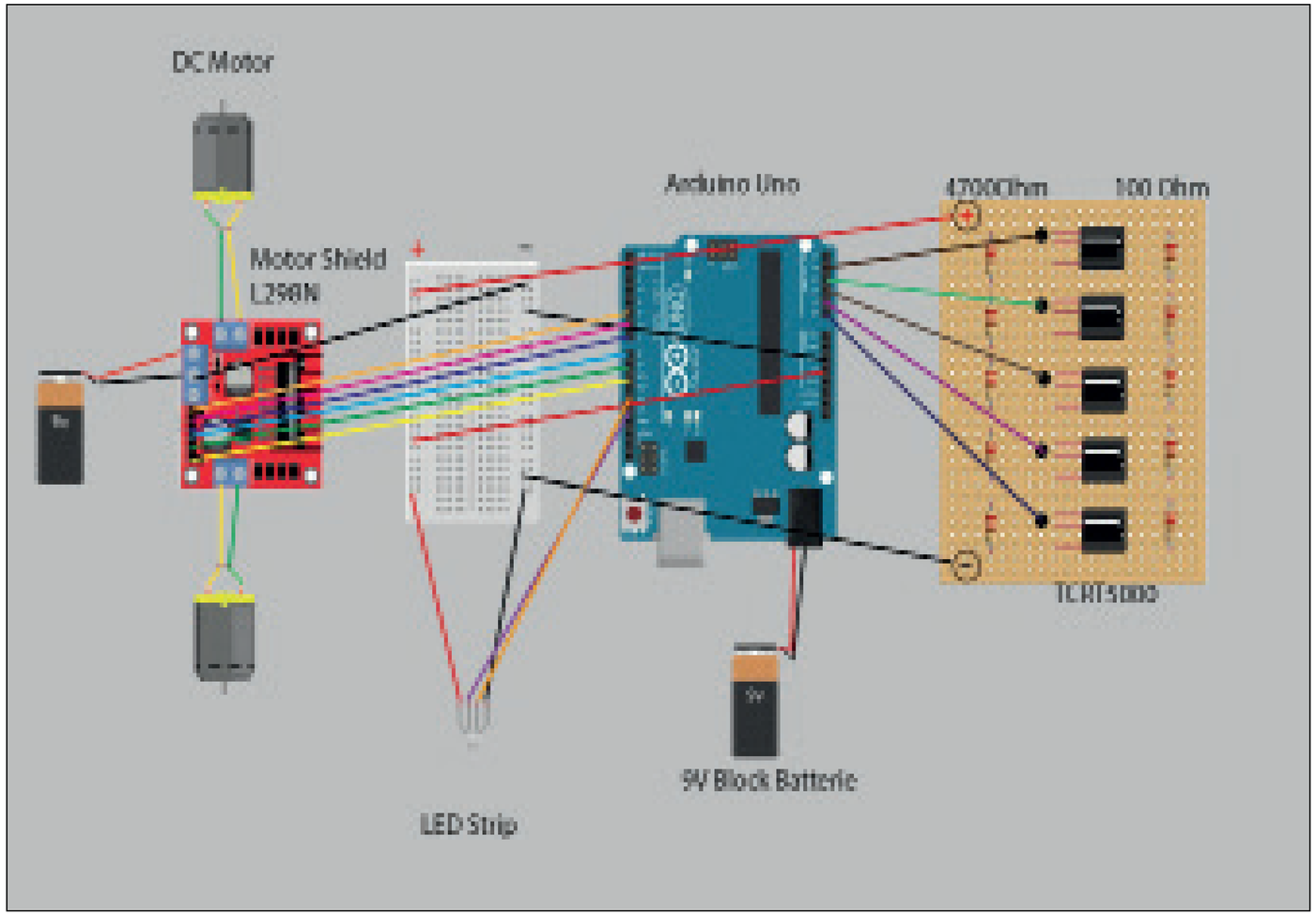
Dokumentation

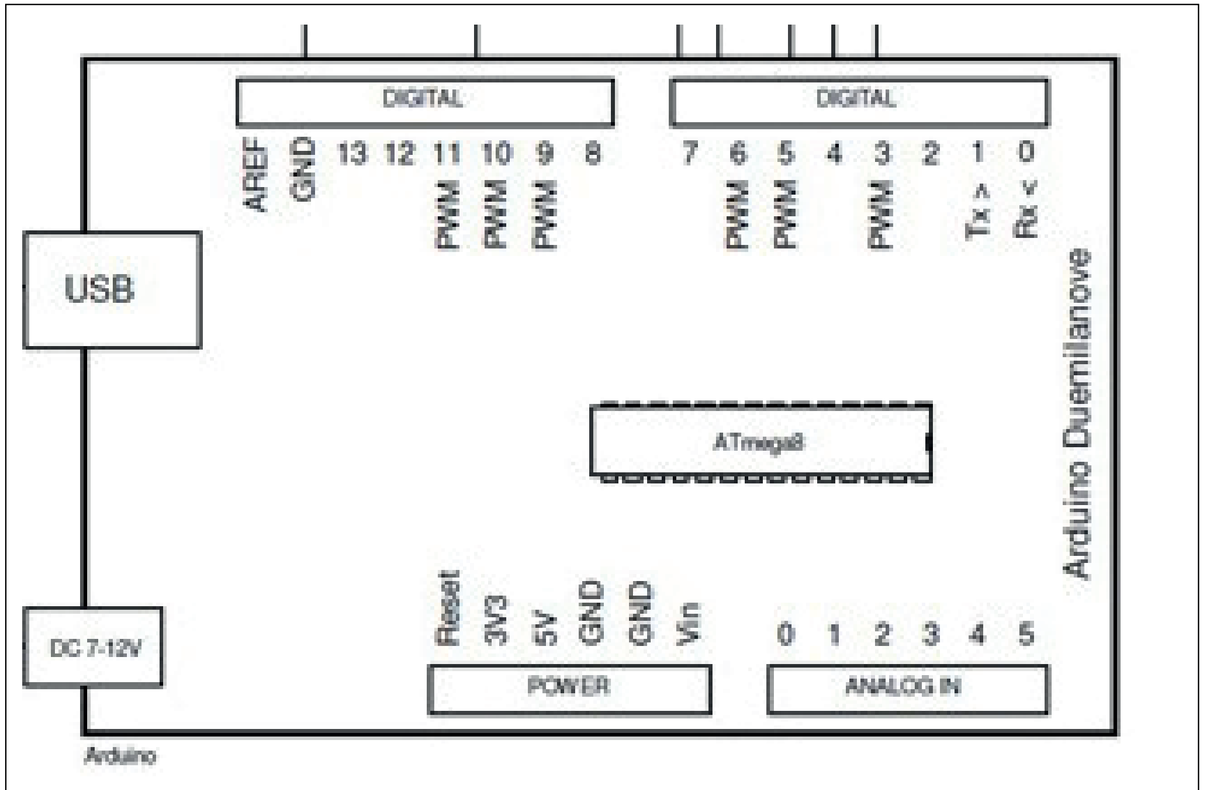
Christian Kolb & Hartmut Idler



Schaltplan

Zur besseren Übersicht haben wir einen Schaltplan angefertigt, dort kann man dann jede Verbindung zwischen den Komponenten sehen. Bei dem Anschluss der Motoren haben wir auf die Drehrichtung geachtet. Eine 9V Block Batterie versorgt die Gleichstrommotoren und die andere Batterie versorgt das Arduini Uno Board, die Sensor Platine sowie den LED Stripe.





