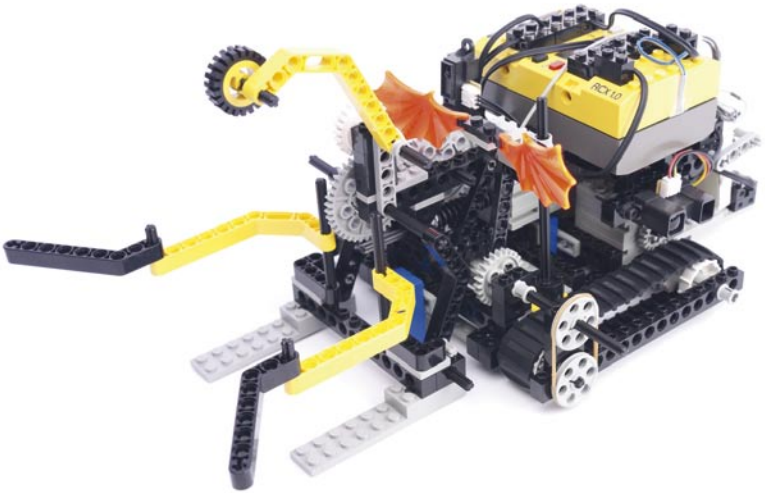
Lego-Roboter „Limier“

Dipl. Ing. Jens Weber, MediaArchitecture
Dipl. Des. Andreas Wolter, MediaArchitecture

Ein Einstieg in die Robotik: „...und er bewegt sich doch!“

Jun.-Prof. Frank Petzold
Dipl. Ing. Jan Sieber



Fakultät Architektur und Fakultät Medien
Bauhaus-Universität Weimar / SS 2007

Seit Karl dem Großen wurden Leit- und Spürhunde eingesetzt und in ihrer reinen Form bis ins 19. Jahrhundert hinein gezüchtet. Der Leithund (auch als „**Limier**“ bezeichnet) galt als die vornehmste Jagdhundeart und diente, je nach Abrichtung, zur „Vorsuche“ auf Edel-, Dam- oder Schwarzwild.
Ursprünglich stellte er nur eine besondere Dressurform des Laufhundes dar, bei der ein zur Riemenarbeit geeigneter Hund mit ausgesprochen feiner Nase verwendet wurde. […] Stets am Leitseil arbeitend, suchten sie mit tiefer Nase, gaben niemals Laut und blieben auf der einmal gefundenen „kalten Fährte“ (ältere Wildfährte), ohne sich durch eine andere ablenken zu lassen. Sie besaßen eine ausgezeichnete Spürnase und galten als ruhig, folgsam und ausgeglichen.

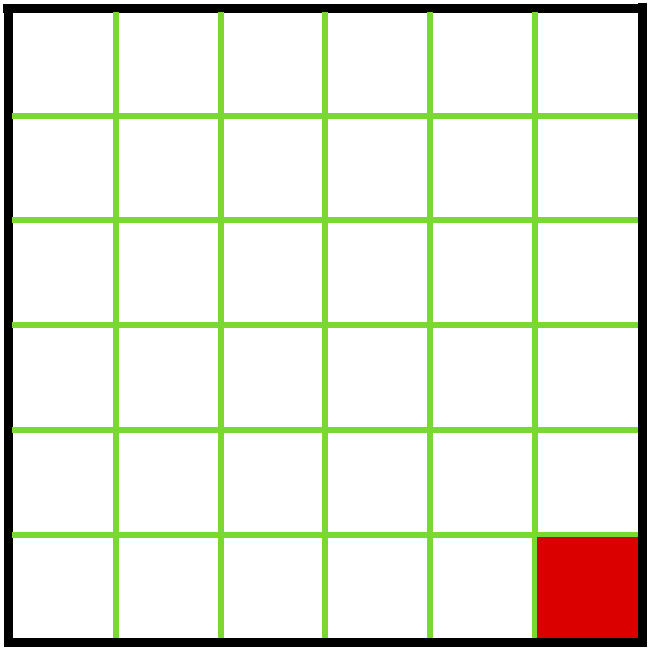
Quelle: Wikipedia

1. Zielstellung / Aufgabe

Bauen Sie einen Roboter, der eine leere Dose finden, aufnehmen und an den Ausgangspunkt zurückbringen kann. Startpunkt ist das rote Feld auf dem in Abbildung 1 dargestellten Spielfeld. Zu Beginn ist der Roboter parallel zu einer gewünschten Seite des Quadrates ausgerichtet. In einem Wettkampf am Ende des Semesters wird der schnellste Roboter gekürt.

