Projekt Lagerroboter (Stand 23.11.23)

im Schuljahr 2023/20245

an der Maristenschule Recklinghausen

Fachlehrer: Stefan Langsch

Stand: 28.11.2023

# Inhalt

# Projektidee

Im Rahmen des Technikunterrichtes der Jahrgangsstufe 10 sollen die Schülerinnen und Schüler sich mit moderner Fertigung und Lagerhaltung beschäftigen. In vielen modernen Betrieben werden fahrerlose Transportsysteme eingesetzt. Ein Modell eines solchen autonomen Flurförderfahrzeugs soll im Rahmen des Technikunterrichtes entwickelt und gefertigt werden.

In diesem Projekt ist Teamarbeit gefragt. Die Fahrzeuge sollen in Teams von je drei Schüler:Innen gebaut werden. Bis zur Fertigstellung müssen zahlreiche technische Fragestellungen untersucht werden.

Dabei geht der Weg immer vom Einfachen zum Komplexen und es nicht erforderlich, dass alle Gruppen mit ihrem Fahrzeug gleich weit kommen.

**Stufe 1**

- Bau eines Fahrgestells aus frei wählbaren Materialien. (ggf. 3D-Design und 3D-Druck; Holz- oder Metallbearbeitung)
- Das Modell fährt mit zwei Getriebemotoren. (Brückenschaltung mit Relais)
- Das Fahrzeug stoppt automatisch, wenn ein Mikroschalter ein Hindernis entdeckt. (Brückenschaltung mit Transistoren)
- Das Fahrzeug erkennt mit Hilfe von Reflexkopplern die Fahrbahnmarkierung. (Erprobung der Reflexkoppler; Ätzen und Bestücken einer Platine mit drei Reflexkopplern)

**Stufe 2**

- Ein Mikrocontroller übernimmt die Steuerung. (Auswertung der Taster- und Reflexkopplersignale)
- Ein Ultraschallsensor verhindert Kollisionen. (ggf. 3D-Design und 3D-Druck; Erkundung und Nutzung des Ultraschallsensors)
- Erweiterte Fahrbahnmarkierungen werden ausgelesen. (ggf. 3D-Design und 3D-Druck; Erkundung und Nutzung des Farbsensors)
- Planung der Fahrbahnmarkierungen für die Fabrikhalle. (Nutzung der Software Inkscape)

**Stufe 3**

- Das Paket bestimmt den Weg. (Wie kann ein RFID-Chip auf dem Paket, die Fahrt des Fahrzeugs bestimmen?)
- Das Fahrzeug teilt sich mit. (Wie können Informationen des Fahrzeugs an die zentrale übermittelt werden? (IoT / MQTT)

## Kurze Einordnung des Projektes

Als Techniklehrer an der Maristenschule (Realschule in Recklinghausen) unterrichte ich derzeit (Schuljahr 2023/2024) einen Differenzierungskurs Technik in der Jahrgangsstufe 10 mit 16Schüler*innen. Im Rahmen des Technikunterrichts sollte ein Projekt umgesetzt werden, bei dem die Schüler*innen ihre bisher erworbenen Kenntnisse und Fähigkeiten nutzen und erweitern können. Da in diesem Schuljahr auch eine Unterrichtssequenz über die Entwicklung der Industrie von der ersten industriellen Revolution bis zur Industrie 4.0 auf dem Plan steht, sollte das Projekt einen Bezug dazu haben. Die Wahl fiel auf einen autonomen mobilen Roboter.

Zu Beginn der Unterrichtsreihe stand eine Betriebsbesichtigung bei der Firma Hella in Recklinghausen. Dort werden zwei verschiedene mobile Roboter eingesetzt, wobei die Integration eines Robotertyps noch nicht abgeschlossen ist. Dabei haben die Schüler*innen viel über die verschiedenen Typen erfahren und vor allem zur Kenntnis genommen, dass es sich um ein höchst aktuelles Thema handelt.
Im Anschluss an diese Betriebsbesichtigung haben wir mit der Planung eines Robotermodells für eine fiktive Produktionshalle begonnen.

An der Maristenschule gibt es nur in jedem dritten Jahr einen Technikkurs in der Jahrgangsstufe 10. Das Projekt Lagerroboter wurde zum ersten Mal durchgeführt. Die letzte Technikkurs entwickelte eine automatische Abfüllanlage mit Fließband. Die Begeisterung für das Projekt bei den Schüler*innen spricht allerdings deutlich für eine Wiederholung.

Die Schüler*innen des Technikkurses sind 15-16 Jahre alt. Sie haben sich in Zweier- und Dreiergruppen aufgeteilt. Es zeigt sich, dass Dreiergruppen für das Vorankommen erheblich günstiger sind. So ist es durch Erkrankungen schon häufiger vorgekommen, dass nur eine Person aus der Gruppe anwesend war und dadurch nur wenig Fortschritt erzielt werden konnte. Es ist auch nicht zu befürchten, dass es bei Dreiergruppen nicht genug Arbeit für jeden gibt.?

# Detail-Infos zum Projekt

## Teile-Liste

Die folgende Teileliste bezieht sich auf einen Roboter. Nicht alle Teile werden/wurden von allen Gruppen genutzt.

| Artikel | Stückzahl | Lieferant | Stückpreis |
|---|---|---|---|
| GY-31 TCS3200 Farbsensor | 1 | Funduino | 8,45 € |
| RFID-KIT mit Mifare RC522 Empfänger | 1 | Funduino | 2,29 € |
| Kippschalter (blau) Rocketswitch EIN / EIN Rastend 125V 5A | 1 | Funduino | 1,13 € |
| Arcade Taster - 27mm Gewindedurchmesser - verschiedene Farben Blau | 1 | Funduino | 1,31 € |
| Arcade Taster - 27mm Gewindedurchmesser - verschiedene Farben Gelb | 1 | Funduino | 2,49 € |
| Ultraschallsensor, Entfernungssensor - HC-SR04 | 1 | Funduino | 1,74 € |
| L298N H-Brücke für zwei Motoren / Motortreiber | 1 | Funduino | 2,58 € |
| 3-6V Getriebemotor mit Rad, Reifen | 2 | Funduino | 1,61 € |
| Stabile Lenkrolle mit Platte - verschiedene Größen 1 Zoll | 1 | Funduino | 1,31 € |
| Abstandssensor, Hinderniserkennungssensor TCRT5000 - Infrarot (IR) - 90° | 3 | Funduino | 1,44 € |
| Funduino MEGA 2560 R3 Mikrocontroller - Arduino kompatibel | 1 | Funduino | 26,01 € |
| Meterware - Verlängerungskabel für LED Streifen, 4-Pin | 1 | Funduino | 1,01 € |
| FOREX® Hartschaumplatte grau matt 150 x 150 x 3 | 1 | Funduino | 1,25 € |
| FOREX® Hartschaumplatte grau matt 150 x 150 x 5 | 1 | Expresszuschnitt | 1,34 € |
| Distanzhülsen (20, 25, 30mm) | 1 | opitec | |
| Quadratlochblech (0,7 x 150 x 500) | 1 | opitec | 4,59 € |
| Mikroschalter | 2 | reichelt | 0,76 € |
| Miniatur-Kippschalter | 1 | reichelt | 0,76 € |
| Batteriehalter | 1 | reichelt | 0,67 € |
| Sperrholz | div. | Fundus | |
| Schrauben und Muttern 3mm | div. | Fundus | |
| LED und Widerstände | div. | Fundus | |

## Geräte-Liste

Den Gruppen sollte bei der Konstruktion des Roboters möglichst viel Freiheit gewährt werden. Dadurch ergab sich, dass mit verschiedenen Materialien und Werkzeugen gearbeitet wurde

- Laubsäge, Feinsäge, Dekupiersäge
- Akkuschrauber, Standbohrmaschine
- Lötkolben und dritte Hand
- Digitalmultimeter
- 3D-Drucker (Ultimaker 2+)
- Heißluftföhn für Schrumpfschläuche
- diverse Kleinwerkzeuge (z-B. Flachzange, Feilen, Schleifpapier, Schraubendreher, Blechschere)