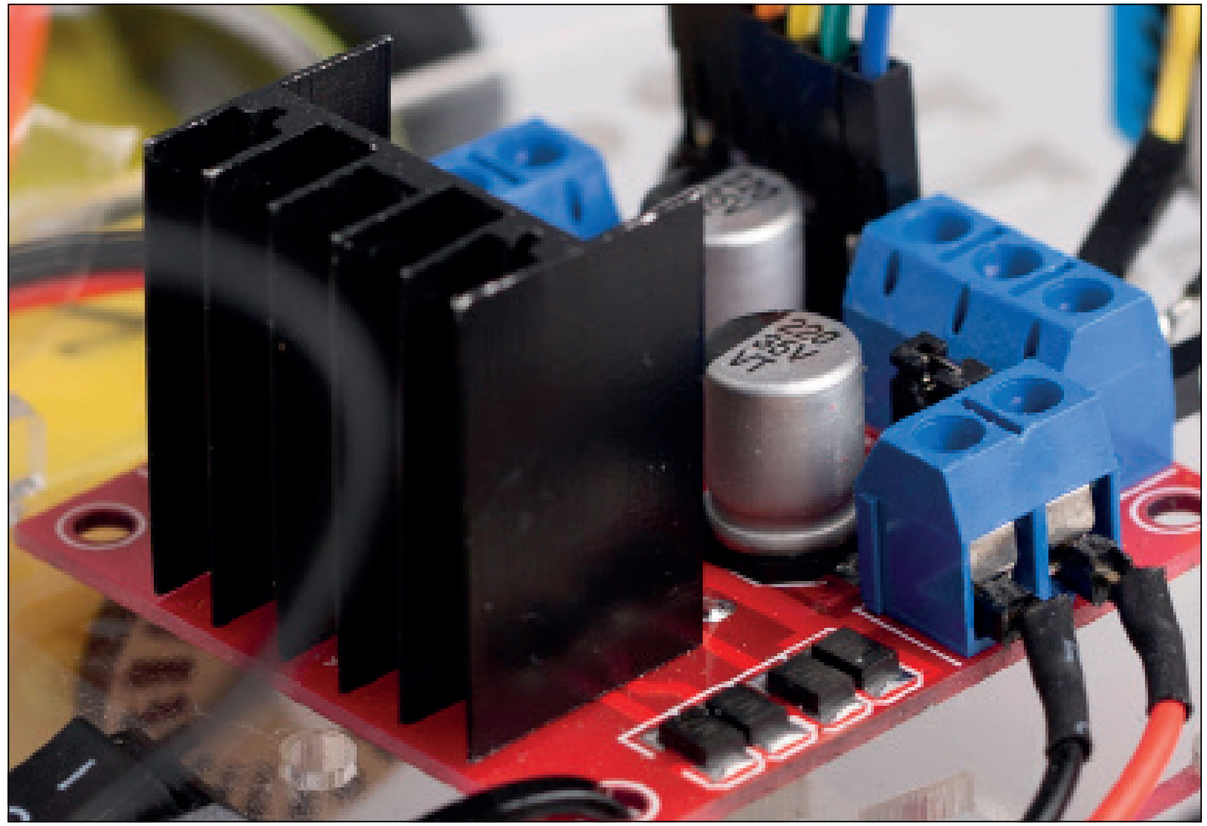
Arduino Uno

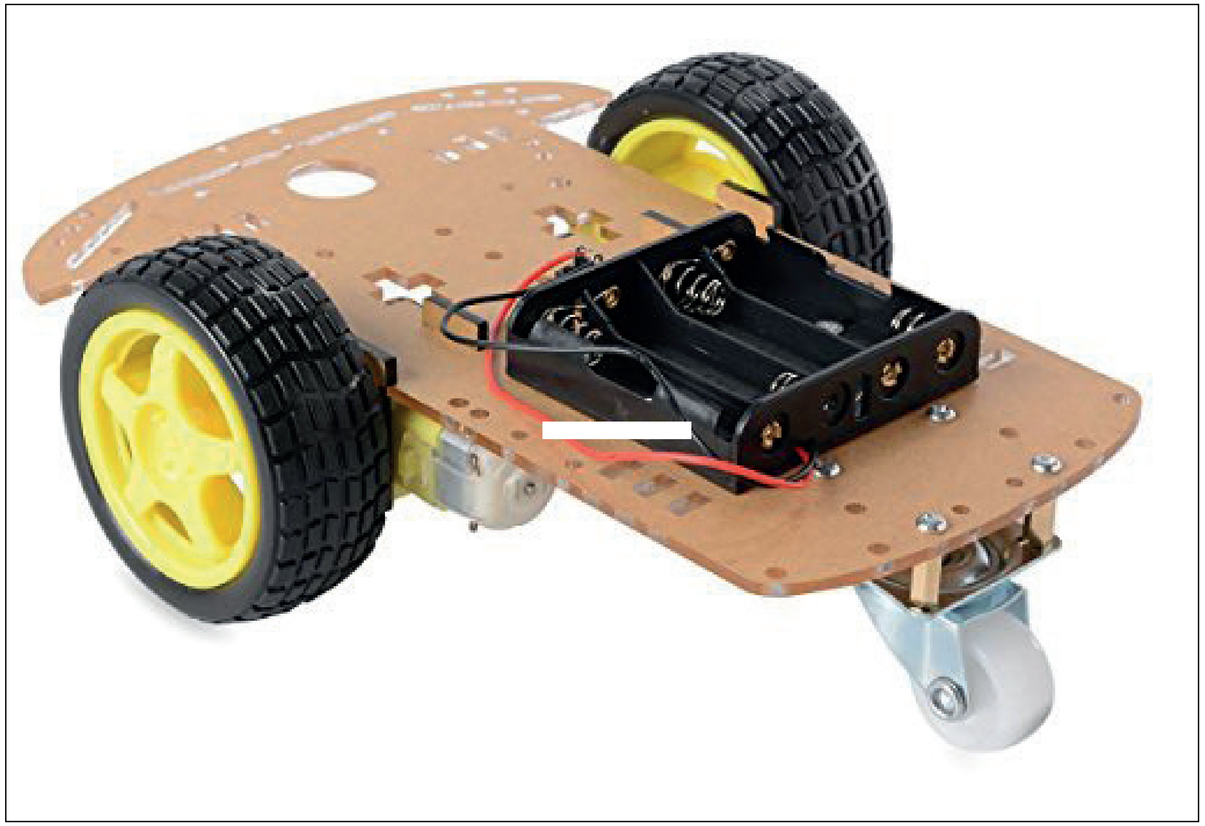
Die analogen Pins A0 bis A5 sind als Sensor-Eingänge zum Messen von Spannungswerten zwischen Null und Fünf Volt geeignet,durch einen eingebauten Analog-Digital-Wandler werden die gemessenen Spannungswerte auf einem Zahlenbereich von 0 (keine Spannung) bis 1023 (maximale Spannung, also Fünf Volt abgebildet.

Die digitalen Pins 0 bis 13 können ebenfalls als Sensor-Eingänge festgelegt werden: Eine anliegende Spannung von >2,5 Volt wird als HIGH (Zahlenwert 1), eine niedrigere Spannung als LOW (Zahlenwert 0) interpretiert.

Die digitalen Pins 0 bis 13 können zudem als digitale Spannungs-Ausgänge festgelegt werden: Sie geben im Modus HIGH eine Spannung von etwa 5 Volt, im Modus LOW eine Spannung von 0 Volt aus. Die Stromstärke ist dabei allerdings auf 40 mA begrenzt; gegebenenfalls wird die Spannung der Pins automatisch herab geregelt, um diese Begrenzung zu erreichen.