Es steht Ihnen Lego Mindstorms mit dem RCX 1.0 zur Verfügung.

1. Zielstellung / Aufgabe



[Abb. 1] Spielfeld, Seitenlänge: 2,20m

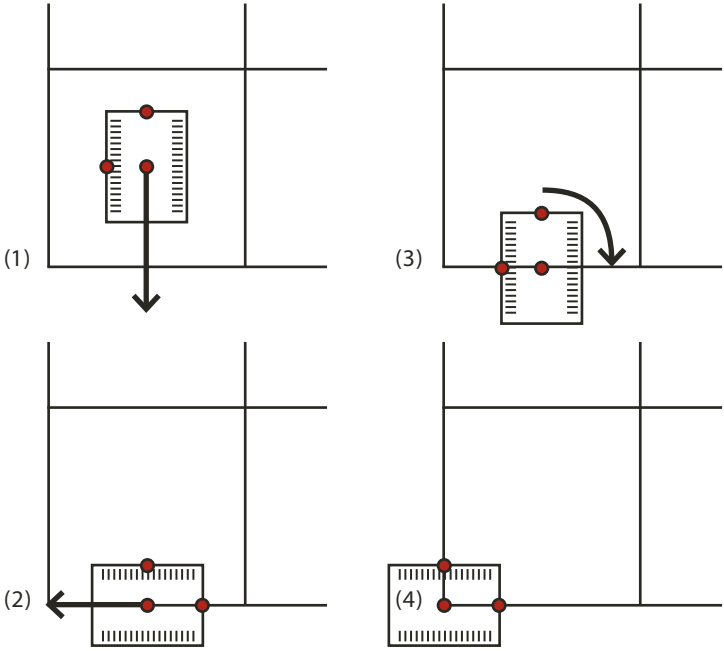
2. Vorüberlegungen

Um die gestellte Aufgabe ideal zu erfüllen, muss der Roboter die folgenden Fähigkeiten besitzen:

– Erkennen der eigenen Ausgangsstellung im Raum
– Merken / Rekonstruieren des Weges bzw. der Dosenposition und der Ausgangsposition
– Erkennen der Dose
– Aufnehmen der Dose
– Transport der Dose
– Erkennen des Ziels

Natürlich sind Lösungen denkbar, die nach irgendeinem Prinzip – Auswerfen von Netzen, Schieben o. ä. – das gesamte Feld in Richtung Ausgangspunkt „leerfegen".
Auch das spezielle Markieren der Dose oder des Ausgangspunkts sind möglich.
Hier soll aber versucht werden, die mit der Aufgabenstellung intendierte Lösung zu verwirklichen.

Die Orientierung im Raum erfolgt am einfachsten über das aufgezeichnete Raster. Andere Lösungen müssten auf eine Dreieckspeilung hinauslaufen (z. B. über Ultraschall), die zusätzliche Sender und Sensoren benötigt.
Durch geeignete Fahrmanöver und Sensoren kann so auch die Ausgangsposition festgestellt werden (s. Abb. 2.1 ff.).



[Abb. 2.1 bis 2.4] Finden der Ausgangsposition mit 3 Sensoren

2. Vorüberlegungen

Für das Auffinden der Dose gibt es mehrere Möglichkeiten:

1. Abfahren aller Quadrate und registrieren des Zusammenpralls mit der Dose über einen Sensor
2. Abfahren aller Quadrate und Erkennen der Dose vor dem Zusammenprall mithilfe der IR-Sensoren des Bricks
3. Abtasten des gesamten Feldes über geeignete Distanzmesser. Hierbei darf nicht über den Spielfeldrand gemessen werden, um keine anderen Gegenstände mit der Dose zu verwechseln.
4. Erkennen der Dose nach dem Prinzip eines Metallsuchgeräts – ein wechselndes elektromagnetisches Feld wird von der Dose als „störendes" Dielektrikum verändert.

Die Lösungen 1 und 2 unterfordern der Mikroprozessor, sind nicht schnell genug und unelegant.
Die übrigen Vorschläge werfen hingegen ein weiteres Problem auf: Die Anzahl der Sensoren und Aktuatoren ist auf jeweils 3 beschränkt.

