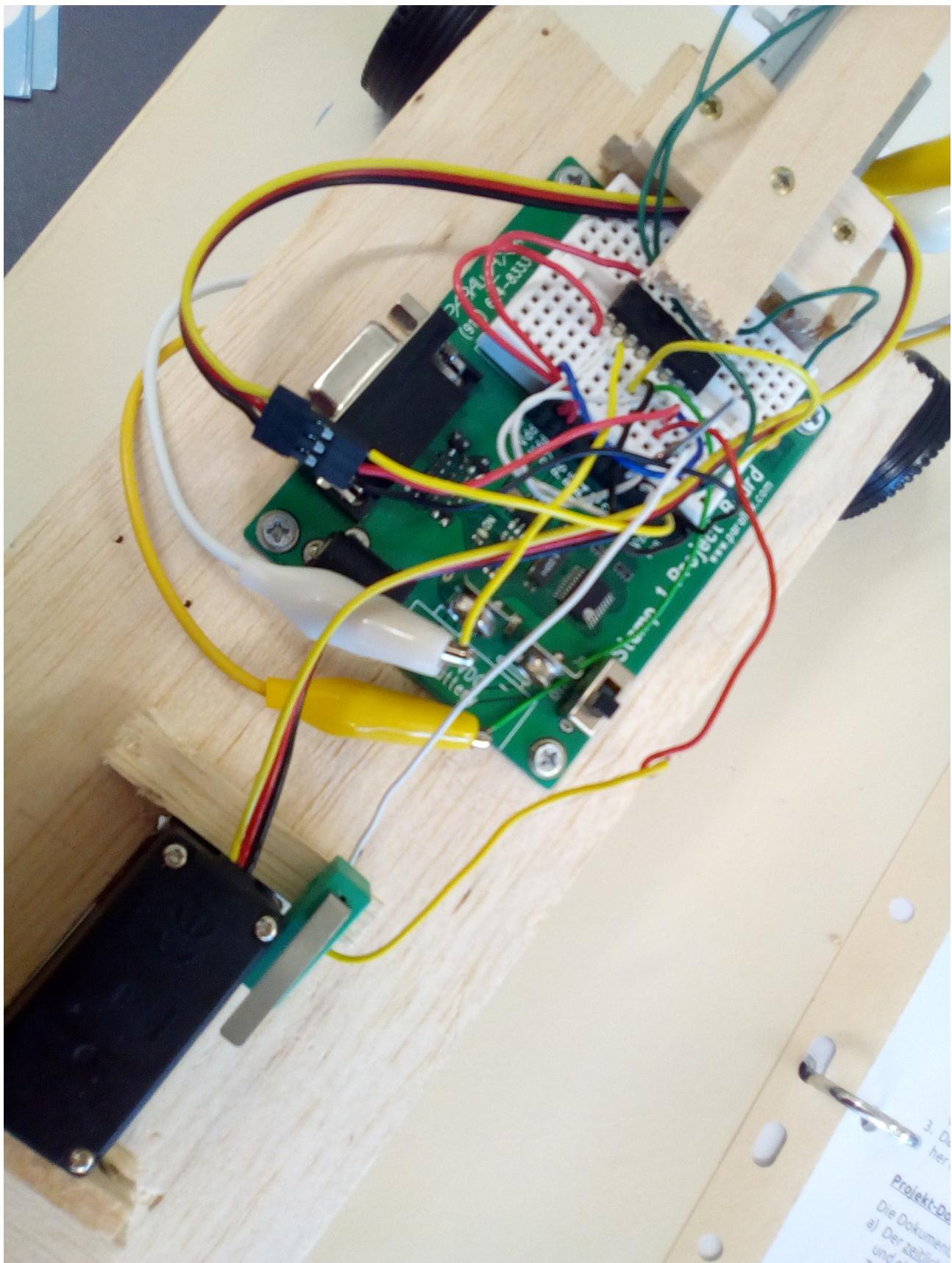


# NWT Projekt – Roboter

## Fabian Exel; 10c



# Inhalt

- 1. Aufgaben Stellung**
- 2. Roboter**
  - 1. Idee**
  - 2. Aufbau**
- 3. Programm**
  - 1. Idee**
  - 2. Aufgaben**
    - 1. Aufgabe 1**
    - 2. Aufgabe 2**
    - 3. Aufgabe 3**
- 4. Basic Stamp Referenz**
- 5. Notizen**

*Laptop: 14*

*Notiz:* Aufgrund späteren überspielen von Dateien auf meinen Raspberry Pi (Konfiguriert als WLAN Hotspot mit Speicher Funktion) könnte es sein, dass der Laptop nicht die neusten Programme von mir hat.