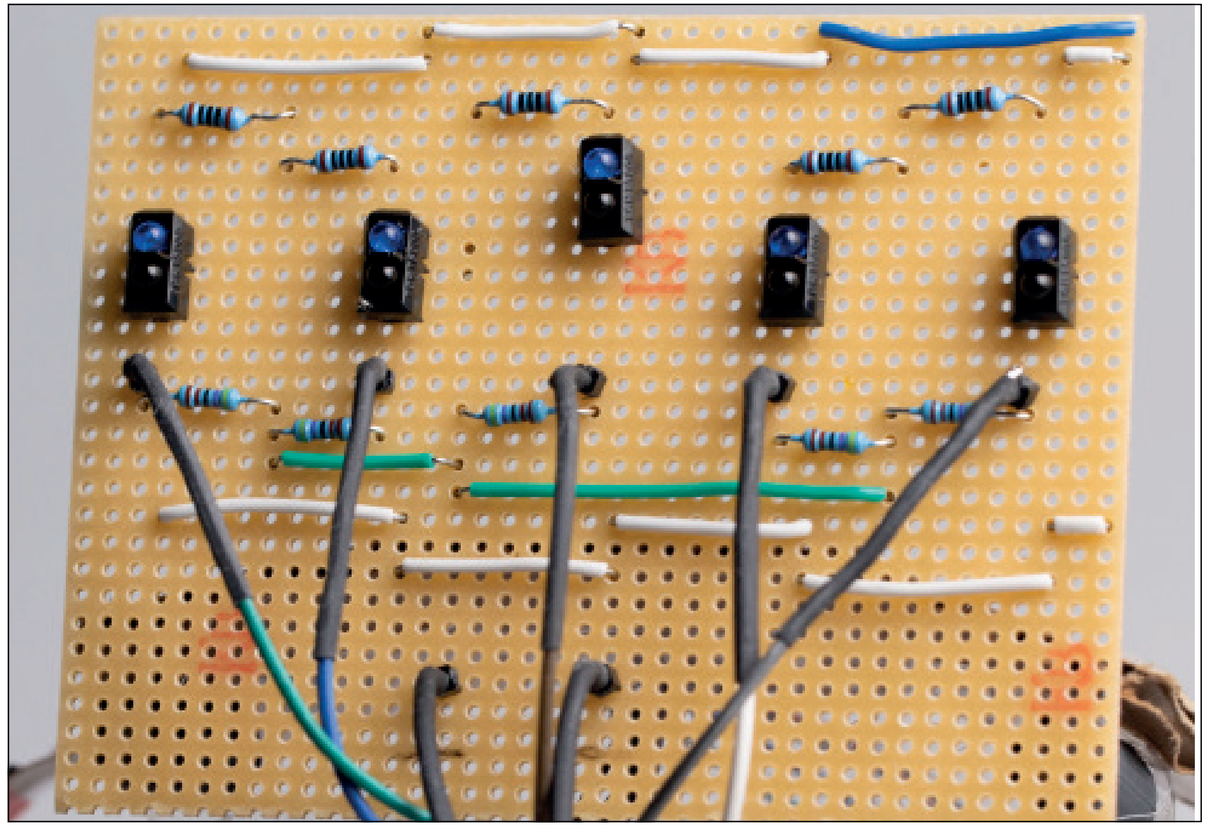
Eine Besonderheit stellt der Digital-Pin 13 dar: Dort ist der Ausgabe-Strom auf nur 20 mA begrenzt, so dass dort eine LED direkt (ohne Vorwiderstand) angeschlossen werden kann (direkt neben Pin 13 ist ein GND-Pin, so dass dafür nicht einmal eine Steckplatine nötig ist). Die mit dem Tilde-Zeichen ~ versehenen Pins (3, 5, 6, 9, 10, 11) können, wenn sie als Ausgabe-Pins festgelegt werden, zudem mittels einer so genannter Pulsweiten-Modulation (PWM) sehr schnell zwischen 0 Volt und 5 Volt hin und her wechseln. Man kann dabei Werte zwischen 0 und 255 angeben, wobei 0 für „immer aus“ und 255 für „immer an“ steht.