Die Motoren verteilen sich folgendermaßen:
– Motor 1 und 2 zur freien Bewegung
– Motor 3 zum Anheben der Dose

2. Vorüberlegungen

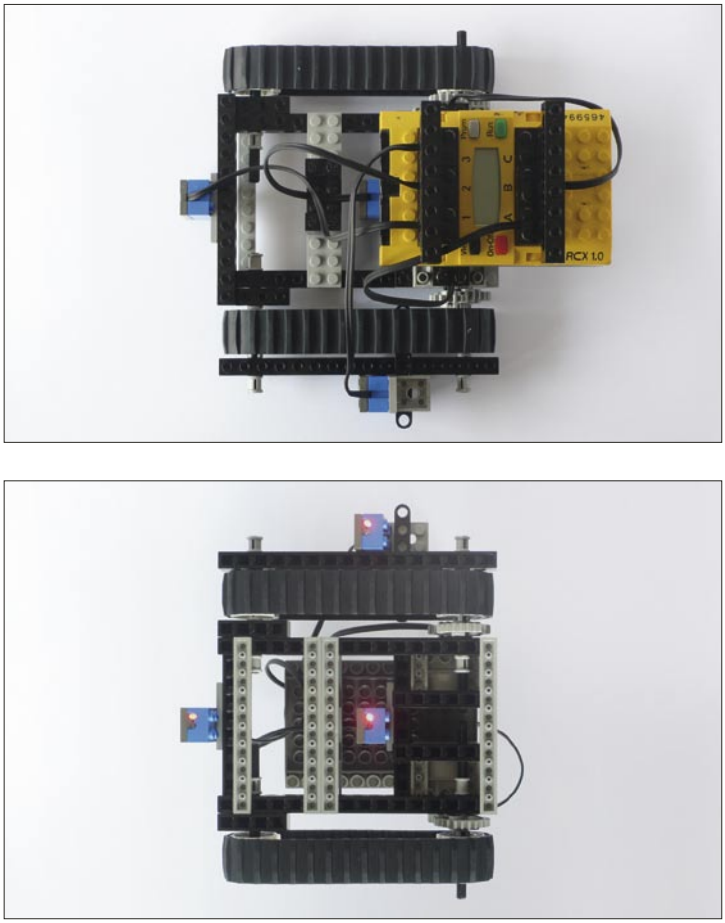
Zum optimalen Verfolgen der Linie sind zwei Sensoren nötig: jeweils einer zum Feststellen einer Abweichung nach links und rechts. Um das Ende der Geraden zu finden, wird ein weiterer Sensor benötigt. Idealerweise hilft ein vierter Sensor, eine exakte 90°-Drehung zu realisieren.

Unter zwei Sensoren ist ein Um-die-Ecke-fahren nicht sicher möglich. Mindestens ein weiterer Sensor wird für das Erkennen der Dose gebraucht.

Motor 3 könnte auch direkt vom Dose-Kollisions-Sensor über eigene Elektronik angesteuert werden und benötigt nicht zwingend den Brick.
Zur Vervielfältigung der Sensoreingänge wird in der Literatur meist ein Multiplexverfahren vorgeschlagen z.B. bei:

Gasperi, Michael
http://www.plazaearth.com/usr/gasperi/lego.htm

Cordes, Dietmar und Lemm, Gunther:
„Konzeption von I/O-Erweiterungen für Lego Mindstorms“
http://users.informatik.haw-hamburg.de/~kvl/lepomux/lepomux.pdf

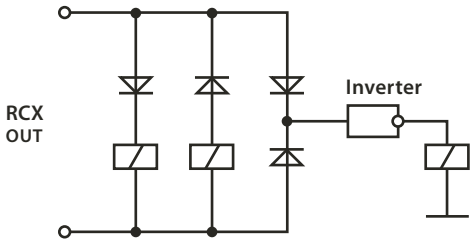


[Abb. 3] Vormodell mit idealer Sensorverteilung

Diesen Vorschlägen möchte ich zwei weitere hinzufügen:

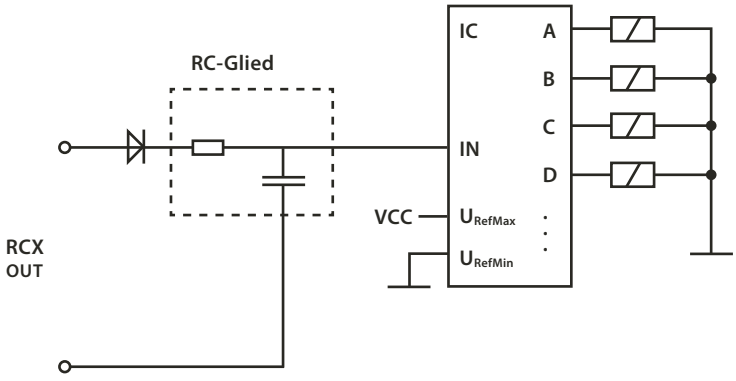
Zunächst schauen wir dazu auf die Möglichkeiten des Ausgangs. Er hat elektronisch gesehen vier Zustände:
1. Vorwärts: PWM mit 1kHz in 7 Stufen
2. Rückwärts: PWM mit 1kHz in 7 Stufen
3. Gestoppt: kurzgeschlossen
4. Aus (Float): hochohmig

Der erste Vorschlag nutzt diese Zustände, um elektronisch zwischen beiden Polaritäten zu unterscheiden und als dritten Zustand das Nichtvorhandensein einer Spannung zu nehmen.



[Abb. 4] Prinzipschaltbild Vorschlag 1

Der zweite Vorschlag in Abb. 5 nutzt die unterschiedlichen Geschwindigkeiten, also die PWM in einer Richtung.

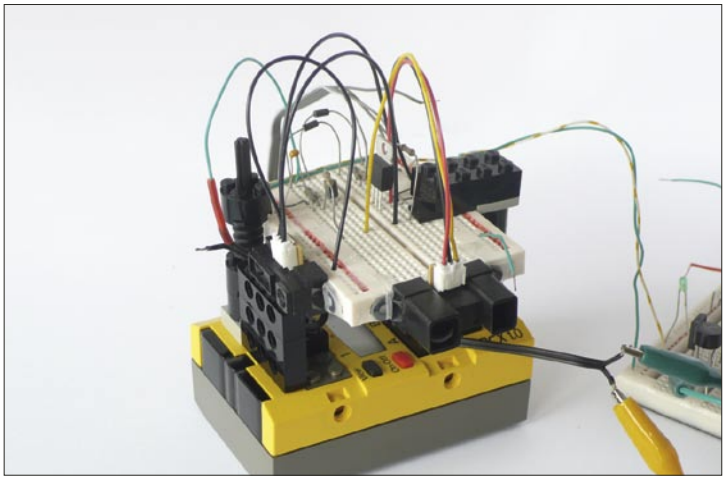


[Abb. 5] Prinzipschaltbild Vorschlag 2

Die pulsweitenmodulierte Spannung am Motorausgang wird durch das RC-Glied in Spannungswerte zwischen 0 und der Betriebsspannung gewandelt.
Der nachgeschaltete IC realisiert eine Punktanzeige, in Abhängigkeit von der Eingangsspannung. Mit den Ausgängen werden Relais geschaltet.
So kann jeder Motor-Leistungsstufe ein Relais zugordnet werden.

Für das eigentliche Aufnehmen der Dose wurden diese Varianten geprüft:

1. Prinzip der Mausefalle
2. Magnete (leider wurden keine Getränkedosen aus Weißblech gefunden, sondern nur Aluminium)
3. Klebepunkte
4. Greifer mit seitlichen Backen
5. Greifer von unten (nutzen die Verjüngung am Boden der Dose)



[Abb. 6] Aufbau zum Test der beiden Sensoren am Brick

3. Konzept

Der folgende Algorithmus wurde realisiert:

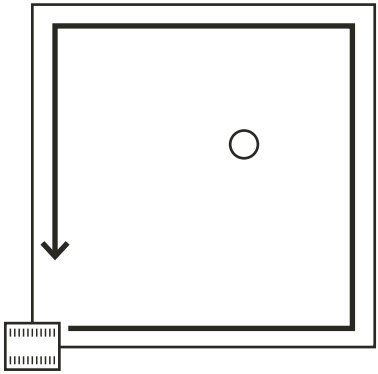
Zunächst fährt der Roboter in Startposition auf die schwarze Linie in die Ecke des Startfeldes. Von dort folgt er der Linie über alle vier Seiten des Spielfeldes **(1)** bis er zum Ausgangspunkt zurückkehrt.

Während dieser Fahrt benutzt er einen IR-Distanzsensor um Objekte im Inneren des Spielfeldes zu detektieren. Dieser Sensor reicht nur bis ca. 2/3 der Spielfeldtiefe, um nicht durch fremde Objekte beeinflusst zu werden.