## 1. Aufgaben Stellung

Da dies die Ausgangssituation des Projektes ist, stehen hier nochmals die Aufgaben. Dies ist wichtig für den Bau und Programmierung des Roboters, um zu wissen, was das Ziel überhaupt ist und man einen Bezug dazu entwickelt.

*Aufgabe 1:* Innerhalb eines durch einen schwarzen Streifen abgrenzenden Bereichs liegt ein Stück Alufolie (etwa 10 cm \* 10 cm). Das Fahrzeug soll die Alufolie „suchen“ und, sobald die Alufolie gefunden ist, stehen bleiben. Das Fahrzeug muss dabei innerhalb des abgrenzenden Bereichs bleiben. **Erledigt!**

\* Man muss anfügen, dass der abgrenzende Bereich ein etwas größerer Bereich war. Dieser stammt außerdem von Lego Mindstorms.

*Aufgabe 2:* Das Fahrzeug soll nur dann losfahren, wenn vorher ein Taster gedrückt wurde. Jedes mal wenn das Fahrzeug über einen schwarzen Streifen fährt, soll es die Fahrtrichtung ändern, also von einer Links- zu einer Rechtskurve (bzw. von einer Rechtskurve zu einer Linkskurve) wechseln. **Erledigt!**

*Aufgabe 3:* Das Fahrzeug soll zwischen zwei schwarzen, 10 cm breiten Streifen möglichst schnell hin und her fahren. Dabei darf das Fahrzeug nicht vollständig über einen Streifen hinaus fahren.

**Erledigt!**

## **2. Roboter**

### **2.1 Idee**

Die Idee war, ein möglichst einfaches, kleines und wendiges Fahrzeug zu bauen. Dies ist wichtig um alle 3 Aufgaben möglichst schnell und einfach zu bewältigen. Außerdem muss dieses Fahrzeug möglichst wenig an Material verbrauchen, was jedoch gar nicht so leicht war, weil es sonst brechen würde.

Die erste Idee, alles wie ein Sandwich oder Hochhaus zu bauen, wurde verworfen, weil diese Idee schwierig umzusetzen war und längere Bauzeit bedeutet hätte. Der Vorteil war aber, dass das Fahrzeug einen sehr kleinen Radstand hat, was zu kleinen Kurven und kleinen Wendekreis führen würde. Des weiteren gab es den Vorteil, dass immer genügend Masse auf die Lenkung kam. Die Platine wurde über zwei Stangen festgemacht. Diese Idee wurde dann im späteren Modell als Loch unter der Platine übernommen (Es soll einfach das Holz unter Platine egal wie eingespart werden).

Bei den ersten Test mit dem Roboter wurde dann schnell klar, dass wenn man wie beim Auto beim Einparken, die Lenkung nach hinten setzt, dass man dann besser in eine Lücke reinkommt, weshalb beim Fahrzeug die Lenkung hinten sitzt und die Vorderräder die Antriebsräder sind. Um außerdem Platz zu sparen wurde der Motor hochkant gestellt und mit 2 Holzwürfeln an der Haupt-Platte befestigt. Das war der andere Teil aus der Sandwich Idee, die übernommen wurde.

Der weitere Aufbau des Fahrzeugs sieht so aus, dass die Platine so liegt dass die Batterie möglichst nah am Servo / der Lenkung liegt. Dies ist wichtig damit der eine Reifen von der Lenkung nicht stehen bleibt und über die Fläche rutscht. Des weiteren wurden viele Gedanken über die Radaufhängung des Lenkrads gemacht. Dies ist für gute Fahreigenschaften wichtig. Anfangs gab es die Idee, am Servo jeweils 2 kleine Holzquader dran zu bauen und dazwischen dann das Rad zu montieren. Diese Idee wurde dann aber auch aus Zeit gründen verworfen und stattdessen eine Aufhängung aus Draht gebaut, welche nicht so stabil ist. Ein weiteres Problem war, dass der Reifen manchmal nicht gerade war und hin und wieder mal nicht drehte. Dieses Problem wurde mit je 2 Perlen gelöst, welche sich direkt am Rad an der Aufhängung sich befinden.

## 2.2 Aufbau

Der Aufbau wurde damit begonnen, dass erst mal die Grundplatte gebaut wird. Es wurde für später alles vorgezeichnet, Platte verkürzt und 2 Löcher gemacht, welche für die Platine (siehe Idee) und für den Servo sind. Danach wurde der Motor wie in der Idee beschrieben eingebaut.

Der Servo war etwas schwierig ein zubauen, weil er dieser je 4 Löcher an den Seiten hat, die wie man gesehen hat, auch gebraucht werden, da er sonst vom Holz wegbrechen würde. Anfangs wurde versucht, den Motor mit je 4 Schrauben von unten zu befestigen, doch im späteren Versuch hat sich gezeigt, das dies nicht hält. Also wurde auf zwei Seiten kleine Hölzer angebracht und größere Schrauben eingebaut, womit es dann hielt.