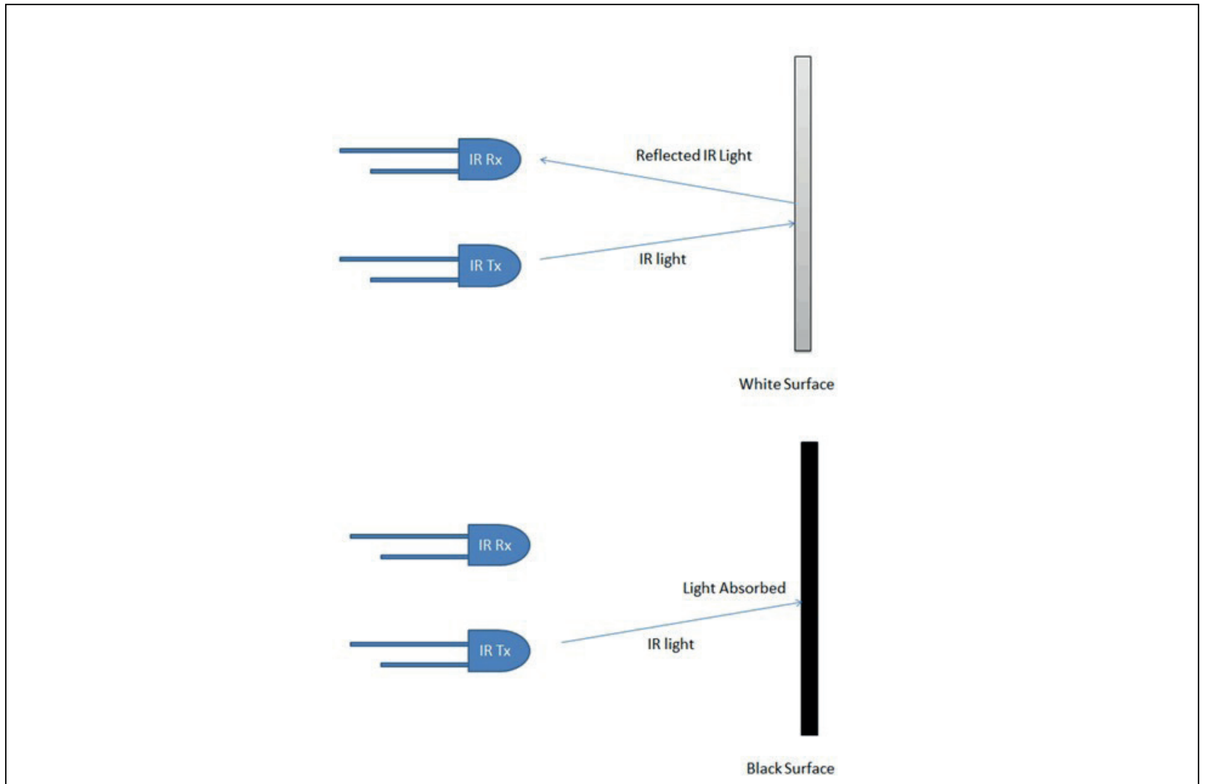




H - Bridge & Gleichstrommotoren

Der L298N ist ein dualer H-Bridge-Motortreiber, der die gleichzeitige Drehzahl- und Richtungssteuerung von zwei Gleichstrommotoren ermöglicht. Das Modul verfügt über zwei Schraubklemmenblöcke für die Motoren A und B und eine weitere Schraubklemmenleiste für den Erdungsstift, den VCC für den Motor und einen 5-V-Stift, der entweder Eingang oder Ausgang sein kann.
Die Pins Enable A und Enable B werden zum Aktivieren und Steuern der Motorgeschwindigkeit verwendet. Wenn an diesem Pin ein Jumper vorhanden ist, wird der Motor aktiviert und arbeitet mit maximaler Geschwindigkeit. Wenn wir den Jumper entfernen, können wir einen PWM-Eingang an diesen Pin anschließen und auf diese Weise die Geschwindigkeit des Motors steuern. Wenn wir diesen Pin mit einem Ground verbinden, wird der Motor deaktiviert. Die Pins Input 1 und Input 2 zur Steuerung der Drehrichtung des Motors A und die Eingänge 3 und 4 für Motor B verwendet. Wenn Eingang 1 LOW ist und Eingang 2 HIGH ist, bewegt sich der Motor vorwärts und umgekehrt, wenn Eingang 1 HIGH und Eingang 2 LOW ist, bewegt sich der Motor rückwärts. Wenn beide Eingänge gleich sind, stoppt der Motor entweder LOW oder HIGH. Gleiches gilt für die Eingänge 3 und 4 und den Motor B.