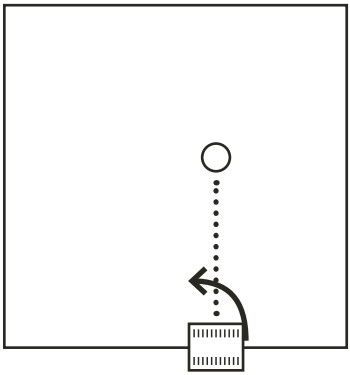
Wird ein Objekt gefunden, schwenkt der Roboter darauf zu **(2)** und benutzt einen zweiten, nach vorne gerichteten Sensor, um es anzupeilen **(3)**.

Das Aufnehmen der Dose startet, wenn sie nahe genug vor dem Sensor (und damit im Greifer) ist.
Je nach Position wählt der Roboter den kürzesten Weg zum Ausgangspunkt zurück und stoppt dort.

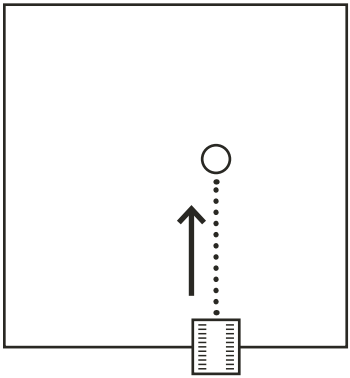
(1)



(2)

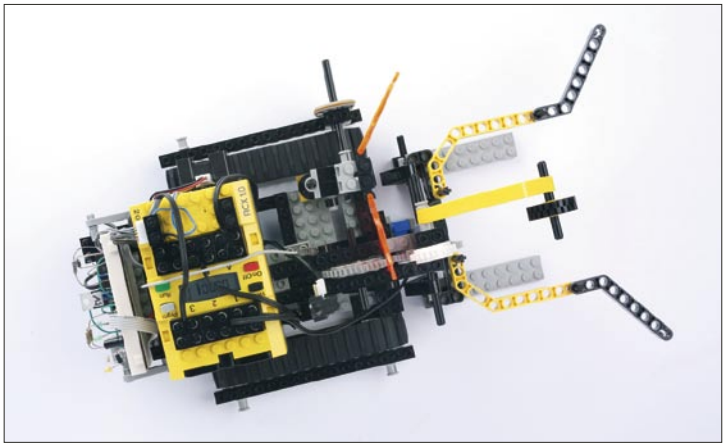


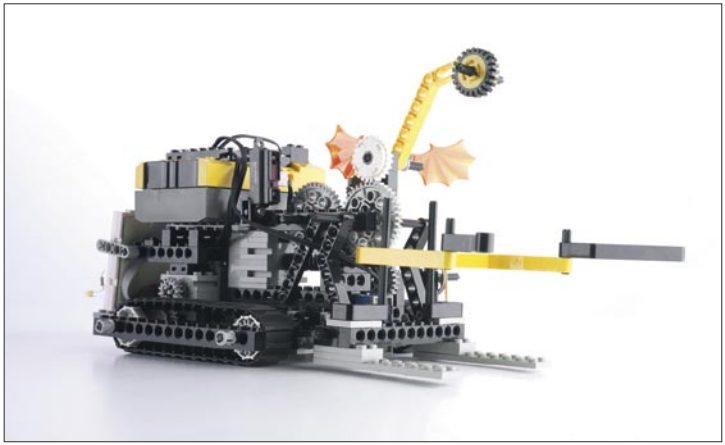
(3)