Der LDR hat einen „Arm“, den man bewegen kann auf Wunsch. Dies ist praktisch, weil man einstellen kann, ob der LDR sich vor den Antriebsräädern oder hinter / neben ihnen befinden soll.

Der Taster wurde ohne einer Befestigung angebaut, da die anfängliche Befestigung nicht so gut hielt. Außerdem war beim Taster das Problem, dass er unter vielen Wackelkontakte leidet. Diese zu beheben kann mit einer losen Vorrichtung einfacher sein.

Da das Holz nicht so stabil ist, wurde weitere Stabilität durch ein kleines Holzbrett unter dem Fahrzeug erzielt. Es ist nicht unbedingt notwendig, aber das Fahrzeug hält besser.

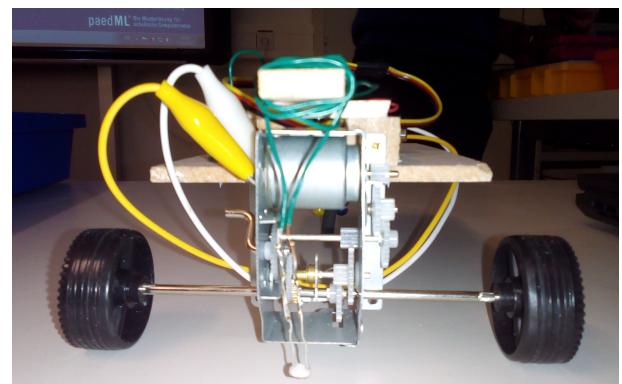
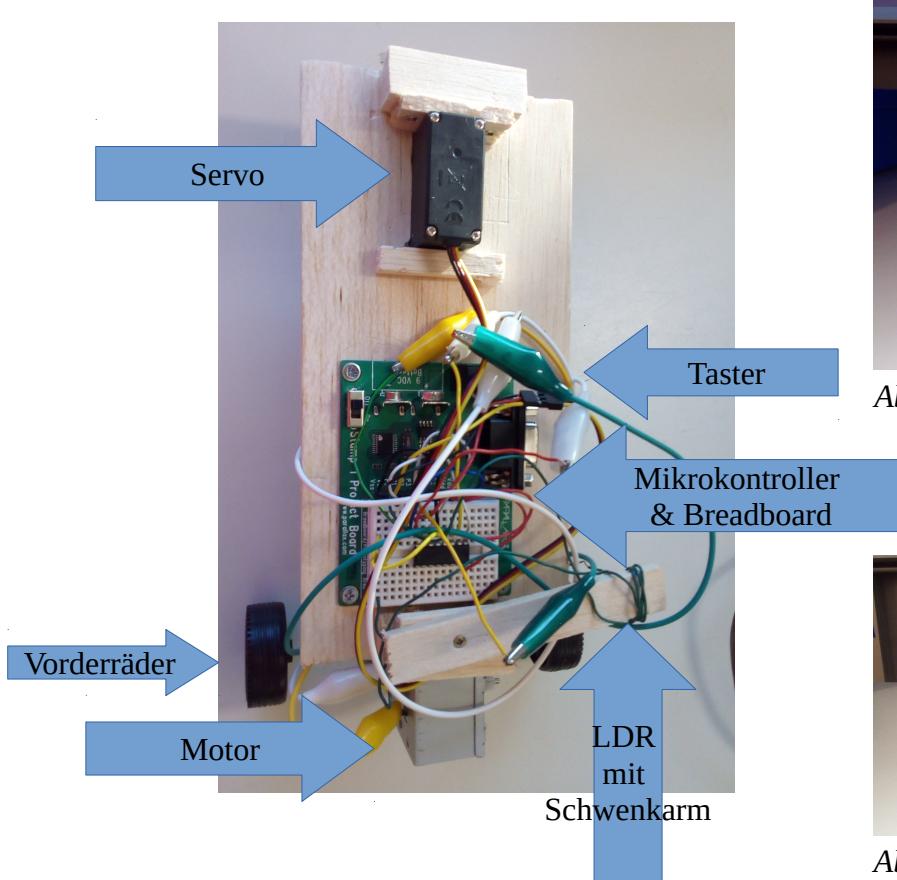