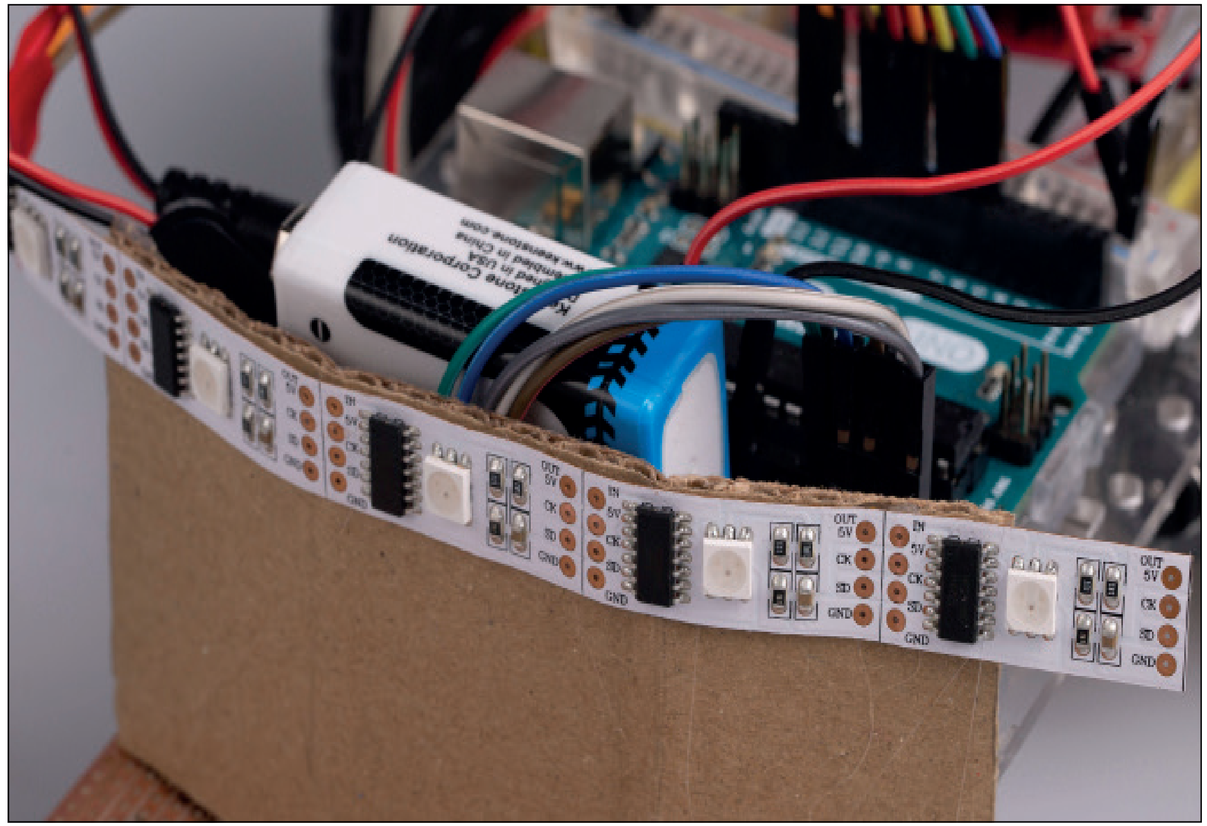


IR - Sensoren

Die Entfernungsmessung ist mit dem IR-Sensor sehr einfach. Ein IR-Strahl wird zum Ziel übertragen und der reflektierte Strahl wird von der Fotodiode erfasst. Die Fotodiode misst die Lichtintensität (hier meistens IR-Licht). Die IR-Diode misst nicht nur die Intensität von IR-Licht, sondern ist auch für sichtbares Licht empfindlich. Hier repräsentiert die Intensität des von der Photodiode aufgezeichneten IR-Lichts die Entfernung zwischen Ziel und Sensor.

Je näher das Ziel ist, desto höher wird die Intensität des Lichts, die mit zunehmender Entfernung abnimmt. Die IR-LED ist mit 100 Ohm des Widerstandes in Reihe zum digitalen Pin des Arduino geschaltet. und die Fotodiode ist mit 4,7 kOhm des Widerstands in Reihe zu 5 V der Versorgung verbunden, und das Signal wird daraus abgenommen, das an den analogen Eingangspin von Arduino geht.

Die Sensor Platine wird über die Sensor Pins ausgelesen, jeder Sensor ist somit mit dem Arduino Board verbunden. Wenn der Line/Grid Follower dann auf dem Grid fährt werden die Null und Eins Werte von den Sensoren über den Sensor Pin von dem Arduino Board ausgelesen. Je nach Grid und Programm weiß er dann ob er nur der Linie folgen muss oder ob er links/rechts abbiegen muss.