## Fotos zum Projekt

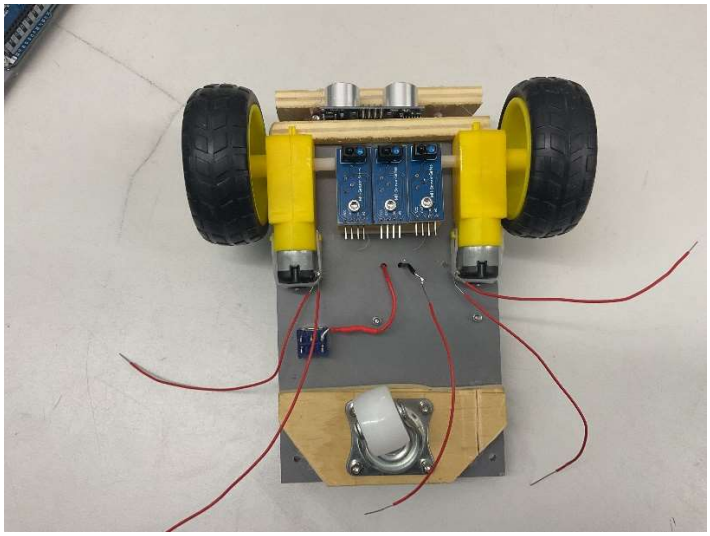


*Abbildung 1: Roboter von unten mit Rädern, Lenkrolle und Liniensensoren*

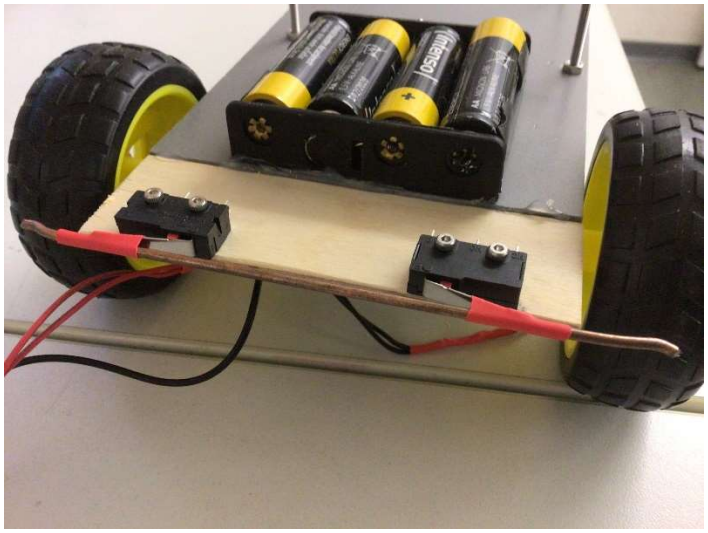


*Abbildung 2: Not-Aus-Taster und Batteriefach eines Roboters*

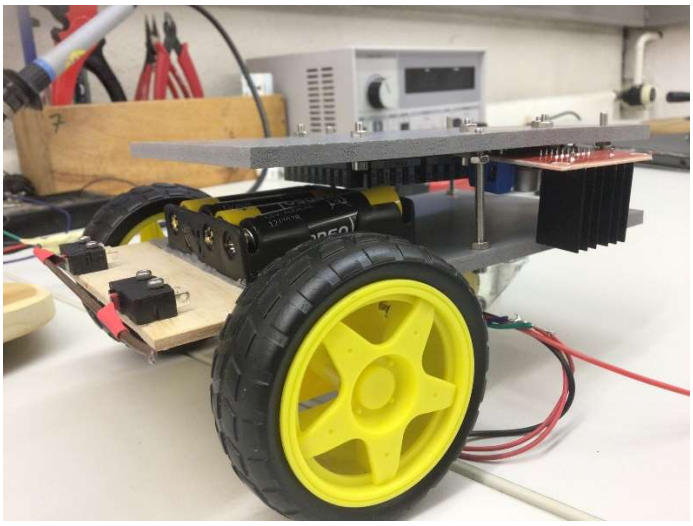


*Abbildung 3: Motortreiber, Batteriefach und Mikrocontroller in der Zwischenebene*

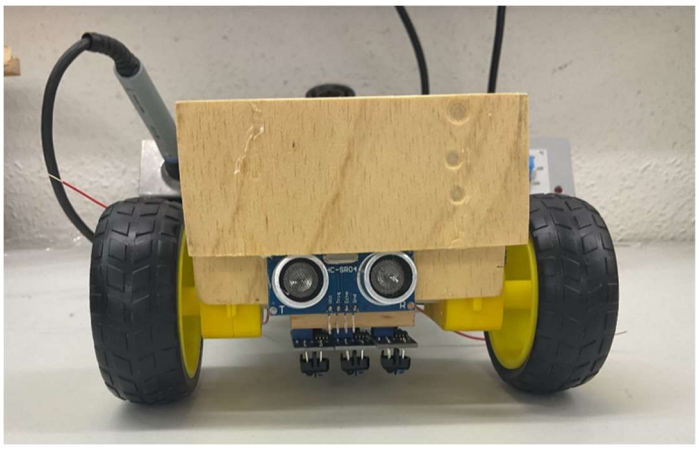


*Abbildung 4; Frontpartie eines Roboters (noch ohne Not-Aus-Taster)*

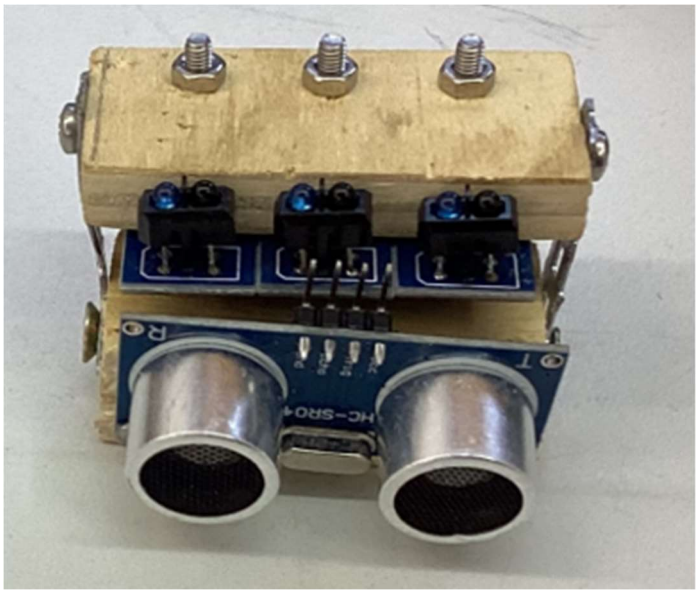


*Abbildung 5: Liniensensor und Ultraschallsensor vor der Montage*

## 3D-Objekte

Von einigen Schülergruppen wurden 3D-Objekte für den Roboter erstellt.

Dafür wurde Tinkercad von Autodesk benutzt. Die Schüler*innen haben bereits in den vergangenen Jahren Erfahrungen mit der App gesammelt.

Vorteile von Tinkercad:

- eingeschränkter Funktionsumfang (→ Übersichtlichkeit)
- Möglichkeit der Arbeit auf allen Endgeräten im Browser oder der App
- Anlegen von Kursen (→ Zugriff auf Entwürfe der Schüler*innen)

**Bisher erstellte Objekte:**

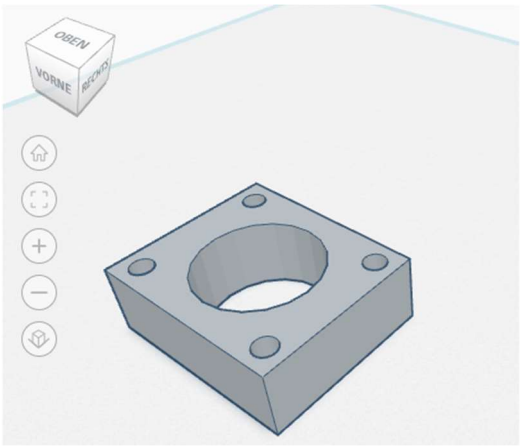



*Abbildung 6: Distanzstück für die hintere Lenkrolle*

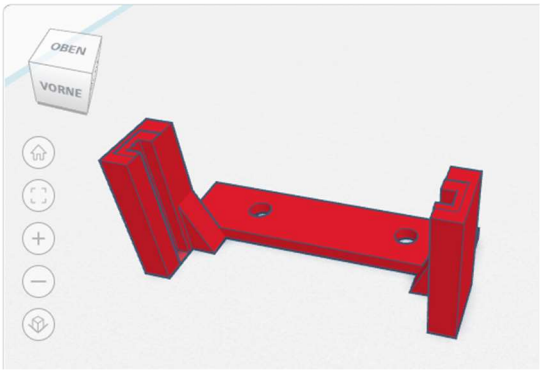



*Abbildung 7: Halterung für den Ultraschallsensor*

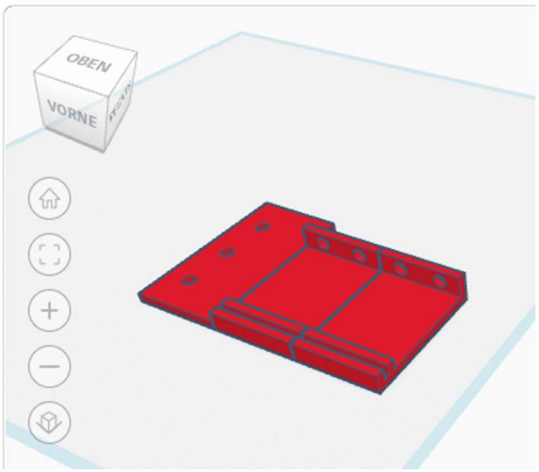



*Abbildung 8: Halteelemente für die Liniensensoren*

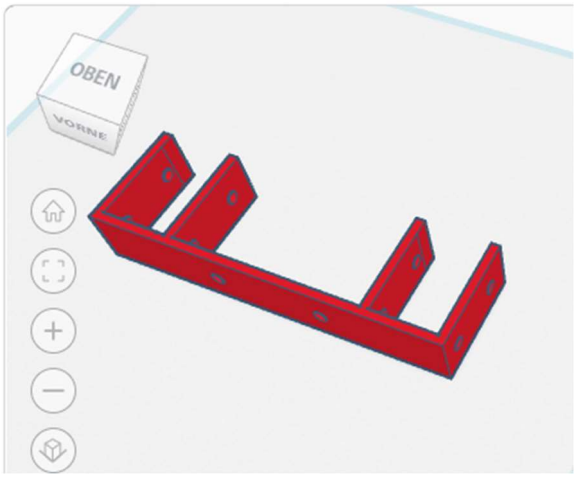



*Abbildung 9: Motorhalterung*

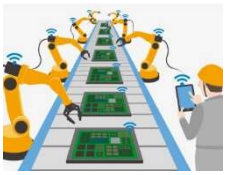
## Kontakt

Bei Fragen zum Projekt melden Sie sich gerne bei:

stefan.langsch@mrr.bistum365.de

## Arbeits- und Informationsblätter

Das vorhandene Technikbuch (starke Seiten Technik, Ernst Klett Verlag GmbH Stuttgart 2020) kann nur sehr eingeschränkt für das Projekt verwendet werden. Daher wurden (und werden) speziell für das Projekt zahlreiche Arbeits- und Informationsblätter erstellt. Die folgende Sammlung beinhaltet die bisher erstellten Arbeits- und Informationsblätter sowie einen Teil einer Kursarbeit, die während der Projektzeit geschrieben wurde. Da das Projekt noch nicht abgeschlossen ist, ist auch die Sammlung noch nicht vollständig.

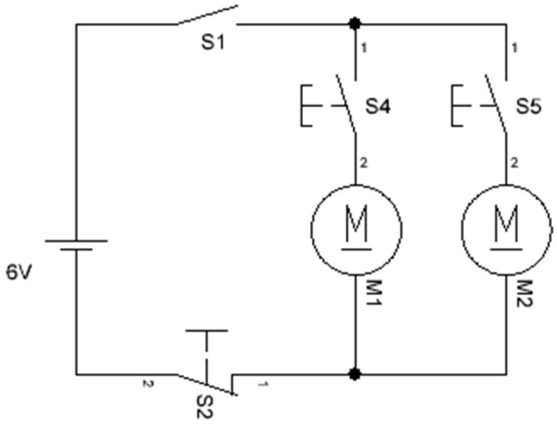


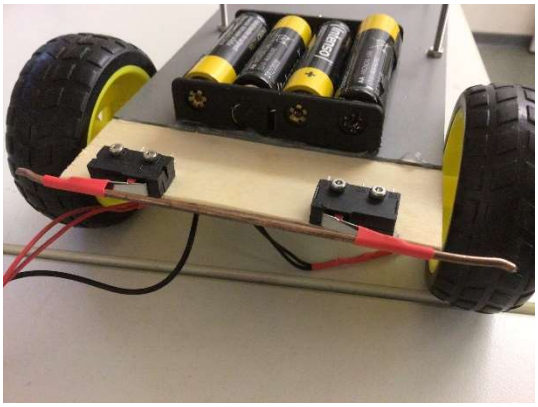
## Manuelle Robotersteuerung

Bei einer Betriebserkundung haben wir erfahren, dass die eingesetzten Roboterfahrzeuge zunächst per Fernbedienung durch die Halle gefahren werden. Dabei scannen sie die gesamte Halle. Unsere Roboter sollen ebenfalls manuell gesteuert werden können.