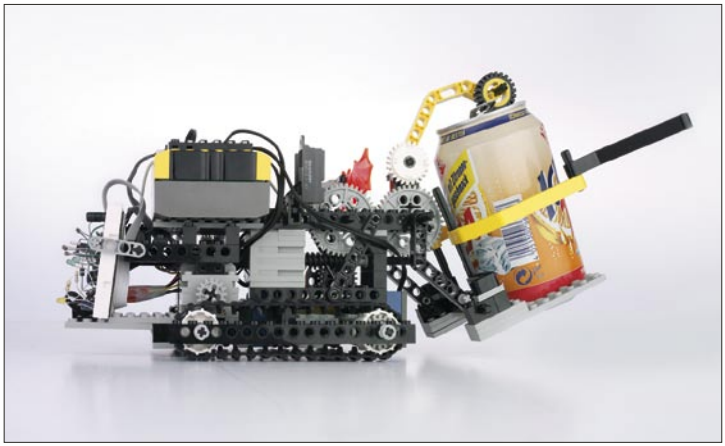
[Abb. 7.1 bis 7.3] Spielfeld

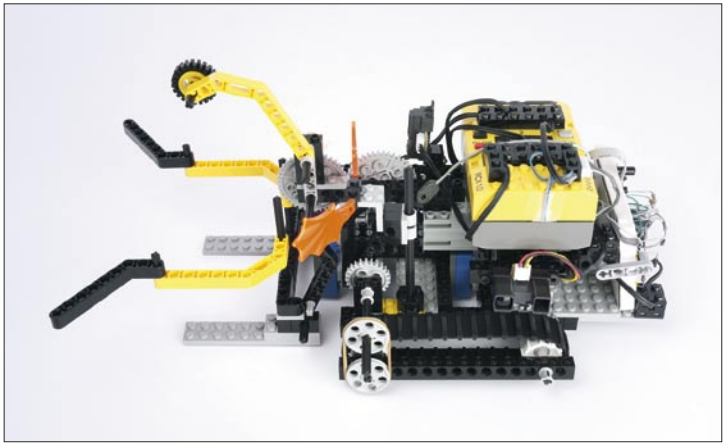
4. Umsetzung





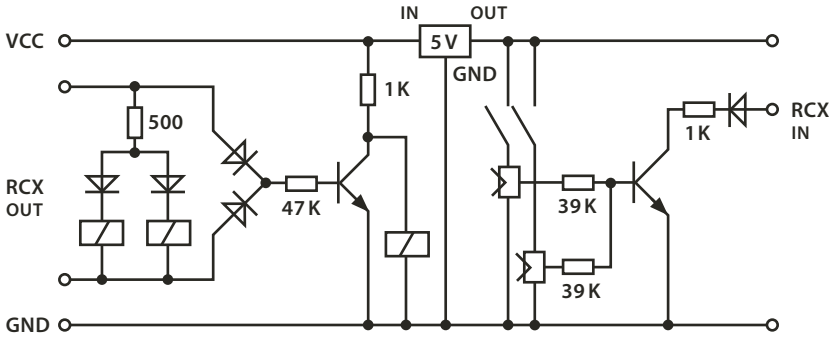
4. Umsetzung

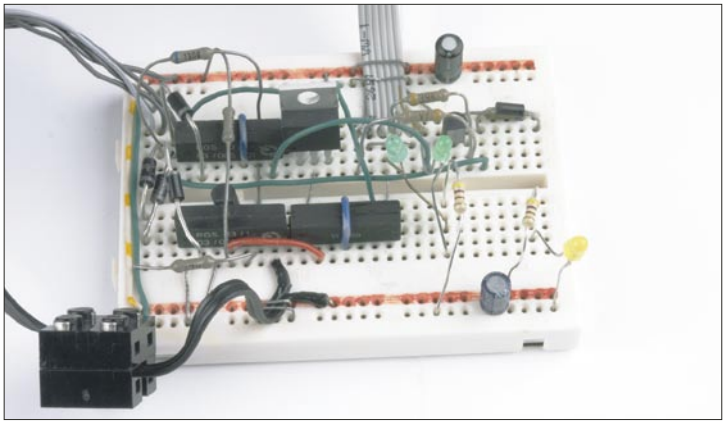




[Abb. 8 bis 11] Konstruktion des Roboters aus Legoteilen

Der folgende Schaltplan zeigt die zusätzliche Elektronik, die benötigt wurde, um über den Ausgang C den Distanzsensor auszuwählen oder die Dose anzuheben. Der NQC-Code befindet sich im Anhang.





[Abb. 12] Aufbau der Schaltung auf einem Breadboard

5. Fazit

Prototypische Lösungen mit Lego Mindstorms zu entwickeln macht großen Spaß und geht verhältnismäßig schnell. Wie gezeigt wurde, führt die beschriebene Lösung in den meisten Fällen zum Erfolg.

Der größte Nachteil liegt in der fehlenden Fähigkeit einer genauen 90-Grad-Drehung. Dazu wäre ein weiterer Sensor nötig gewesen.
Nachteilig war auch die sehr starke Abhängigkeit von der Batterieleistung. Das Fahrverhalten verändert sich gerade in Belastungssituationen, wie den Drehungen, massiv.