LED - Stripe

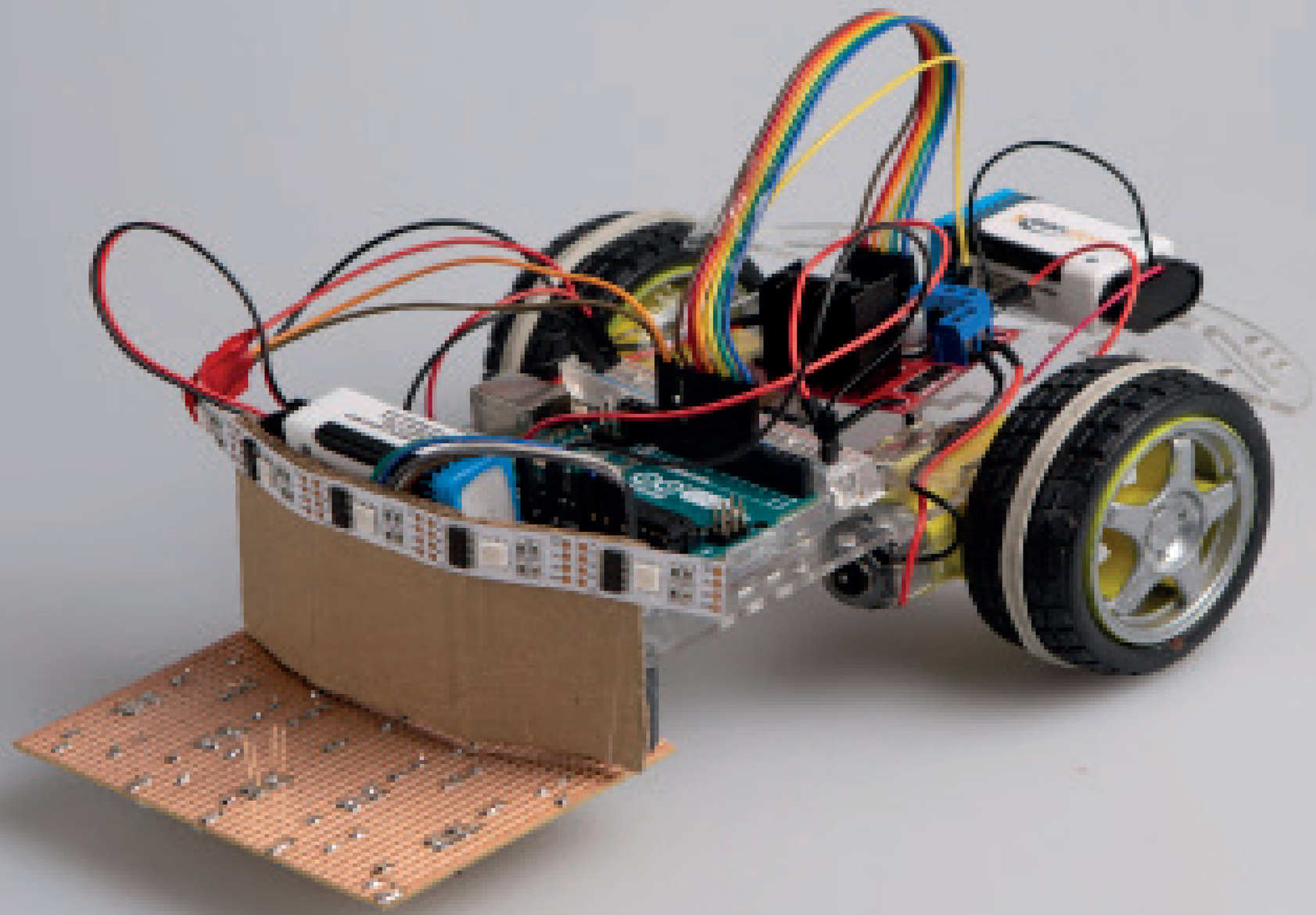
Der LED Stripe wird an der IN Seite angeschlossen, über den 5 Volt und den GND Pin wird der Stripe mit Spannung versorgt. Der dataPin gibt jedes Bit aus. Der clockPin wird zum Umschalten verwendet, sobald der data-Pin auf den korrekten Wert (int) gesetzt wurde. Mit der Bibliothek Adafruit wird der Stripe in den Code integriert.

Der LED Strip zeigt die Sensor Aktivität von den fünf Sensoren an, ob gerade ein Signal Null oder Eins anliegt. Wir haben es so programmiert das bei Schwarz die Sensoren den Wert Eins bekommen und somit auch der LED Stripe die Aktivität visualisiert.

Software

Dokumentation

Christian Kolb & Hartmut Idler



```ino
#include "Adafruit_WS2801.h"
#include "SPI.h"

int pin[5] = { A1, A2, A3, A4, A5};
int senWo[5] = {0, 0, 0, 0, 0};

//int counter = 0, countOn = 0, countState = 0;


uint8_t dataPin = 13;
uint8_t clockPin = 12;

//Motor

int geschwR;
int geschwL;

//motor rechts
int in1 = 6;
int in2 = 7;
int ena = 5;

//motor links
int in3 = 8;
int in4 = 9;
int enb = 10;
```

```ino
void setup() {
  Serial.begin (9600);
  for (int i = 0; i < 5; i++) {
    pinMode(pin[i], INPUT);
  }
  pinMode(dataPin);
  pinMode(clockPin);

  strip.begin();
  strip.show();

  //motor definieren
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  pinMode(ena, OUTPUT);
  pinMode(enb, OUTPUT);
}
```

```ino
void loop() {
  for (int i = 0; i < 5; i++) {
    senWo[i] = digitalRead(pin[i]);
    Serial.print("SW:\t");
    Serial.print(senWo[i]);
    Serial.print(" ");
    if (senWo[i] == 1) {
      strip.setPixelColor(i, 10 , 0, 0);
    } else {
      strip.setPixelColor(i, 0, 0, 0);
    }
    strip.show();

    //motor an
    //motor rechts
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
    analogWrite(ena, geschwR);

    //motor links
    digitalWrite(in3, HIGH);
    digitalWrite(in4, LOW);
    analogWrite(enb, geschwL);


    lineFollow();
```