Eine selbstgebaute kabelgebundene Fernsteuerung könnte so aussehen, wie in der Abbildung rechts.

## Elektrische Schaltung



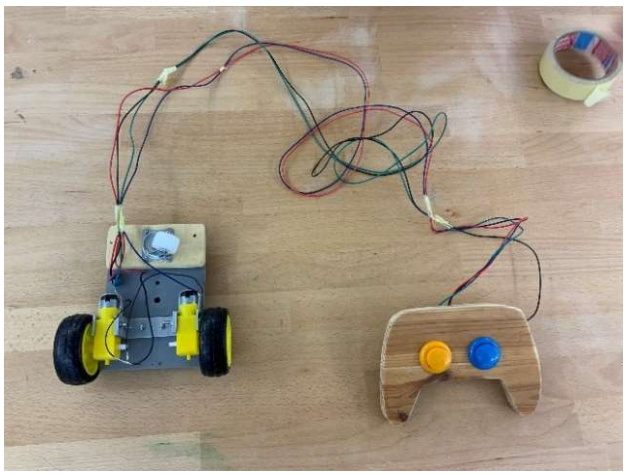



Sobald der Hauptschalter S1 geschlossen ist, können die Motoren getrennt voneinander mit den Tastern S4 und S5 angesteuert werden.

Wird der Öffner-Taster S2 an der Front des Roboters angebracht, wie auf dem Foto oben, so stoppt der Roboter im Falle einer Kollision. Solche Not-Aus-Bumper haben wir bei der Betriebserkundung an den Robotern entdeckt.

**Aufgaben**

1. Begründe, warum es nicht sinnvoll ist die beiden Motoren in Reihe zu schalten.
2. Die Microumschalter auf dem Foto können als Öffner und als Schließer genutzt werden. Die Funktionsweise hängt davon ab, an welche Anschlösse man die Leitungen lötet.
   Peter hat den Schalter so angeschlossen, dass dieser nun die Funktion eines Schließers hat. Erkläre, welche Folge das für die Funktionsweise hat.



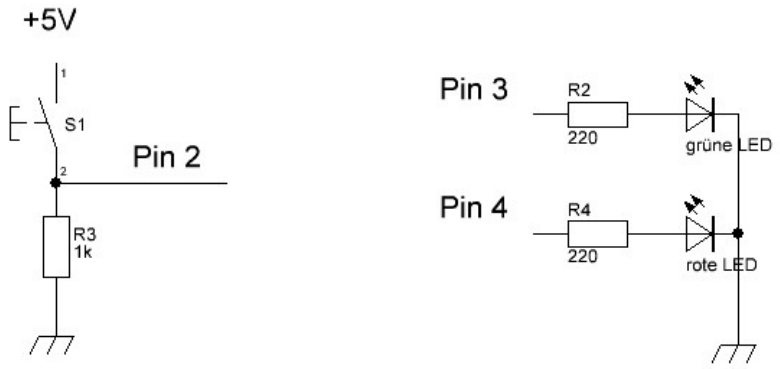
So könnte der manuell steuerbare Roboter aussehen:

## Statusanzeige mit Leuchtdioden

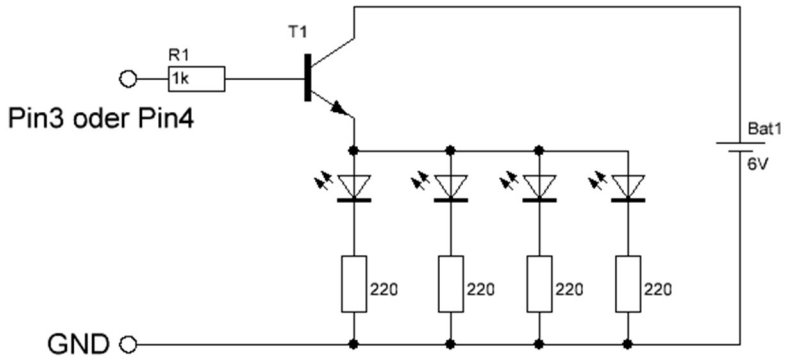
Bei der Betriebsbesichtigung konnten wir beobachten, dass die Roboterfahrzeuge ihren Status über farbiges Leuchten anzeigen. Leuchten sie grün, so ist alles in Ordnung. Ein rotes Leuchten zeigt, einen Fehler an. Eine solche Funktionalität soll unser Fahrzeug ebenfalls haben.
Solange der Not-Aus-Bumper nicht betätigt ist, leuchtet eine grüne LED. Sobald er betätigt wird, erlischt diese und eine rote LED leuchtet. Wird der Taster wieder freigegeben, ändert sich die Anzeige wieder. Das Programm wiederholt sich ständig.

**Blockdarstellung**


**Schaltung mit Signalaufbereitung**

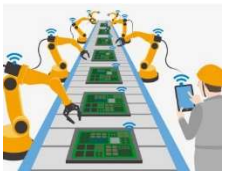



Sollen mehrere grüne oder rote LED verwendet werden, muss man ein Verstärkerschaltung nutzen. Das liegt daran, dass die Leistung der Mikrocontroller-Ausgänge nicht für das Betreiben größerer Lasten ausgelegt ist.




**Aufgaben:**

1. Zeige mit einer Rechnung, dass ein 220Ω Widerstand eine gute Wahl als Vorwiderstand ist.
   Betriebsdaten der LED: $U_{LED}$ = 2V; $I_{LED}$ = 20mA = 0,02A
   Erinnerung: In der Reihenschaltung gilt: $I_1=I_2=I_3=\ldots$ und $U_{ges}=U_1+U_2+\ldots$
   außerdem gilt: $R = \frac{U}{I}$



**Code des zugehörigen Programms**

```
/*  Hier können Informationen zum Programm stehen
Programm: Statusanzeige des Roboters
Version:
Programmierer:
Datum:
*/

// Hier können vorab Variablen festgelegt werden.
int taster = 2;     //Der Taster ist an Pin 2 angeschlossen.
int grueneLED = 3; //Die grüne LED ist an Pin 3 angeschlossen.
int roteLED = 4;    //Der rote LED ist an Pin 4 angeschlossen.
int tasterzustand; //Erzeugung einer Variablen zum Speichern des Tasterzustandes

// Alles was im folgenden Block steht, wird nur einmal ausgeführt.
void setup() {
  // initialisieren der Ein- und Ausgänge.
  pinMode(taster, INPUT);
  pinMode(grueneLED, OUTPUT);
  pinMode(grueneLED, OUTPUT);

  digitalWrite(roteLED, HIGH);   // rote LED einschalten
  digitalWrite(grueneLED, LOW); // grüne LED ausschalten
  delay(2000); //warte 2000ms bzw. 2s
}

// Alles was im folgenden Block steht, wird in einer
// Dauerschleife wiederholt.
void loop() {
  tasterzustand = digitalRead(taster); //Lesen und Speichern des Tasterzustands

  if (tasterzustand ==  LOW){       // wenn der Schalter nicht gedrückt ist
    digitalWrite(grueneLED, HIGH); // schalte die grüne LED an
    digitalWrite(roteLED, LOW);     // schalte die rote LED aus
  }
  else {                            // wenn der Schalter doch gedrückt ist
    digitalWrite(grueneLED, LOW); // schalte die grüne LED aus
    digitalWrite(roteLED, HIGH);  // schalte die rote LED an
  }
}
```

Übrigens:
Die Programmierumgebung kann man hier kostenfrei herunterladen:
http://arduino.cc/en/Main/Software

## Übersicht zur Programmierung

### Allgemeine Tipps zur Programmierung:

- Beginne dein Programm mit einer Beschreibung und Informationen für den Leser:
  ```
  /* meine Informationen */
  ```
- Kommentier die einzelnen Programmzeilen, so dass ein unbekannter Leser das Programm auch verstehen kann:
  ```
  // mein Kommentar
  ```
- Benutze sinnvolle, merkbare Namen bei eigenen Benennungen.
- Speichere dein Programm unter einem sinnvollen Namen.
- Generell sind alle Befehle mit einem Semikolon zu versehen.
  (Ausnahme: Nach geschweiften Klammern kommt nie ein Semikolon!!)
- Bei Befehlen GENAU auf Groß- und Kleinschreibung achten.

Übrigens:
Ein Arduino-Programm nennt man **Sketch**.

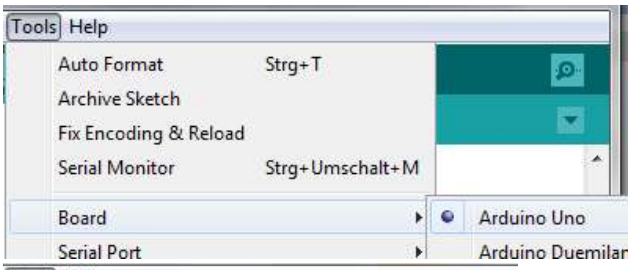
### Ein Programm auf Syntaxfehler testen:

Klicke auf  oder drücke **strg + r.**

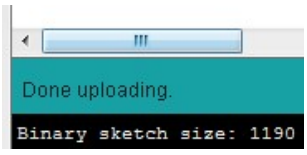
### Ein Programm auf den Mikroprozessor übertragen

Kontrolliere, ob das Board (z.B. "Arduino UNO") und der Port richtig ausgewählt sind. Das Programm gibt eine Fehlermeldung heraus, falls dies nicht der Fall ist.

Klicke auf  oder drücke **strg + u.**
Hat alles funktioniert, so erscheint am unteren Bildschirm die Meldung "*Done uploading.*", falls nicht, wird auf Fehler hingewiesen.