6. Anhang

```
int sensor_rot, sensor_sw, sensor_weiss, finished;
int timer90grd, num_edges=0;
int mode=0;
int turnmode,findtimer,turntime,toggle;

// rot = 49
// sw = 33
// gr = 41
// weiss = 48



task main()
{
  SetSensor (SENSOR_1, SENSOR_LIGHT); // vorn
  SetSensor (SENSOR_2, SENSOR_LIGHT); // Mitte

  SetSensorType (SENSOR_3, SENSOR_TYPE_TOUCH); // ent-
fernungssensoren
  SetSensorMode (SENSOR_3, SENSOR_MODE_RAW);

  SetUserDisplay (mode, 0);

  // Auf Ausgangsposition
  // 1+2 auf sw-Linie, vom roten Quadrat wegschauend
  find_startposition();

  start follow_line;
  start observe_left;
}


// Warten zum analysieren f¸r die langsamen Menschen
void pause()
{
  PlaySound (SOUND_CLICK);
  Off(OUT_A+OUT_B);
  Wait(100);
}
```

```
// zurücksetzen, bis mitte auf sw
// 90grd rechts bis mitte und vorn auf sw
sub find_startposition()
{
  sensor_rot = SENSOR_1;
  OnRev(OUT_A+OUT_B);
  until (SENSOR_2 < sensor_rot -12);
  // schwarze Linie gefunden - jetzt drehen um 90grd
  sensor_sw = SENSOR_2;
  pause();

  // drehen 90grd
  OnFwd(OUT_A+OUT_B);
  Wait(20);
  ClearTimer(0);
  OnFwd(OUT_A);
  OnRev(OUT_B);
  until (SENSOR_1 < sensor_sw +8);
  timer90grd = FastTimer(0);
  Wait(5); // etwas weiter in die Linie reindrehen
  Off(OUT_A+OUT_B);

  mode = 1;
  PlaySound (SOUND_CLICK);

}

task follow_line()
{
  while (num_edges < 4)
  {
       if (SENSOR_1 < sensor_sw +5) // es reicht nur den
vorderen Sensor zu beobachten
       {
         OnFwd (OUT_A+OUT_B);
       }
       else
       {
         // korrigieren nach links
         ClearTimer(0);
         OnFwd(OUT_B);
         Off(OUT_A);
         until (FastTimer(0) > 25 || SENSOR_1 < sen-
sor_sw +5);
         if (SENSOR_1 < sensor_sw +5) Wait(5);
         Off(OUT_A+OUT_B);
         pause();
```

```
         // vielleicht auch rechts, aber erstmal wieder
zurück (doppelte zeit)
         if (SENSOR_1 > sensor_sw +5)
         {
           ClearTimer(0);
           OnFwd(OUT_A);
           Off(OUT_B);
           until (FastTimer(0) > 40 || SENSOR_1 < sen-
sor_sw +5);

           if (SENSOR_1 < sensor_sw +5) Wait(5);
           Off(OUT_A+OUT_B);
           pause();

           // Linienende
           if (SENSOR_1 > sensor_sw +5)
           {
             // dann ist die Linie wohl zuende - links-
rum 90grd
             PlaySound (SOUND_LOW_BEEP);

             // suche zur¸ckdrehen
             ClearTimer(0);
             OnFwd(OUT_B);
             OnRev(OUT_A);
             until (FastTimer(0) > 12);

             OnFwd (OUT_A+OUT_B);
             until (SENSOR_3 < sensor_sw +10);
             Wait(15); // etwas weiterlaufen

             OnFwd(OUT_B);
             OnRev(OUT_A);
             until (SENSOR_1 < sensor_sw +5);
             Off(OUT_A+OUT_B);

             num_edges = num_edges + 1;

             pause();
           }

         }

       }
  }
}
```