Abbildung 1: Fahrzeug Vorderseite

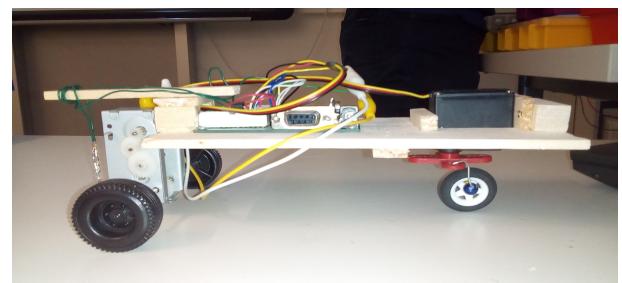
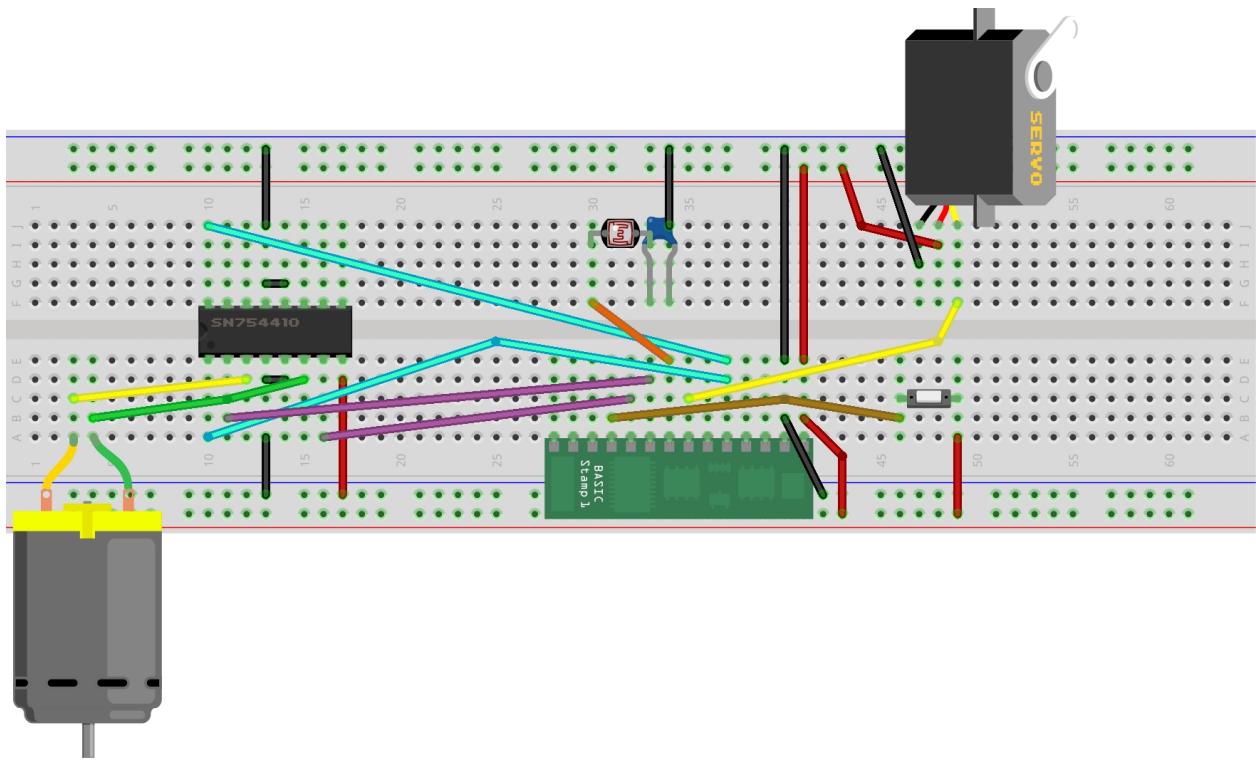
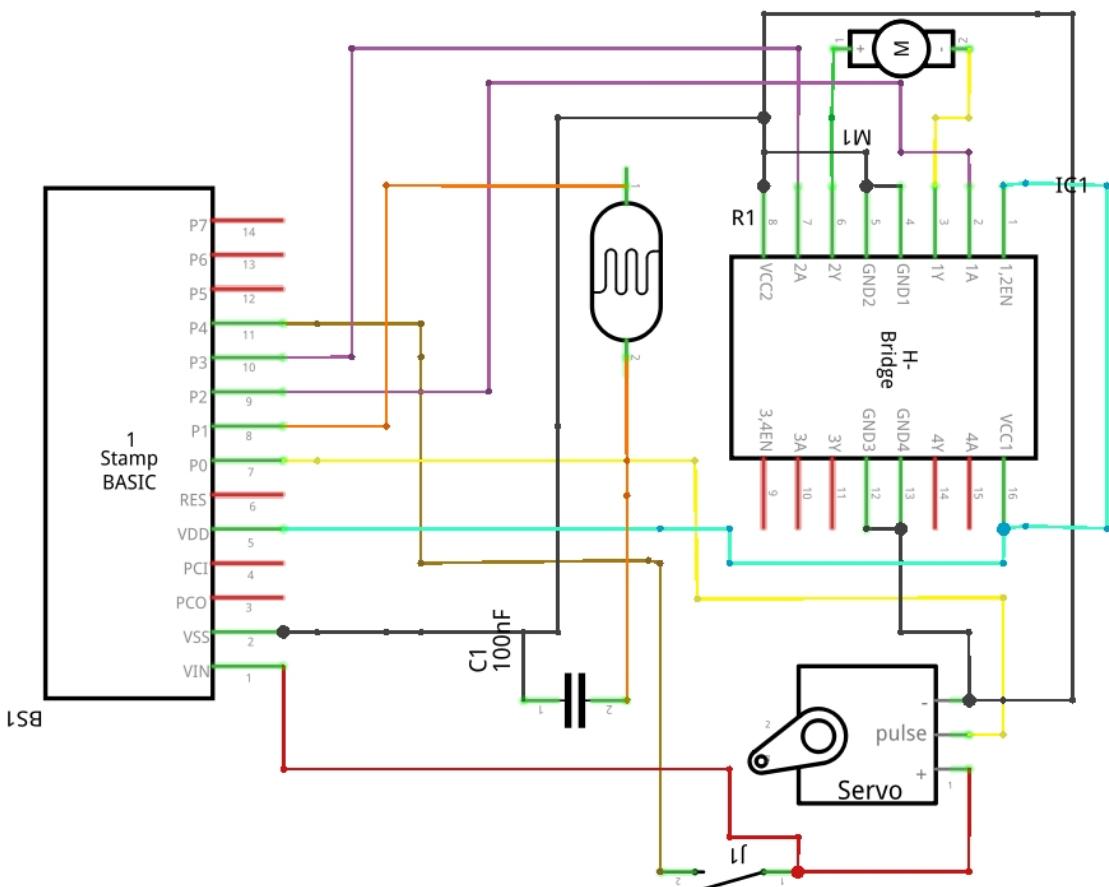