### Aufbau eines Programmes:

1. Beschreibung des Programmes als Kommentar
2. Definition von Variablen
3. `void setup() { meine Festlegungen}`
   Der Setup-Teil in den geschweiften Klammern wird ein Mal zu Beginn ausgeführt. Dort lassen sich z.B. die einzelnen Pins des Arduinos als Ein- oder als Ausgang festlegen.
4. `void loop() { mein Hauptprogramm }`
   Der Loop-Teil in den geschweiften Klammern wird immer wieder ausgeführt. Er stellt das eigentliche Hauptprogramm dar.

Beispielsprogramm, was NICHTS tut:

```
/*Ich mache rein gar nichts, da
sowohl das setup als auch der loop leer sind!*/

void setup() {
}
void loop() {
}
```

## Kommentare schreiben

`/*` Beginn eines mehrzeiligen Kommentars

`*/` Ende eines mehrzeiligen Kommentars

`//` Einzeiliger Kommentar

## Festlegen der Funktionen der Anschlüsse

```
void setup() {
      pinMode(2, OUTPUT);       // Pin 2 soll ein Ausgang sein:
      pinMode(3, OUTPUT);       // Pin 3 soll ein Ausgang sein:
 }
```

## Wiederholungsschleife (Hauptprogramm)

```
void loop() {
      digitalWrite(2, HIGH); // Pin2 wird high gesetzt und damit die LED
eingeschaltet.
      delay(500);            // Pause für 500ms = 0,5 Sekunden
      digitalWrite(2, LOW);  // Pin2 wird low gesetzt und damit die LED
ausgeschaltet.
      delay(500);            // Pause für 500ms = 0,5 Sekunden
}
```

## Befehlsübersicht I:

| Befehl | Kurbeschreibung | Kommentar |
|---|---|---|
| `void setup() { }` | Der Code in den geschweiften Klammern wird genau 1x ausgeführt. | |
| `void loop() { }` | Der Code in den geschweiften Klammern wird immer wieder ausgeführt. | |
| `pinMode(a,b);` | Der Pin a wird als Ein- oder als Ausgang eingestellt.<br>b kann OUTPUT oder INPUT sein. | Festlegen, was Ein- und was Ausgänge sind. |
| `digitalWrite(a,b);` | Der Pin a wird auf den Wert b gelegt. b kann entweder HIGH oder LOW sein. | Steuern von Ausgängen. |
| `delay(a);` | Der Arduino macht für a Millisekunden eine Pause. | |
| `digitalRead(a);` | Es wird der Zustand des Pins a ausgelesen. | Einlesen von Eingängen. |

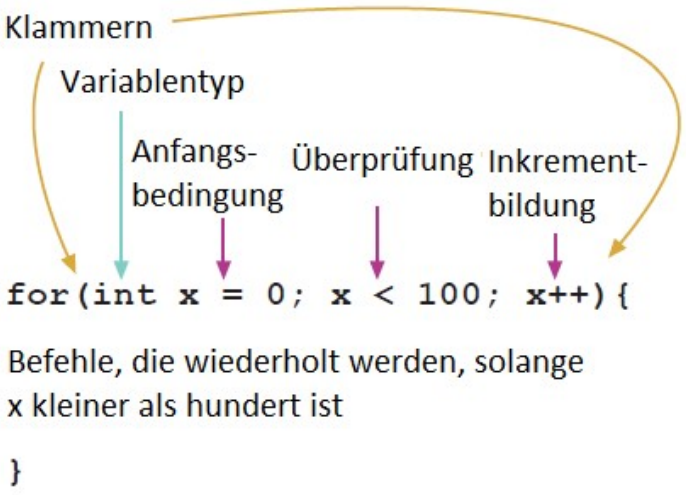
## Nutzung von Variablen, Zuweisungen und Wenn- Dann- Bedingungen

Die LEDs an Pin 9 sollen allmählich heller werden bis sie ihre volle Helligkeit erreichen

```
int helligkeit = 0;    // wie hell die LEDs sind
int schrittweite = 5;    // Schrittweite der Helligkeitszunahme

void setup()  {
  pinMode(9, OUTPUT);
}

void loop()  {
  analogWrite(9, helligkeit);      // Pin 9 wird der  Helligkeitswert helligkeit zugewiesen

  helligkeit = helligkeit + schrittweite;

  if helligkeit == 255 {  //Wenn der Helligkeitswert 255 beträgt, dann …
    schrittweite = 0;   // Denn der Wert darf 255 nicht überschreiten.
  }
  delay(30);
}
```

## for-Schleife

Soll eine Befehlsfolge mehrfach wiederholt werden, kann eine Schleife programmiert werden. Die for-Schleife wird solange wiederholt bis die Abbruchbedingung (hier x =100) erreicht ist.



## Übungen zur Programmierung 1

1. Notiere, wie ein einzeiliger Kommentar beginnt.
2. Notiere, wie ein mehrzeiliger Kommentar beginnt und endet.
3. Erkläre kurz, was der Unterschied zwischen den Blöcken ***setup { }*** und ***loop { }*** ist.
4. Gegeben ist ein Ausschnitt aus einem Mikrocontroller-Programm.

```
8  // Hier können vorab Variablen festgelegt werden.
9  int taster = 2;     //Der Taster ist an Pin 2 angeschlossen.
10 int grueneLED = 3; //Die grüne LED ist an Pin 3 angeschlossen.
11 int roteLED = 4;    //Der rote LED ist an Pin 4 angeschlossen.
12 int tasterzustand; //Erzeugung einer Variablen zum Speichern des Tasterzustandes
13
14 // Alles was im folgenden Block steht, wird nur einmal ausgeführt.
15 void setup() {
16   // initialisieren der Ein- und Ausgänge.
17   pinMode(taster, INPUT);       //taster (Pin 2) wird als Eingang festgelegt
18   pinMode(grueneLED, OUTPUT);  //
19   pinMode(roteLED, OUTPUT);
20
21   digitalWrite(roteLED, HIGH);   // rote LED einschalten
22
23   for (int i=0; i<10; i=i+1){ //Start der for-Schleife
24     digitalWrite(grueneLED, HIGH); // grüne LED ausschalten
25     delay(500);                    //
26     digitalWrite(grueneLED, LOW);  //
27     delay(500);
28   }  //Ende der for-Schleife
29 } //Ende des Setups
```

   a) Notiere sinnvolle Kommentare für die Zeilen 18, 25 und 26.

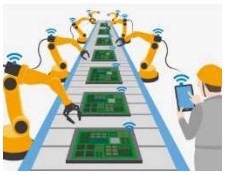
   b) Notiere, wie oft die for-Schleife durchlaufen wird. Tipp: i = i + 1 in Zeile 23 bedeutet, dass der Wert von i nach jedem Durchlauf der Schleife um 1 erhöht wird.

   c) Marvin überträgt das Programm und beobachtet die Leuchtdioden. Notiere seine Beobachtung.

5. Dennis möchte eine weiße LED als Scheinwerfer an seinem AMR betreiben. Die LED schließt er an Pin 5 an. Notiere, welche Programmzeilen er einfügen muss, damit die LED nach ausgeführtem Setup dauerhaft leuchtet.
6. Michaela möchte ihr Programm auf den Mikrocontroller übertragen. Es wird aber angezeigt, dass es Fehler in den Zeilen 44 und 52 gibt. Korrigiere die fehlerhaften Programmzeilen.

   44 | digitalwrite(grueneLED, HIGH);

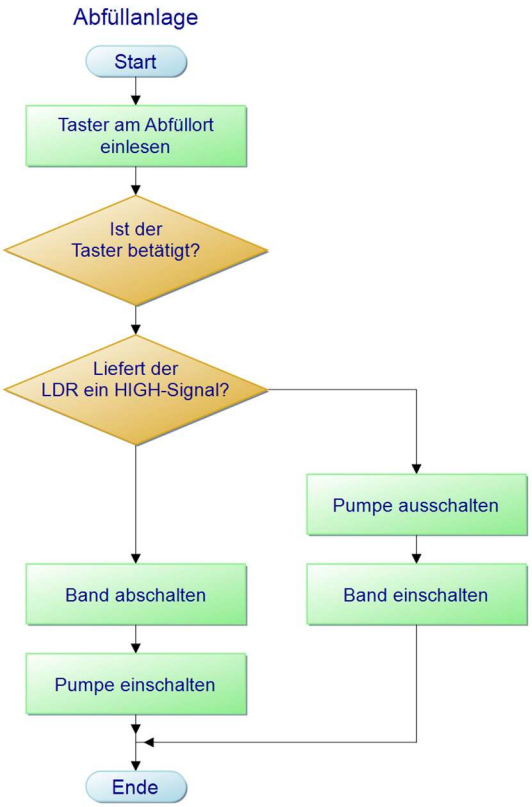
   52 | delay(1000)

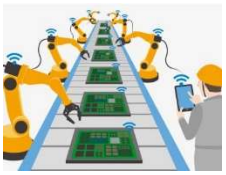


## Übungen zur Programmierung 2

1. Zur Steuerung einer Abfüllanlage wurde ein kleines Mikrocontroller-Programm geschrieben. Sobald ein Becherglas den Abfüllort erreicht, betätigt es einen Taster. Mit einem Lichtabhängigen Widerstand (LDR) kann ermittelt werden, ob das Glas schon voll ist.
   a) **Zeichne** ein Blockschaltbild für das Fließband (EVA).
   b) **Korrigiere** die Fehler im Setup des Programms.
   c) **Notiere**, welcher Befehl in Zeile 17 ergänzt werden muss.
   d) **Notiere** Kommentare für die Zeilen 15, 19 und 20.
   e) Wenn man zu Beginn eines Programms (im Setup) nicht festlegt, ob ein Ausgang HIGH oder LOW geschaltet sein soll, dann ist er Anfangszustand der Aktoren ungewiss.
   **Notiere** die Möglichkeiten, die sich für die Abfüllanlage ergeben.
   f) **Notiere** die Befehle, die sinnvollerweise nach Zeile 10 im Setup ergänzt werden sollten.

```
1  int Taster = 1;   //Taster an pin 1
2  int LDR    = 2;   //LDR an pin 2
3  int Band   = 3;   //Bandmotor an pin 3
4  int Pumpe  = 4;   //Pumpe an pin 4
5
6  void setup() {    //wird nur einmal durchlaufen
7   pinMode(Taster, INPUT);  //Taster als Eingang
8   pinMode(LDR, INPUT);     //LDR als Eingang
9   pinMode(Band, INPUT);    //Bandmotor als Ausgang
10  pinMode(Pumpe, INPUT);   //Pumpe als Ausgang
11 }
12
13 void loop() {    //wird immer wieder durchlaufen
14   if (digitalRead(Taster == HIGH))  //wenn der Taster betätigt wird
15     if (digitalRead(LDR)== HIGH) {
16       digitalWrite(Band, LOW);      //Bandmotor ausscha
17                                  }   //Pumpe einschalten
18     else {
19       digitalWrite(Pumpe, LOW);     //
20       digitalWrite(Band, HIGH);}
21 }
```