Einbindung der Hardware

Die Einbindung der Hardware ist in drei Schritte unterteilt. Im ersten Schritt auserhalb einer Methode werden die Pins auf dem Arduino Bord zur jeweiligen Komponente definiert. In der Setup Methode werden als erstes die Sensoren mit Serial.begin gestartet. Mit den beiden Strip befehlen wird der LED-Stripe angesteuert. Am Schluss werden die H-Bridge Pins auf Output gestellt.

```
87 void lineFollow() {
88   if ((senNo[0] == 1) && (senNo[1] == 0) && (senNo[2] == 0) && (senNo[3] == 0) && (senNo[4] == 0)) //rechts
89   {
90     geschwL = 32;
91     geschwR = 92;
92   }
93   if ((senNo[0] == 0) && (senNo[1] == 1) && (senNo[2] == 0) && (senNo[3] == 0) && (senNo[4] == 0)) //rechts
94   {
95     geschwL = 75;
96     geschwR = 105;
97   }
98   if ((senNo[0] == 0) && (senNo[1] == 0) && (senNo[2] == 1) && (senNo[3] == 0) && (senNo[4] == 0)) //rechts
99   {
100    geschwL = 102;
101    geschwR = 105;
102  }
103  if ((senNo[0] == 0) && (senNo[1] == 0) && (senNo[2] == 0) && (senNo[3] == 1) && (senNo[4] == 0)) //rechts
104  {
105    geschwL = 105;
106    geschwR = 75;
107  }
108  if ((senNo[0] == 0) && (senNo[1] == 0) && (senNo[2] == 0) && (senNo[3] == 0) && (senNo[4] == 1)) //rechts
109  {
110    geschwL = 92;
111    geschwR = 32;
112  }
113
114 }
```

Manueller Linefollower

Nach dem die Hardware komponenten initialisiert wurden geht es an die Methode die den Follower fahren lässt. Die Methode wird im Loop aufgerufen. Es werden der Reihe nach die Sensoren von rechts nach Links abgefragt ob sie ein Signal haben. Wenn der ganz Rechte Sensor einen wert hat soll die Geschwindigkeit nach Rechts gehen. Deshalb ist sie höher. Bei den äussersten Sensoren soll so schnell wie möglich der Weg zurück auf die Linie gefunden werden deshalb bekommt nur eine Geschwindigkeit einen hohen Wert. Der mittlere Sensor soll so gerade wie möglich fahren.

Berechneter Linefollower

Mit der Pid_value wird der Optimale Wert berechnet um den Follower auf der Linie zuhalten. Die Sensorenwerte bekommen unterschiedliche Werte zugewiesen die dann in der Rechnung verechnet werden. Es wird auch der letzte Sensorwert in die Rechnnung einbezogen.Die Rechten Sensoren bekommen negative Werte und die Linken Sensoren positive Werte. Dadurch entsteht der Unterschied zwischen links und recht abbiegen. Diese Methode ist sehr interresant und funktioniert auch gut. Am ende wird bei analog Write der einen Geschwindigkeit der Pid_ value Wert addiert und der anderen abgezogen.

```
void linefollow() {


  counterone = 0;

  if ((sensVal[0] == 1) && (sensVal[1] == 0) && (sensVal[2] == 0) && (sensVal[3] == 0) && (sensVal[4] == 0))
  {
    sensorCase = -60;
    counterone = 0;
  }

  if ((sensVal[0] == 1) && (sensVal[1] == 1) && (sensVal[2] == 0) && (sensVal[3] == 0) && (sensVal[4] == 0))
  {
    sensorCase = -30;
    counterone = 0;
  }

  if ((sensVal[0] == 0) && (sensVal[1] == 1) && (sensVal[2] == 0) && (sensVal[3] == 0) && (sensVal[4] == 0))
  {
    sensorCase = -10;
    counterone = 0;
  }

 if ((sensVal[0] == 0) && (sensVal[1] == 0) && (sensVal[2] == 1) && (sensVal[3] == 0) && (sensVal[4] == 0))
  {
    sensorCase = -5;
    counterone = 0;
  }

  if ((sensVal[0] == 0) && (sensVal[1] == 0) && (sensVal[2] == 1) && (sensVal[3] == 1) && (sensVal[4] == 0))
  {
    sensorCase = 0;
    counterone = 0;
  }

 if ((sensVal[0] == 0) && (sensVal[1] == 0) && (sensVal[2] == 1) && (sensVal[3] == 1) && (sensVal[4] == 0))
  {
    sensorCase = 5;
    counterone = 0;
  }

  if ((sensVal[0] == 0) && (sensVal[1] == 0) && (sensVal[2] == 0) && (sensVal[3] == 1) && (sensVal[4] == 0))
  {
    sensorCase = 10;
    counterone = 0;
  }

if ((sensVal[0] == 0) && (sensVal[1] == 0) && (sensVal[2] == 0) && (sensVal[3] == 1) && (sensVal[4] == 1))
  {
    sensorCase = 30;
    counterone = 0;
  }

  if ((sensVal[0] == 0) && (sensVal[1] == 0) && (sensVal[2] == 0) && (sensVal[3] == 0) && (sensVal[4] == 1))
  {
    sensorCase = 60;
    counterone = 0;
  }

  P = sensorCase;
  I = I + sensorCase;  // I = I + sensorCase;
  D = sensorCase - prevSensorCase;
  pid_value = (Kp * P) + (Ki * I) + (Kd * D);

  prevSensorCase = sensorCase;
  geschwL = 90;
  geschwR = 90;

  analogWrite (ena, (geschwL - pid_value));
  analogWrite (enb, (geschwR + pid_value));