Abbildung 2: Fahrzeug von links

Schaltpläne (Erstellt mit Fritzing (entwickelt von der Universität Kiel) auf Ubuntu 17.10):



Schaltung auf einem Breadboard



Schaltung als Schaltplan (Farben sind gleich geblieben)

### 3. Programm

#### 3.1 Idee

Die Idee war es, ein Programm zu schreiben, welches aus Bausteinen besteht. Das habe ich so gemacht, weil die Entwicklung damit schneller geht und besser verständlich wird. Ein weiterer Vorteil ist es, dass ich in den Programmen später nicht alles speziell und umständlich erklären muss. Doch auch schon in den Programm Bausteinen hier (eingefärbt mit Visual Studio Code (Microsoft) auf Ubuntu 17.10) verweise ich auf den 4. Punkt, die Programm Referenz für die Befehls-Erklärung. In den grünen Notizen stehen die Erklärungen.

##### Sensor / Taster Abfrage mit POT

```
' {$STAMP BS1}  
Start:          'Programm Start  
POT 1, 80, B0   'Abfrage, was Sensor / Taster Wert ist  
DEBUG B0        'Ausgabe des Wertes  
IF B0 < 100 THEN Start  'falls dieser Wert unter 100 liegt, dann soll nichts passieren  
                         DEBUG "Hallo"    'andererseits springe in die nächste Zeile und gebe "Hallo" aus  
GOTO Start       'gehe zum Programm Start
```

##### Motor langsam drehen lassen mit PWM

```
' {$STAMP BS1}  
OUTPUT 3      'H-Bridge  
OUTPUT 2      'H-Bridge  
OUTPUT 0      'Für den Fall der Fälle - ein Servo (wird im Programm aber nicht aktiviert)
```

```
FOR B1=1 TO 10  'For-Schleife, damit der Motor nicht immer, aber eine Zeit sich dreht  
                 LOW 2           'Port 2 auf Low schalten um für eine Potenzial-Differenz zu  
                           sorgen  
                 PWM 3,200,10  'PWM lässt den Motor langsam drehen  
                 LOW 3           'Ausschalten, um Schleife weiterhin wiederholen zu können  
NEXT B1         'For-Schleife bis zum Ende ausführen
```

##### Servo steuern mit PULSOUT

```
' {$STAMP BS1}  
B3 = 0          'Servo Ausgang  
  
FOR B2=0 TO 15  'For-Schleife für den Servo, wird benötigt, da der Servo sonst immer  
                 nur kleine Schritte machen würde  
                 LOW B3         'Port 3 ausgeschaltet, damit der Servo die Frequenz erkennt  
                 PULSOUT B3, 150 'Servo wird am Port 5 gerade gestellt mit einer Stromfrequenz  
                           von 150ms  
                 PAUSE 20       'Pause, damit der Servo Zeit hat sich zu bewegen  
                 HIGH B3        'Port 3 anschalten, damit es zur keiner vom Motor ausgelösten Falsch-  
                           Frequenz kommen kann (Potenzial-Differenz ~ 0)  
NEXT B2         'For-Schleife bis zum Ende ausführen
```