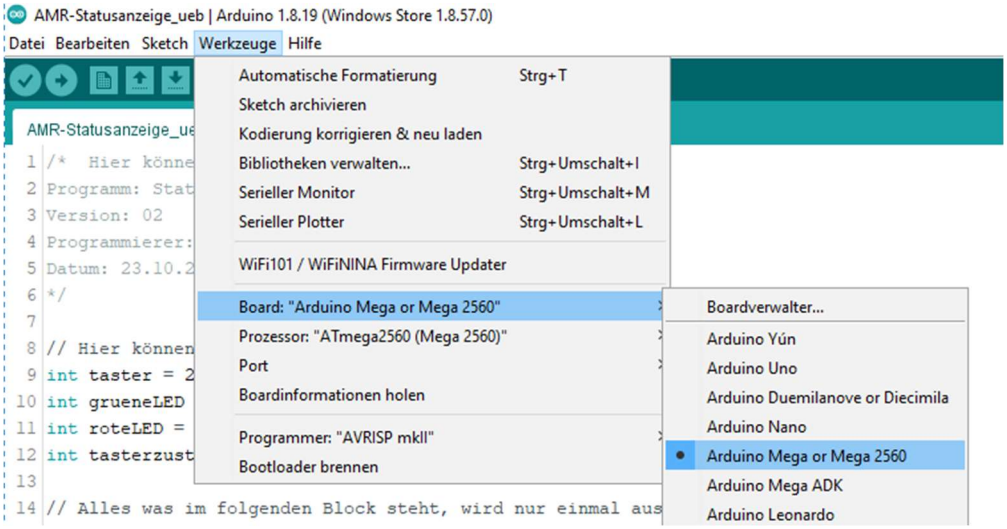


## Programm prüfen und übertragen

Bevor das Programm auf den Mikrocontroller übertragen werden kann, müssen in der Programmierumgebung einige Einstellungen vorgenommen. Bei jedem neuen Start der Programmiersoftware müssen diese Einstellungen überprüft werden.

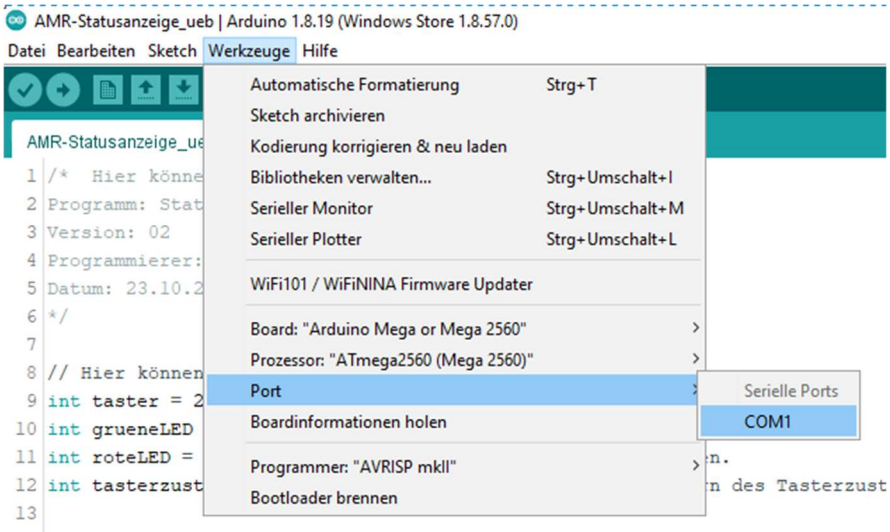
### Auswahl des Mikrocontrollers

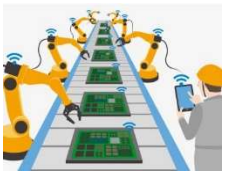
Es gibt zahlreiche Mikrocontroller. Damit die Übertragung funktioniert, muss unter der Rubrik *Werkzeuge* der derzeit verwendete ausgewählt werden. In unserem Fall ist dies das Board: „Arduino Mega 2560“




### USB-Port einstellen

Außerdem muss dem Programm ebenfalls in der Rubrik *Werkzeuge* mitgeteilt werden, an welchem USB-Port der Mikrocontroller angeschlossen ist. Hier muss man vielleicht durch Ausprobieren herausfinden, welches der richtige Port ist.

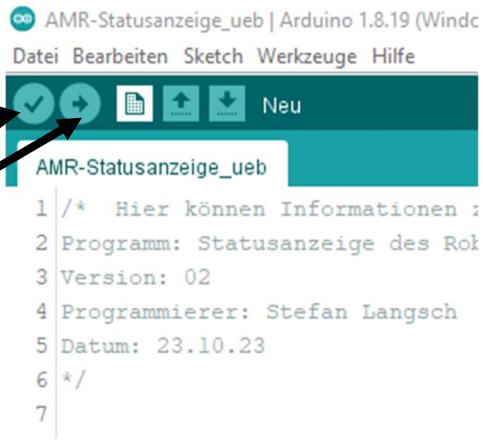




## Überprüfung und Übertragung des Programms

Bevor das Programm übertragen wird, sollte es auf Fehler überprüft werden.
Dazu klickt man einfach auf das Hakensymbol.

Ist das Programm fehlerfrei, kann es mit einem Klick auf den Pfeil auf den Mikrocontroller übertragen werden.




## Fehlersuche

Bei der Fehlersuche hilft die Programmierumgebung mit. In dem folgenden Beispiel wurde in Zeile 31 das Semikolon vergessen.

Im Editor wird die Zeile, in welcher der Fehler aufgefallen ist, hellrot markiert.

Im unteren Fensterbereich wird genauer beschrieben, was das Problem ist. In diesem Fall wird **vor (before)** digitalWrite(roteLED, HIGH); ein Semikolon **erwartet (expected)**.

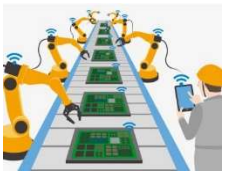
```
30  pinMode(grueneLED, OUTPUT);  //
31  pinMode(roteLED, OUTPUT)
32
33  digitalWrite(roteLED, HIGH);   // rote LED einschalten
34
35  for (int i=0; i<10; i=i+1){ //Start der for-Schleife
36    digitalWrite(grueneLED, HIGH); // grüne LED ausschalten
37    delay(500);                    //
```

```
expected ';' before 'digitalWrite'
```

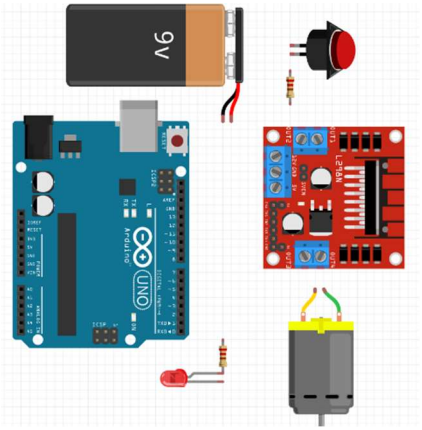
```
AMR-Statusanzeige_ueb:21:3: error: expected ';' before 'digitalWrite'

  digitalWrite(roteLED, HIGH);  // rote LED einschalten

  ^~~~~~~~~~~~

exit status 1

expected ';' before 'digitalWrite'
```

Bereits während der Programmierung wird es erleichtert, Schreibfehler zu erkennen. Bekannte Ausdrücke färbt der Editor automatisch. Im folgenden Beispiel ist der Befehl in Zeile 26 nicht orange geworden.
Was stimmt in der Zeile nicht?

```
23  for (int i=0; i<10; i=i+1){ //Start der for-Schleife
24    digitalWrite(grueneLED, HIGH); // grüne LED ausschalten
25    delay(500);                    //
26    digitalwrite(grueneLED, LOW);  //
27    delay(500);
28  }  //Ende der for-Schleife
```