```

```
void gridCount() {

  //Zählen
  if ((senWo[1] == 1) && (senWo[2] == 1) && (senWo[3] == 1) )
  {
    counterCase = 1;
  }

  if (counterCase == 1 && counterState == 0)
  {
    zlr++;
    counterState = 1;
    Serial.println(zlr);

  }
  else if (counterState == 1 && counterCase == 0)
  {
    counterState = 0;
  }


  if ((senWo[0] == 1) && (senWo[1] == 1) && (senWo[2] == 1) )
  {
    counterCase = 1;
  }

  if (counterCase == 1 && counterState == 0)
  {
    zlr++;
    counterState = 1;
    Serial.println(zlr);
  }
  else if (counterState == 1 && counterCase == 0)
  {
    counterState = 0;
  }

  if ((senWo[1] == 1) && (senWo[3] == 1) && (senWo[4] == 1)) {
    counterCase = 1;
  }

  if (counterCase == 1 && counterState == 0)
  {
    zlr++;
    counterState = 1;
    Serial.println(zlr);
  }
  else if (counterState == 1 && counterCase == 0)
  {
    counterState = 0;
  }
  if (zlr == 5) {
    counterCase = 0;
    counterState = 0;
    zlr = 0;
    directionState = 1;
  }

}
```

```
void motorRight() {
  //motor an
  //motor rechts
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  analogWrite(enA, geschwL = 92);

  //motor links
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  analogWrite(enB, geschwR = -30);



  if ((senWo[0] == 0)&&(senWo[1] == 1)&&(senWo[2] == 0))
  {

    turnState = 1;
  }



  if ((senWo[0] == 0) && (senWo[1] == 0) && (senWo[2] == 0) && (senWo[3] == 0) && (senWo[4] == 0))
  {

    stopState = 1;
  }


  if (turnState == 1 && stopState == 1 && (senWo[2] == 1)) {

    zlr = 0;
    directionState = 2;

  }
}
```

Gridfollower Runde

Für die Runde um den Grid werden zwei Methoden zu der gerade aus Methode gebraucht. Wir haben einen directionState eingerichtet der in der im loop stetig abgefragt wird. 0 bedeutet gerade aus, 1 bedeutet rechts abbiegen.

Am Anfang fährt er normal gerade aus und ruft in der Linefollow Methode die gridCount-Methode auf. In der gridCount Methode wird verglichen wie viele Sensoren auf der Linie sind, wenn 3 Sensoren gleichzeitig auf der Linie sind wird der Zähler um eins erhöht. Am Ende mit der If Abfrage wird überprüft ob der Zähler 5 ist. Wenn der Zähler 5 ist wird der directionState auf 1 erhöht.

Bei directionState 1 wird dann die Motor Right Methode aufgerufen. Hier gibt es den turnState und Stop State. Der turnState wird auf 1 gesetzt wenn einzelne Sensoren auf der Linie sind und der Follower biegt ab. Wenn einmal alle 5 Sensoren auf Weiß sind wird der StopState auf 1 gesetzt. Wenn der stopState und turnState auf 1 und einer der Mittleren Sensoren auf der Linie ist wird der DirectionState wieder auf 2 gestellt und stopt für 20ms. So fährt der Follower ständig eine Runde.

```
void gridCount2() {

  ///daten
  if ((senWo[1] == 1) && (senWo[2] == 1) && (senWo[3] == 1) )
  {
    counterCase = 1;
  }

  if (counterCase == 1 && counterState == 0)
  {
    zlr++;
    counterState = 1;
    Serial.println(zlr);

  }
  else if (counterState == 1 && counterCase == 0)
  {
    counterState = 0;
  }


  if ((senWo[0] == 1) && (senWo[1] == 1) && (senWo[2] == 1) )
  {
    counterCase = 1;
  }

  if (counterCase == 1 && counterState == 0)
  {
    zlr++;
    counterState = 1;
    Serial.println(zlr);
  }
  else if (counterState == 1 && counterCase == 0)
  {
    counterState = 0;
  }

  if ((senWo[2] == 1) && (senWo[3] == 1) && (senWo[4] == 1)) {
    counterCase = 1;
  }

  if (counterCase == 1 && counterState == 0)
  {
    zlr++;
    counterState = 1;
    Serial.println(zlr);
  }
  else if (counterState == 1 && counterCase == 0)
  {
    counterState = 0;
  }
  if (zlr == 3) {
    counterCase = 0;
    counterState = 0;
    zlr = 0;
    directionState = 3;
  }

};

```

```
void motorStop()
{
  //motor an
  //motor rechts
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);


  //motor links
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  delay (500);

}
```

Gridfollower Koordinate

Von der Runde bis zu einer bestimmten Koordinate ist der Weg nicht mehr weit. Wir haben 2 weitere directionStates eingefügt. directionState 2 ist nochmal gerade ausfahren. DirectionState 3 bedeutet motor Stop. Die States werden wieder im Loop abgefragt.

Der Linefollower macht die erste Abbiegung identisch wie bei der Runde. Nach dem Abbiegen wird der State dann auf 2 gesetzt. LineFollow2 und gridCount 2 sind identisch wie die anderen Methoden mit dem Unterschied das der Zähler Wert geringer sein kann und dadurch ein andere Punkt angesteurt werden kann.

Wenn der gridFollower dann bsp. bei 4/3 ist und nach dem Abbiegen das 2te mal der Linie gefolgt ist wird der directionState auf 3 gesetzt wenn der Zähler Wert erreicht ist. Das bedeutet motorStop, der gridFollower Stopt an dem gewünschten Punkt.