## 3.2 Aufgaben

Die Programme wurden unter anderem aus den obigen Bausteinen zusammen gesetzt. Dort steht dann auch, wie diese Bausteine genau funktionieren. Hier wurde die Erklärung als Notiz in grün geschrieben (eingefärbt mit Visual Studio Code (Microsoft) auf Ubuntu 17.10). Es wurde bei den Programmen auch darauf geachtet, dass nicht unnötige Labels brauchen. Um bei manchen Befehlen noch mal eine andere Erklärung zu erhalten, verweise ich auf 4. Referenz.

### Aufgabe 1 – Programm Version 3

```
' {$STAMP BS1}  
'An welchen Port was angeschlossen ist  
OUTPUT 2  'H-Bridge  
OUTPUT 3  'H-Bridge  
OUTPUT 0  'Servo  
B8 = 0    'Start-Wert für die Auto-Umlenk Variabel
```

```
Start:      'Programm-Start  
B8 = B8 + 1 'Ändere die Auto-Umlenk Variabel um 1  
IF B8 > 10 THEN ender  'Falls die Auto-Umlenk Variabel 10 beträgt, gehe zur Richtungs-  
                        Änderung  
    POT 1, 80, B0   'Messe den Wert vom LDR, um heraus zu finden, auf welchem  
                      Untergrund man sich befindet  
    DEBUG B0       'Gebe zur Kontrolle diesen Wert aus  
IF B0 < 50 THEN fahren  'Falls dieser unter 50 liegt, ist es weiß --> fahre weiter  
IF B0 > 60 THEN zurueck  'Falls dieser über 60 liegt, ist es schwarz --> fahre ein Stück  
                        zurück  
    DEBUG CR, "Gefunden"  'alles zwischen 50 und 60, ist es grau --> Blatt gefunden  
                          (gebe dies hier auch als Ausgabe aus)  
END          'da dass Blatt gefunden wurde, beende das Programm
```

```
zurueck:          'der Part für den Fall, der Roboter kommt auf Schwarz  
DEBUG CR, "zurueck"  'sag mir, dass er zurück fährt  
FOR B2=0 TO 15    'stelle den Servo auf links  
    LOW 0  
    PULSOUT 0, 120  
    PAUSE 20  
    NEXT B2  
    HIGH 0  
FOR B1=1 TO 60    'Fahre ein Stück zurück  
    LOW 2  
    PWM 3,100,10  
    LOW 3  
    NEXT B1  
    GOTO gerade  'Jetzt ist der Roboter weg vom Schwarzen, bedeutet er kann wie  
                  davor weiter machen, doch davor muss der Servo erst mal  
                  gerade gestellt werden (er springt zu diesen Part)  
fahren:            'der Part ist, wenn das Fahrzeug über die weiße Fläche fährt  
FOR B1=1 TO 5     'Das Fahrzeug soll dann nur ein Stück weiter fahren  
    LOW 3
```

```

PWM 2,100,10
LOW 2
NEXT B1      'Wenn dies getan wurde springt das Programm in die nächste
              Zeile um zur Kontrolle den Servo gerade zu stellen
gerade:        'Hier wird soll der Servo gerade gestellt werden. Andere
              Programm Abschnitte springen auch hier hin
FOR B2=0 TO 30    'Servo wird gerade gestellt
  LOW 0
  PULSOUT 0, 150
  PAUSE 20
  NEXT B2
  HIGH 0
  GOTO Start    'gehe wieder zum Anfang
ender:         'springe hier hin, wenn die Richtung einfach so nach einer Zeit
              geändert werden soll (erhöht die Chance das graue zu finden)
DEBUG CR, "richtungswechsel"  'zeige an, dass er die Richtung wechselt
FOR B2=0 TO 20      'stelle den Servo auf rechts
  LOW 0
  PULSOUT 0, 180
  PAUSE 20
  NEXT B2
  HIGH 0
FOR B1=1 TO 15      'Fahre jetzt ein Stück
  LOW 3
  PWM 2,100,10
  LOW 2
  B8 = 0
  NEXT B1
  GOTO gerade   'stelle den Servo wieder gerade (springe zum jeweiligen Part)