## Verdrahtungsübung

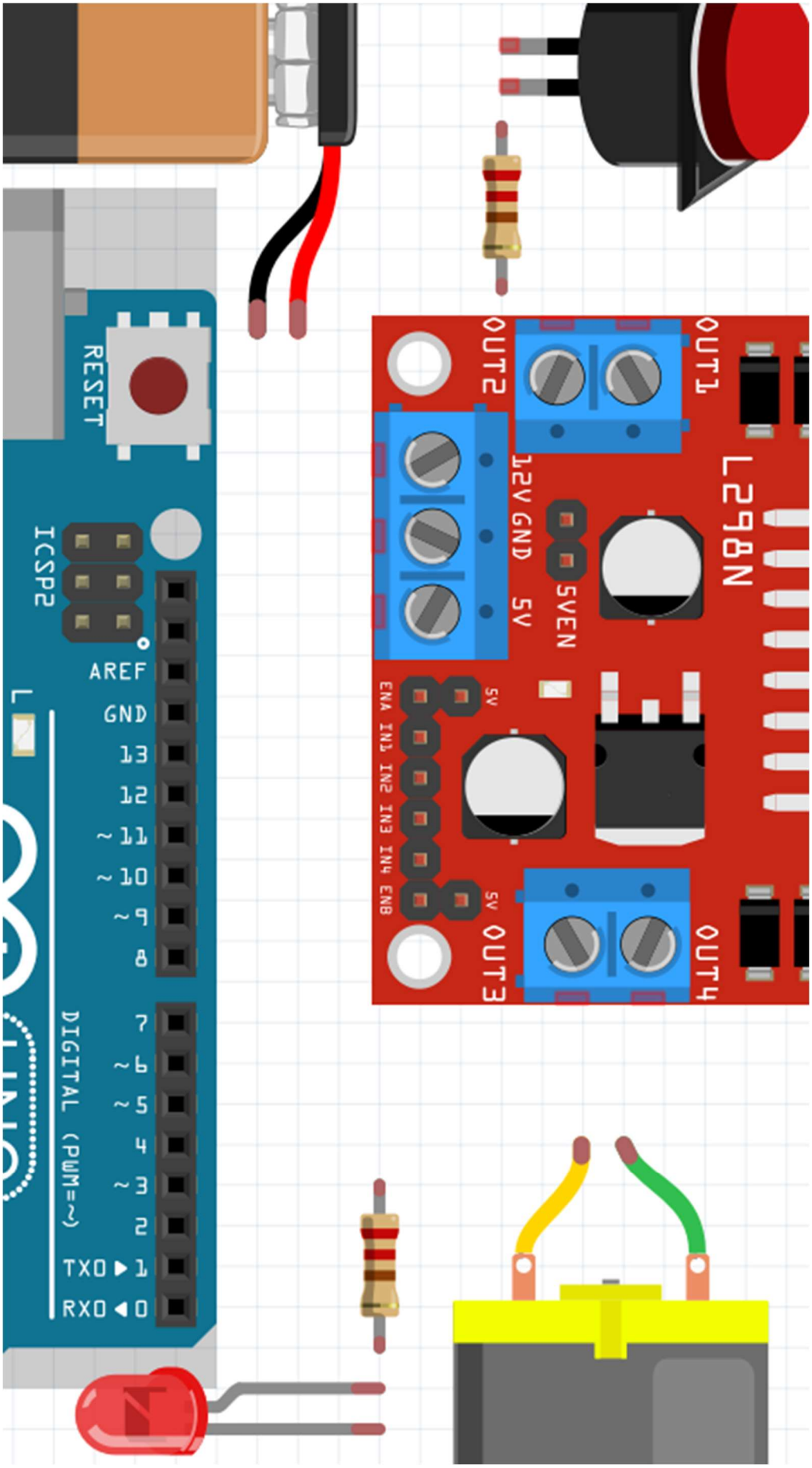


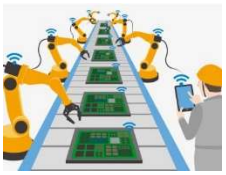

Ein Mikrocontroller soll die Steuerung eines Motors übernehmen. Immer wenn ein Taster betätigt wird, soll eine rote LED und ein Motor angehen. Lässt man den Taster los, so stoppt der Motor und die LED erlischt.

**Aufgaben:**

1. Ein Kollege von dir hat das zugehörige Programm erstellt und die Teile zurechtgelegt. Leider hat er es nicht mehr geschafft, die Schaltung in Betrieb zu nehmen und das Programm zu kommentieren. **Verdrahte** die Schaltung. Nutze dazu die Angaben im Mikrocontroller-Programm.
2. Ergänze den fehlenden Befehl in Zeile 26.

```
1  int in3 = 13;        //Motor an Pin 13
2  int in4 = 12;        //Motor an Pin 12
3
4  int led = 2;         //LED an Pin 2
5  int taster = 11;     //Taster an Pin 11
6  int tasterzustand;   //zum Speichern des Tasterzustands
7
8  void setup(){
9    pinMode(taster,       );  //wird als Eingang festgelegt
10   pinMode(led,          );
11   pinMode(in3,    OUTPUT);  //wird als Ausgang festgelegt
12   pinMode(in4,    OUTPUT);  //wird als Ausgang festgelegt
13 }
14
15 void loop(){
16   tasterzustand = digitalRead(taster);  //Einlesen des Tasters
17
18   if (tasterzustand == HIGH){  //Wenn der Taster gedrückt war...
19     digitalWrite(in3, LOW);    // Motor einschalten
20     digitalWrite(in4, HIGH);   // Motor einschalten
21     digitalWrite(led, HIGH);
22     }
23   else                         // wenn nicht...
24     digitalWrite(in3, LOW);    // Motor ausschalten
25     digitalWrite(in4, LOW);    // Motor ausschalten
26                                // LED ausschalten
27     }
28 }
```


RESET
ICSP2
AREF
GND
13
12
~ 11
~ 10
~ 9
8
7
~ 6
~ 5
4
~ 3
2
TX0 ▶ 1
RX0 ◀ 0
DIGITAL (PWM=~)
L
OUT2
OUT1
12V GND 5V
5VEN
L298N
ENA IN1 IN2 IN3 IN4 ENB
5V
OUT3
OUT4

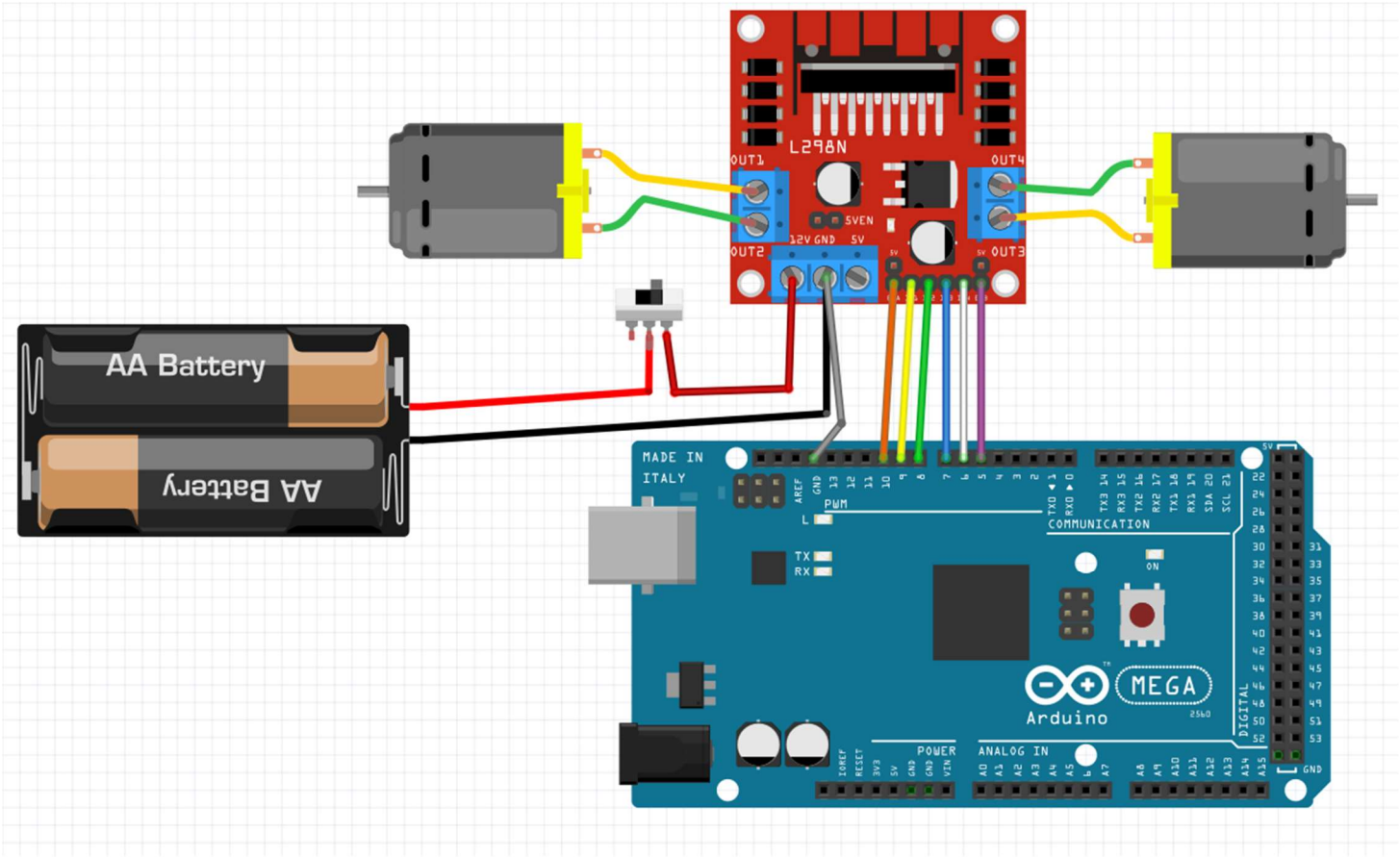


## Die erste Fahrt

Die Motoren müssen vom Mikrocontroller gesteuert werden. Die Leistung der Ausgänge des Mikrocontrollers reicht dafür aber nicht aus. Für die Leistungsanpassung zwischen Mikrocontroller und Motoren sorgt der Motortreiberbaustein L298N, der sich auf einer Anschlussplatine befindet.

**Motortestschaltung**

3. Stelle die abgebildeten Kabelverbindungen her (siehe auch Programmcode).
4. Schließe die Motoren an die Anschlüsse OUT1, OUT2 sowie an OUT3 und OUT4 an.



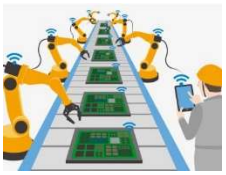

**Aufgaben:**

1. Übertrage das Programm von der nächsten Seite auf den Mikrocontroller.
2. Drehen sich die Motoren falsch herum? Notiere zwei Möglichkeiten, wie sich die Drehrichtung ändern lässt.
3. Verändere den Wert der Variablen „tempo“ in Zeile 24 auf Werte zwischen 0 und 255. Notiere den Wert, der dir für unseren AMR am sinnvollsten erscheint.