6. Anhang

```
// nach links schauen und auf die Dose warten
task observe_left()
{
   Wait(10);
   while (mode == 1)
   {
      if (SENSOR_3 < 300 && Timer(3) < 2)    // 1000
      {
         // Dose links gesehen
         mode = 2;
         stop follow_line;
         start go2can;
         start finetuning;
      }
      if (SENSOR_3 < 300)                    // 1000
      {
         PlaySound (SOUND_CLICK);
         ClearTimer(3);
      }
   }
}

// von der sw Line aus einschwenken
// konstant anpeilen der Dose
// Dose kann auch noch unsichtbar sein für den Front-
Sensor
task go2can()
{

    // 90grd einschwenken
    ClearTimer(0);
    OnFwd(OUT_A);
    OnRev(OUT_A);
    until (FastTimer(0) > timer90grd);

    // geradeaus
    OnFwd(OUT_A+OUT_B);
    Wait(50);

    while (true)
    {
       locate_can();
```

```
    }

}

// wenn Dose nicht vor der Nase
// links und rechts suchen
// wenn nicht findbar geradeaus weiter
// sonst darauf zu
sub locate_can()
{
     if (SENSOR_3 > 1000)
     {
       // Dose nicht vor der Nase
       turnmode = -1;
       findtimer = -2;
       ClearTimer(3);

   while (turnmode < 2 && findtimer < 0)
   {
      if (turnmode == -1 && FastTimer(3) > 100) {turnmo-
de = 1;ClearTimer(3);}  // erst links
      else if (turnmode == 1 && FastTimer(3) > 200)
turnmode = 2;  // dann rechts

      if (findtimer == -2 && SENSOR_3 < 900)
      {
         // Anfang Objekt
         findtimer = -1;
         ClearTimer(1);
      }
      else if (findtimer == -1 && SENSOR_3 > 1000)
      {
         // Ende Objekt
         findtimer = FastTimer(1);
         PlaySound (SOUND_DOUBLE_BEEP);
      }

      if (toggle%2 == 0)
      {
         Off(OUT_A+OUT_B);
         Wait(2);
      }
      else
      {
         if (turnmode == -1)
         {
            OnFwd(OUT_B);
            OnRev(OUT_A);
         }
```

```
         else
         {
            OnFwd(OUT_A);
            OnRev(OUT_B);
         }
       }
       toggle=toggle+1;
   }

   if (findtimer > 0) turntime = findtimer/2; // Objekt-
mitte
   else turntime = 100; // 90grd

   // zurueckdrehen
   ClearTimer(1);
   while (FastTimer(1) < findtimer/2)
   {
          if (toggle%2 == 0)
          {
            Off(OUT_A+OUT_B);
            Wait(2);
          }
          else
          {
            OnFwd(OUT_A);
            OnRev(OUT_B);
          }
          toggle=toggle+1;
   }
   Off(OUT_A+OUT_B);
     }
     else
     {
       // Dose steht vor uns
       // geradeaus
       OnFwd(OUT_A+OUT_B);
       Wait(100);
     }
}

// Erkennen, wenn Dose direkt vor uns
// neu anpeilen und draufzu
task finetuning()
{

    // vorderen Sensor aktivieren
    OnFwd(OUT_C);
```

```
    while (mode == 2)
    {
       if (SENSOR_3 < 230)
       {
          mode = 3;
          PlaySound (SOUND_UP);

          stop go2can;
          start pickup_can;

          locate_can();
       }
    }

}

// nimmt die Dose auf, wenn nah genug
task pickup_can()
{

   while (mode == 3)
   {
     if (SENSOR_3 < 208 && Timer(0) < 2)
     {
         mode == 4;
         Wait(20);
         Off(OUT_A+OUT_B);
         OnRev(OUT_C); // Anheben
         Wait(150);
         OnFwd(OUT_C);

         start go_home;

     }
     if (SENSOR_3 < 208)
     {
       PlaySound (SOUND_CLICK);
       ClearTimer(0);
     }
   }
}
```

```
task go_home()
{
   if (num_edges == 0)
   {
      turn_left();
      until_line();
      find_line_left();
      start follow_line;
   }
   else
   if (num_edges == 1)
   {
      until_line();
      find_line_left();
      start follow_line;
   }
   else
   if (num_edges == 2)
   {
      until_line();
      find_line_right();
      start follow_line;
   }
   else
   if (num_edges == 3)
   {
      turn_right();
      until_line();
      find_line_right();
      start follow_line;
   }
}

sub until_line()
{
   OnFwd(OUT_A+OUT_B);
   until (SENSOR_2 < sensor_sw +5);
}

sub turn_left()
{
   // links 90
   ClearTimer(0);
   OnFwd(OUT_B);
   OnRev(OUT_A);
   until (FastTimer(0) > timer90grd);
}
```

```
sub turn_right()
{
   // rechts 90
   ClearTimer(0);
   OnFwd(OUT_A);
   OnRev(OUT_B);
   until (FastTimer(0) > timer90grd);
}

sub find_line_left()
{
   OnFwd(OUT_B);
   OnRev(OUT_A);
   until (SENSOR_1 < sensor_sw +5);
}

sub find_line_right()
{
   OnFwd(OUT_A);
   OnRev(OUT_B);
   until (SENSOR_1 < sensor_sw +5);
}
```


RCX 1.0
e0.33l