```

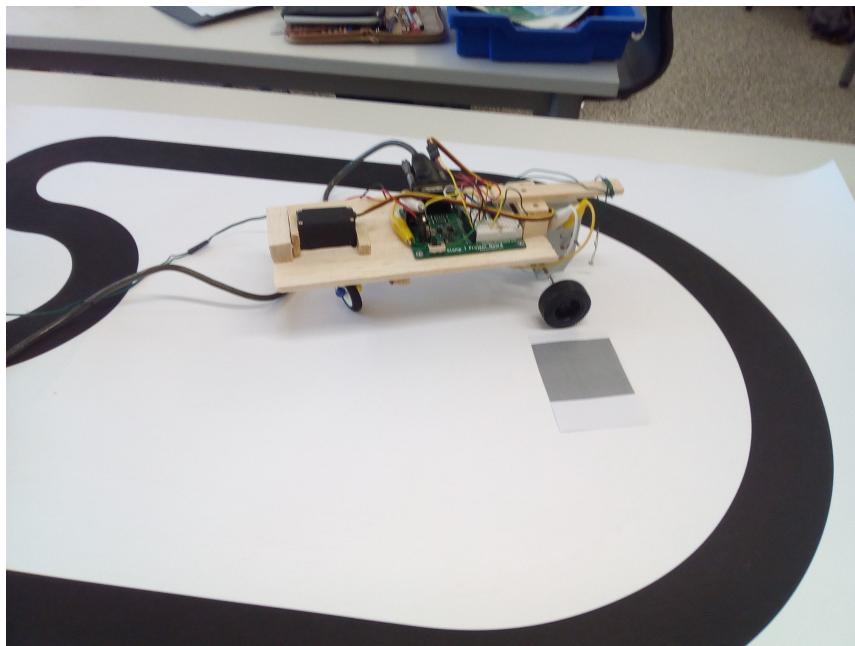


Abbildung 3: Ausprobieren der Aufgabe 1

## Aufgabe 2 – Version 3

```
' {$STAMP BS1}  
OUTPUT 2  'H-Bridge  
OUTPUT 3  'H-Bridge  
OUTPUT 0  'Servo  
INPUT 4   'Taster  
B3 = 120  'Start-Wert für die Lenkung, Variabel für das lenken
```

```
beginn:          'Programm Start  
DEBUG CR, "Programm startet"  'Ausgabe als Rückmeldung  
FOR B2=0 TO 15      'Erstmal die Servo-Grundstellung - Gerade aus stellen  
    LOW 0  
    PULSOUT 0, 150  
    PAUSE 20  
    NEXT B2  
    HIGH 0  
wait: 'Warte-Schleife für den Taster  
POT 4, 80, B0      'Abfrage ob Taster gedrückt ist (kann man auch durchaus mit POT machen - Ein Taster ist nichts anderes als ein Sensor)  
IF B0 < 100 THEN wait  'Falls dieser nicht gedrückt wurde, warte weiter  
start:           'Falls er gedrückt wurde, fange mit dem eigentlich Programm an  
POT 1, 80, B0      'Schau beim LDR nach, ob das Fahrzeug gerade über Schwarz fährt  
DEBUG B0          'Zur Kontrolle eine Ausgabe  
IF B0 > 60 THEN lenkung  'Falls man über Schwarz ist, soll man lenken, ansonsten kann man weiter fahren  
    POT 4, 80, B0  'Vor dem weiter fahren aber noch mal schauen, ob der Taster gedrückt wurde, um das Programm zu unterbrechen und zur Start-Schleife zurück zu gehen  
IF B0 > 100 THEN beginn  'If Abfrage zum oben stehenden (Zeile darüber)  
FOR B1=0 TO 25      'Ein Stück fahren, bevor Taster und Schwarze Farbe noch mal abgefragt werden  
    LOW 3  
    PWM 2,150,10  
    LOW 2  
    NEXT B1  
    GOTO start      'Gehe zum Start zurück (also die Abfrage ob Schwarz und ob der Taster gedrückt wurde)  
lenkung:          'Hier fängt der Part mit der Lenkung an, wenn man über Schwarz gefahren ist.  
DEBUG CR, "lenken"  'Ausgeben, dass das Fahrzeug jetzt lenkt  
if B3 = 120 then rechts  'Abfrage, in welche Richtung das Fahrzeug zuletzt lenkte - und dies dann zu wechseln  
B3 - 60          'wenn zuvor 180 (rechts) war, rechne - 60 um auf 120 (links) zu kommen  
lenken:          'Dies ist der Abschnitt zum lenken.  
FOR B2=0 TO 15      'Stelle den Servo in die gewünschte Richtung (links oder rechts)  
    LOW 0  
    PULSOUT 0, B3  
    PAUSE 20
```

```

NEXT B2
HIGH 0
FOR B1=0 TO 50      'Fahre ein Stück
    LOW 3
    PWM 2,150,10
    LOW 2
NEXT B1
FOR B2=0 TO 15      'stelle den Servo gerade
    LOW 0
    PULSOUT 0, 150
    PAUSE 20
NEXT B2
HIGH 0
GOTO start          'gehe zurück zur Schwarz/Taster abfrage
rechts:              'falls das Fahrzeug zuvor links war, springe hier hin
B3 + 60              'wie obige Rechnung 120 + 60 = 180 = rechts herum
GOTO lenken          'gehe jetzt zum Lenkungs-Part