**Testprogramm**

```
// Gleichstrommotor 1
int GSM1 = 10;
int in1 = 9;
int in2 = 8;

// Gleichstrommotor 2
int GSM2 = 5;
int in3 = 6;
int in4 = 7;

int tempo = 180; //max. 255

void setup(){
  pinMode(GSM1, OUTPUT);
  pinMode(GSM2, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
}

void loop(){
  digitalWrite(in1, LOW);      // M1 vorwärts
  digitalWrite(in2, HIGH);
  analogWrite(GSM1, tempo);    // Geschwindigkeit = tempo

  digitalWrite(in3, HIGH);     // M2 vorwärts
  digitalWrite(in4, LOW);
  analogWrite(GSM2, tempo);    // Geschwindigkeit = tempo
  delay(2000);

  digitalWrite(in1, LOW);    // Stopp für 2 Sekunden
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  delay(2000);
}
```

Beschreibung und Beispielprogramme: https://funduino.de/nr-34-motoren-mit-h-bruecke-l298n-ansteuern (Das dort vorgestellte Beispiel weicht von dem obigen ab. Es ist etwas komplizierter. 😉)



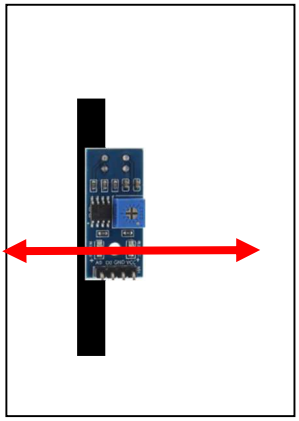
## Einstellen des Helligkeitssensors

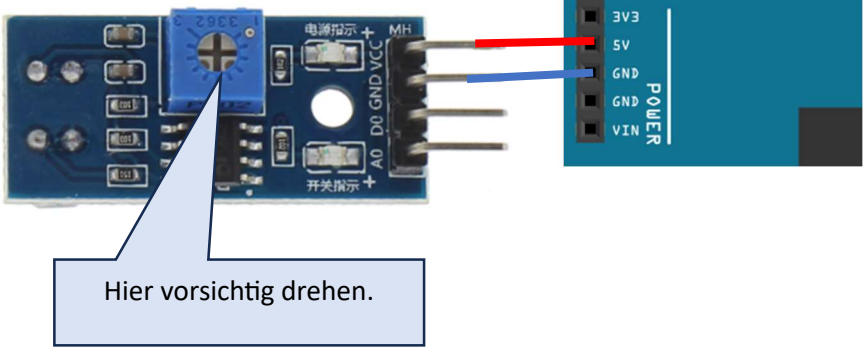
Mit der Sensorplatine soll die Linie auf der Fahrbahn erkannt werden. Direkt neben einem Lichtsensor ist eine Infrarot-LED eingebaut. Der Sensor misst die Stärke des reflektierten Lichtes. Der Wert kann über den Ausgang A0 der Platine abgefragt werden.

Für den Einsatzzweck an unserem Linienfolger wird dieser Wert nicht benötigt. Das Modul soll lediglich erkennen, ob es sich über einer schwarzen Linie oder dem hellen Boden befindet. Damit dies gelingt, muss der Sensor kalibriert werden. Dies geschieht über da

**Aufgaben:**

1. Stelle folgende Kabelverbindungen her: **VCC an 5V** und **GND an GND**.
2. Klebe einen Streifen schwarzes Isolierband auf ein weißes Blatt.
3. Halte den Sensor in genau dem gleichen Abstand über die schwarze Linie, wie er später auch über der Fahrbahn sein wird.
4. Bewege den Sensor seitlich über den Boden und die Linie.
   Es sollte eine grüne LED an und ausgehen.
5. Verhält sich die LED nicht wie gewünscht, so drehe etwas an dem Potentiometer und wiederhole dann Schritt 4 (und 5).






Hier vorsichtig drehen.

---

Beschreibung und Beispielprogramme: https://funduino.de/arduino-infrarot-abstandssensor

| | |
|---|---|

Mögliche Kursarbeit (mit Hallenplan)

## Jgst. 10 - Technik - Kursarbeit Nr. 1

| Name: | |
|---|---|

*In einer Produktionshalle werden Sahnetorten in mehreren automatisierten Prozessschritten hergestellt und verpackt. Als Techniker*in betreust du sowohl die Prozesslinien als auch die autonomen mobilen Roboter (AMR), die in der Halle eingesetzt werden.*

____ | P.

1. Als Techniker*in betreust du zwei Schüler*innen, die ihr dreiwöchiges Betriebspraktikum in der Firma machen. Du gibst ihnen die Aufgabe, einen kleines Lagerrobotermodell mit zwei Getriebemotoren zu bauen.

a) **Erstelle** für die Praktikant*innen eine Schaltzeichnung zur Ansteuerung der Motoren, welche die folgenden Bedingungen erfüllt:

- Es gibt einen Hauptschalter, mit dem das Robotermodell ein- und ausgeschaltet werden kann.
- Jeder Motor lässt sich einzeln über einen Taster ein- und ausschalten.
- Es gibt einen Taster, der bei Betätigung beide Motoren stoppt.

b) Als die Praktikanten mit dem Aufbau des Roboters fertig sind und diesen einschalten, passiert Folgendes.

- Ein Roboter fährt gar nicht. Allerdings werden die Batterien sehr heiß.
- Der andere Roboter scheint zunächst zu funktionieren, doch als er auf den Tisch gestellt wird, dreht er sich nur um seine eigene Achse im Kreis, wobei beide Motoren funktionieren.

**Nenne** mögliche Ursachen und **notiere** Tipps zur Behebung der Probleme.

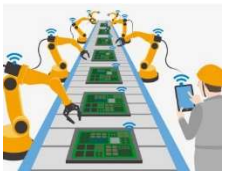
____ | P.

2. Dein Chef hatte zunächst die Sorge, dass es zu Unfällen mit den Robotern kommen könnte. Du konntest ihn aber schnell beruhigen. **Erkläre** wie bei dem Roboter mit einem redundantem* System schwere Unfälle vermieden werden sollen.

____ | P.

3. Deine Firma will expandieren und plant eine neue Produktionshalle. Als Experte fütr die beiden AMR, die dort eingesetzt werden sollen, hast du die Aufgabe, die Halle und die Wege der Roboter zu planen. Als Vorgabe erhältst du einen Plan der Halle (siehe Anlage M1).

a) **Ergänze** in dem Plan ein Leergutlager, ein Lager für die fertigen Torten und Ladestationen für die Roboter.
b) Die Roboter sind Linienfolger. **Zeichne** die Fahrwege der Roboter in den Hallenplan ein.
c) **Erkläre**, wie es gelingen kann, dass der Roboter die einzelnen Prozesslinien anfahren kann.
d) Überlege dir, wie der Roboter erkennen kann, dass er an einem Lager oder einer Prozesslinie angekommen ist. **Notiere** deine Idee.
*e) (Zusatzaufgabe) Überlege dir, wie der Roboter unterscheiden kann an welcher Quelle oder Senke er gerad im Lager steht.*

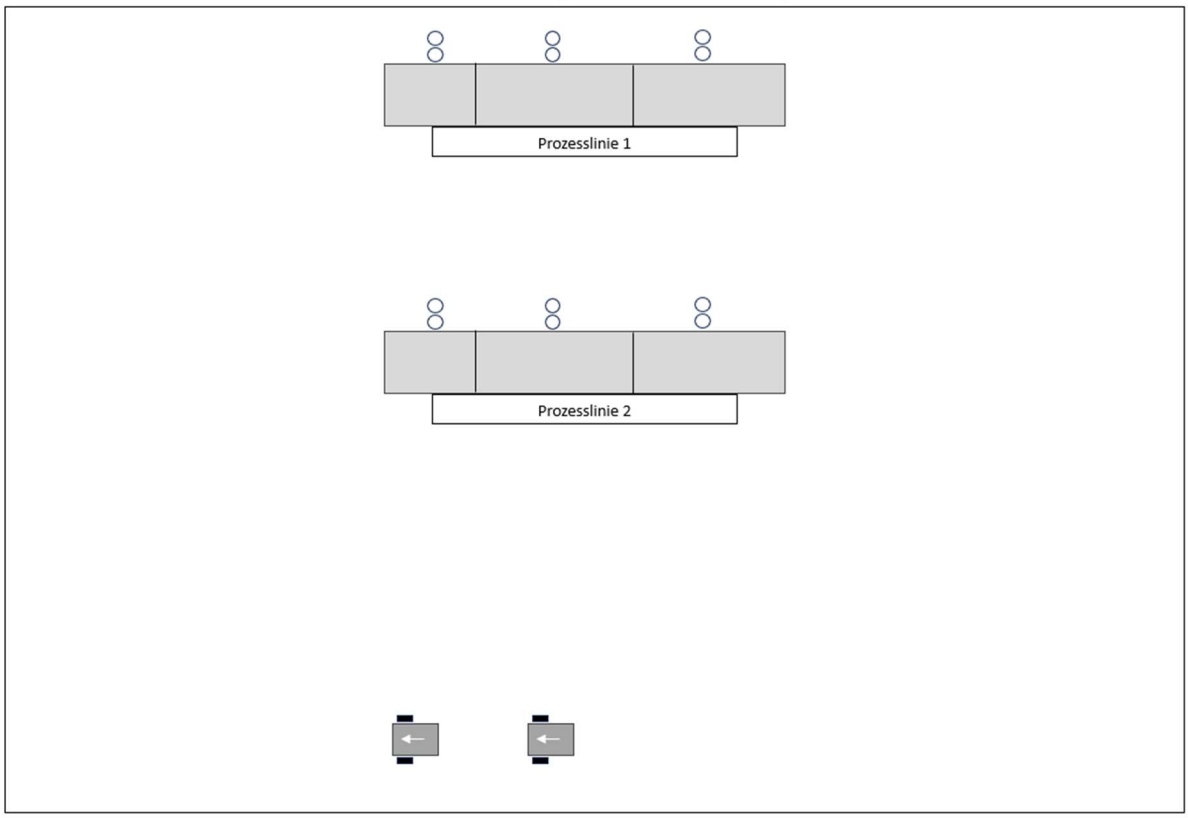


| | |
|---|---|
| | |

____ | P.

5. Als die beiden Praktikanten sich die Prozesslinie 1 anschauen, kommt es zu einem Ausfall des Prozesschrittes 2, was durch eine rote Lampe über dem Prozesschritt angezeigt wird. Die Praktikanten wundern sich, dass die Prozessschritte 1 und 3 einfach weiterlaufen. **Erkläre**, wie dies möglich ist und warum es sinnvoll ist.

Material M1: Hallenplan: Cheesecake-Factory

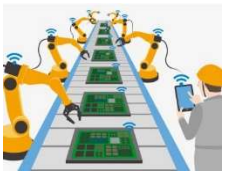



____ | P.

Darstellungsleistung:

**Summe:**

____ | P.



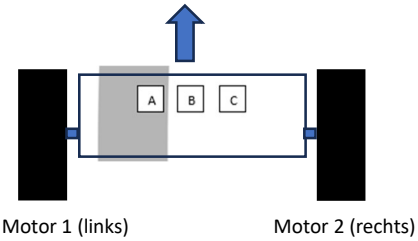
## Fahren auf der Linie

### Kontrollstruktur if-Anweisung

Die Signale der Lichtsensoren werden zunächst eingelesen und in den Variablen LSA, LSB und LSC gespeichert: LSA = digitalRead(sensorA);
Ein Sensor liefert über einer schwarzen Linie ein LOW-Signal und über der weißen Fläche ein HIGH-Signal. Aber wie müssen die Motoren auf die Signale reagieren?
Man kann in if-Anweisungen auch mehr als eine Bedingung abfragen. Dabei handelt es sich im Prinzip um logische Verknüpfungen, wie sie aus der Digitaltechnik bekannt sind. Jede einzelne Bedingung steht dabei in einer Klammer.

Dabei steht

- && für eine UND-Verknüpfung
- || für eine ODER-Verknüpfung

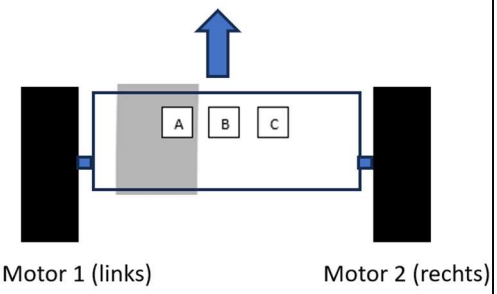



<u>Beispiele:</u>

```
if ( (taster1==HIGH) && (taster2==HIGH) )     //beide Taster müssen gedrückt sein

if ( (taster1==HIGH) || (taster2==HIGH) )     //es reicht, wenn einer der Taster gedrückt ist
```

### Fahranweisungen für den AMR

Bevor der Programmcode geschrieben wird, ist es hilfreich, zunächst in Worten zu formulieren, was geschehen soll. **<u>Aufgabe</u>**: Ergänze die Angaben in den Boxen.




**<u>AMR fährt geradeaus</u>**

Wenn Sensor A ein LOW-Signal liefert <u>und</u> Sensor B ein HIGH-Signal liefert,

dann soll Motor 1 und Motor 2 laufen.

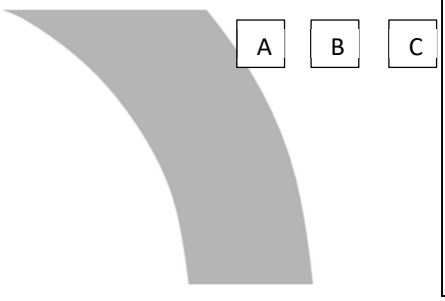
```
if ( (LSA==LOW) && (LSB==HIGH) ) {
  M1 einschalten;  // Hier wird später der passende Befehl eingesetzt.
  M2 einschalten;  // Hier wird später der passende Befehl eingesetzt.
}
```

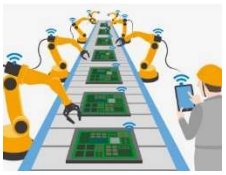
**<u>AMR fährt linke Kurve</u>**

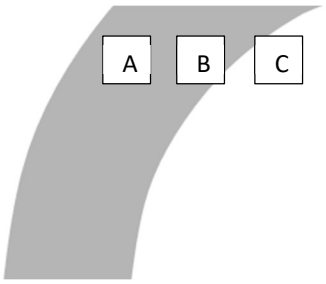
Wenn Sensor A ein _____-Signal liefert <u>und</u> Sensor B ein _____-Signal liefert,

dann soll Motor 1 _____ und Motor 2 _____.

```
if ( (LSA==_____) && (LSB==_____) ) {
  M1 _____schalten;
  M2 _____schalten;
}
```





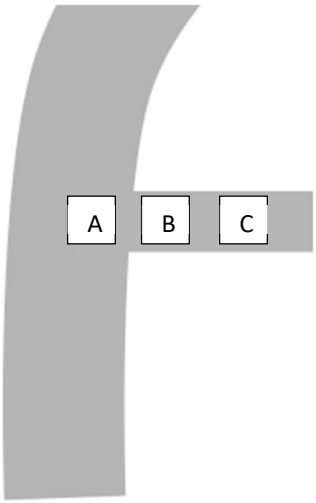



**AMR fährt rechte Kurve**

Wenn Sensor A ein _____-Signal liefert und Sensor B ein _____-Signal liefert,

dann soll Motor 1 _____ und Motor 2 _____.

```
if ( (LSA==_____) && (LSB==_____) ) {
  M1 _____schalten;
  M2 _____schalten;
}
```

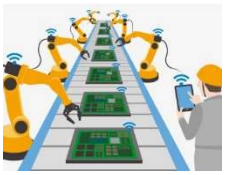



**AMR stoppt**

Wenn Sensor A ein _____-Signal liefert und Sensor B ein _____-Signal liefert

und Sensor C ein _____-Signal liefert,

dann soll Motor 1 _____ und Motor 2 _____.



## Ultraschallsensor

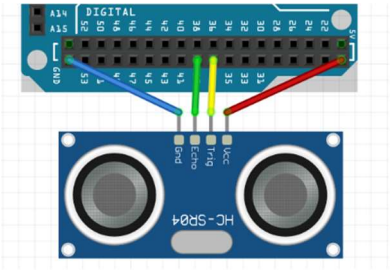
Der Ultraschallsensor soll an dem AMR eingesetzt werden, um Hindernisse zu erkennen. Wie dies funktioniert, kann mit einem einfachen Aufbau erprobt werden. Hierfür wird die interne LED des Mikrocontroller-Boards genutzt, die mit Pin 13 verbunden ist.

**Funktionsprinzip des Sensors**

Auf der Sensorplatine befindet sich ein Ultraschallsender und ein Ultraschallempfänger. Im Prinzip wird die Zeit ermittelt, die benötigt wird, bis das Echo wieder empfangen wird.

**Testprogramm**

Sobald sich ein Gegenstand näher als 15cm vor dem Sensor befindet, soll die interne LED angehen. Wird der Gegenstand entfernt, erlischt die LED wieder.

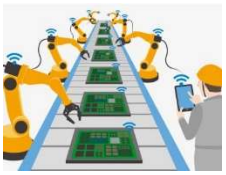



**Aufgaben:**

1. Stelle folgende Kabelverbindungen her:
   VCC an 5V; END an GND; Echo an Pin 39; Trig an Pin 37
2. Übertrage das Testprogramm auf den Mikrocontroller und beobachte, wie sich die interne LED verhält, wenn etwas vor den Sensor gehalten wird.
3. Ändere in Zeile 21 den Wert in der „if-Abfrage“ und erkunde die Auswirkungen.
4. Ist Zeile 28 notwendig? Kann die Pause auch kleiner sein? Was ist das Problem an einer langen Pause?

```
1  int trigger=37; //Trigger-Pin an Pin37 des Arduino-Boards
2  int echo=39; // Echo-Pin an Pin39 des Arduino-Boards
3  long dauer=0; // Speichervariable für die Laufzeit
4  long entfernung=0; // Speichervariable für die berechnete Entfernung.
5
6  void setup(){
7  pinMode(trigger, OUTPUT); // Trigger-Pin ist ein Ausgang
8  pinMode(echo, INPUT); // Echo-Pin ist ein Eingang
9  pinMode(LED_BUILTIN, OUTPUT); //eingebaute LED ist Ausgang
10 }
11
12 void loop(){
13 digitalWrite(trigger, LOW); //Ton ausschalten
14 delay(5); // Pause für 5 Millisekunden
15 digitalWrite(trigger, HIGH); //Jetzt sendet man eine Ultraschallwelle los.
16 delay(10); //Dieser „Ton“ erklingt für 10 Millisekunden.
17 digitalWrite(trigger, LOW);//Dann wird der „Ton“ abgeschaltet.
18 dauer = pulseIn(echo, HIGH); //Zeitmessung in ms
19 entfernung = (dauer/2) * 0.03432; //Berechnung der Entfernung
20
21 if (entfernung > 15 ){ //Wenn die Entfernung über 15cm ist...
22   digitalWrite(LED_BUILTIN, LOW); //LED ausschalten
23   }
24 else {                          // Ansonsten..
25   digitalWrite(LED_BUILTIN, HIGH);// LED einschalten
26   }
27 delay(1000); //Pause für 1 Sekunde.
28 }
```

Beschreibung und Beispielprogramme: https://funduino.de/nr-10-entfernung-messen



| | |
|---|---|

## Farbsensor (in Bearbeitung)

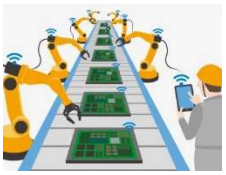

Farbige Markierungen auf der Fahrbahn können dem Roboter bei der Orientierung in der Lagerhalle nutzen.

So könnte er beim Erkennen einer roten Fahrbahnmarkierung z.B. 30 Sekunden warten, bis seine Ladung entnommen wurde.

Eine blaue Markierung verrät ihm vielleicht, dass bei der nächsten Abfahrt auf der linken Seite eine Ladebox erreicht werden kann.

Beschreibung und Beispielprogramm https://funduino.de/nr-07-farbsensor-am-arduino

| | |
|---|---|



## RFID (in Bearbeitung)

Der Lagerroboter soll anhand der Ladung erkennen, zu welcher Prozesslinie er fahren muss. Um dies zu erreichen, kann ein RFID-Chip genutzt werden.
Der Roboter liest den Chip an der zu transportierenden Ladung aus und entscheidet dann, ob er an der rechten Kante oder an der linken Kante entlangfahren muss.


Beschreibung und Beispielprogramme: https://funduino.de/nr-18-rfid-kit

## Rückblick und Ausblick

An einigen Stellen wurde von der ursprünglichen Planung abgewichen. So wurden die Motoren z.B. nicht über Transistoren angesteuert. Auch wurde keine Platine mit Reflexkopplern erstellt. Sowohl bei der Motoransteuerung als auch bei der Linienerkennung werden nun Platinen aus dem Handel genutzt.

Noch sind die Fahrzeuge nicht fertiggestellt. Nach Beendigung des Projektes werden die Roboterfahrzeuge, wie in der Projektidee beschrieben, nicht alle über die gleiche Funktionalität verfügen.

Wenn man die Schüler*innen bei der Arbeit beobachtet, erkennt man sehr schnell, dass alle mit viel Engagement bei der Sache sind. Eine Evaluation wurde allerdings noch nicht durchgeführt.

Die Fahrzeuge verbleiben nach Fertigstellung im Fachraum und stehen dann den Kursen der folgenden Schuljahre zu Verfügung. Langfristiges Ziel ist die Simulation einer kleinen Fabrik mit Roboterarm, Fließband mit Stanz- oder Abfüllmöglichkeit und Lagerroboter.

Mit den Fahrzeugen lässt sich aber sicherlich auch gut für die Wahl des Faches Technik werben.