```

## Aufgabe 3 – Version 2

```

' {$STAMP BS1}
OUTPUT 3  'H-Bridge
OUTPUT 2  'H-Bridge
OUTPUT 0  'Servo

FOR B2=0 TO 30      'Stelle zur Sicherheit am Anfang den Servo gerade
LOW 0
PULSOUT 0, 150
PAUSE 20
NEXT B2
HIGH 0
gehe:                'hier fängt das eigentliche Programm erst an
FOR B1=1 TO 10      'fahre erst mal ein schnelles Stück gerade aus, um nicht auf Schwarz
                     zu sein
    LOW 2
    PWM 3,200,10
    LOW 3
NEXT B1
gerade:              'Der Part, wo das Fahrzeug fährt und zwischen drin auch misst
FOR B1=1 TO 2        'ganz schneller, kleiner Motor schritt
    LOW 2
    PWM 3,200,10
    LOW 3
NEXT B1
POT 1, 80, B0        'Messe den Wert vom LDR
DEBUG CR, B0          'gebe diesen als Kontrolle aus

```

<b>IF B0 &lt; 70 THEN</b> gerade	'Falls dieser unter 70 ist (weiß), soll das Fahrzeug weiter schauen, wo der schwarze Streifen kommt'
<b>FOR B1=1 TO 10</b>	'Falls Schwarz erkannt wurden ist, springe hier hin. Fahre dann ein schnelles Stück zurück, um nicht auf Schwarz zu sein (Wie oben nur in die andere Richtung)
LOW 3	
PWM 2,200,10	
LOW 2	
NEXT B1	
<b>zurueck:</b>	'Hier fährt das Auto ein schnelle Stücke zurück und misst, ob schwarz da ist'
<b>FOR B1=1 TO 2</b>	'kleine, schnelle Schritte zurück'
LOW 3	
PWM 2,200,10	
LOW 2	
NEXT B1	
<b>POT 1, 80, B0</b>	'LDR messen'
<b>DEBUG CR, B0</b>	'Den Wert zur Kontrolle ausgeben'
<b>IF B0 &gt; 70 THEN</b> gehe	'Falls dieser über 70 ist, ändere die Richtung und springe zum Start (gehe)'
<b>GOTO</b> zurueck:	'Wenn nicht, schau weiter wann Schwarz kommt'

## 4. Referenz

### Referenz Basic Stamp

Befehl	Bedeutung	Erklärung
DEBUG [Text, Variabel]	Ausgabe am Laptop	Über ein extra Fenster wird ein programmiert Text angezeigt
'(\$ Stamp BS1)	Mikrocontroller Modell	Für welchen Mikrocontroller das Programm ist
B[nummer]=[var]	Definieren von Variablen	
[var] = [zahl] [operator] [zahl]	Rechnung	
PAUSE [ms]	Programmunterbrechung	
END	Ende des Programms	
DEBUG CR [Text, Variabel]	Ausgabe am Laptop mit neuer Zeile	Über ein extra Fenster wird ein programmiert Text in neuer Zeile angezeigt
OUTPUT [pin]	Port Ausgang	Definition wo welches Bauteil angeschlossen ist
PIN [pin] = [bit var]	Wert für den Port	1 = High; 0 = Low; Port an oder aus schalten
[[label]] „code“ GOTO [label]	Endlosschleife / Springe zum Punkt[[label]]	Ein Code wird wiederholt
FOR [var1] to [var2]	For-Schleife	Schleife startet mit var1 und geht bis var2
NEXT [var1]	(gehört zur For-Schleife)	var1 wird um 1 vergrößert, um irgendwann mal zum Wert von var2 zu kommen
'	Kommentar	
SYMBOL [name] = [var]	Variabel Namen setzen	
INPUT [pin]	Port Eingang	Definition wo welches Bauteil angeschlossen ist
POT [Port], [Faktor], [bit var]	Auswertung analoger Sensoren	Am Port wird ein Wert empfangen, mit einem faktor berechnet und dann als var gespeichert
IF [condition] THEN [label]	Falls-Abfrage	Falls condition eintritt gehe zu label
PULSOUT [pin] [zeit]	Puls Abgabe	Kurze Zeit Spannung auf einem Pin (z.B: für Servosteuerung)
PWM [Pin] [duty] [cycles]	Spannung bestimmen	Spannung wird bestimmt, indem ganz schnell der Port an und aus geht, der resultierende Mittelwert ist dann die Spannung.