

# SDP - Regelwerk

## 1. Generelle Regeln und Informationen

- Es muss selbständig ein autonomer LEGO Mindstorms-Roboter entwickelt werden, der einer schwarzen Linie auf weißem Untergrund folgt und dabei verschiedene Aufgaben bewältigt (z. B. Erkennen von Barcodes, Anhalten vor einer Schranke, Aufnahme und Abgabe eines Balls).
- Die Steuerung soll über einen Regelalgorithmus erfolgen (z. B. PID-Regler, oder einem ähnlichen Regler), der auf kontinuierlichen Sensordaten basiert.
- Der abgegebene Code und der Zwischenbericht werden mit Software auf Plagiate überprüft.
- Die Dokumentation des Algorithmus (PID-Regler, Kalibrierlogik, Schwellenanpassung) ist Teil der Bewertung.

## 2. Projektaufgaben und Wettbewerbsregeln

- Das gesamte Projekt wird in Einzelschritte („Challenges“) gegliedert, die sowohl während der Entwicklungsphase (Meilensteinprüfung) als auch im Abschlusswettbewerb bewertet werden.
- Beim Wettbewerb müssen alle Aufgaben innerhalb einer maximalen Fahrzeit von 4 Minuten erfüllt werden.
- Der Wettbewerb gilt als bestanden, sobald die zweite Lichtschranke durchquert wurde und der Roboter über eine funktionsfähige Ballaufnahme- und Ballabgabevorrichtung verfügt.

Eine funktionsfähige Ballaufnahme ist definiert als eine Vorrichtung, die in der Lage ist, einen von oben fallenden Ball mit einem Durchmesser von 5,2 cm zuverlässig aufzunehmen.

Eine funktionsfähige Ballabgabe ist definiert als eine Vorrichtung, die den aufgenommenen Ball am Ziel gemäß den in Abschnitt 2.7 beschriebenen Vorgaben, selbstständig in den Korb befördern kann. Hierfür müssen sowohl die erforderliche Mechanik als auch die entsprechende Steuerungssoftware implementiert und betriebsbereit sein.

- Manuelle Eingriffe, externe Steuerung oder Nachjustieren während der Fahrt sind nicht gestattet.
- Jeder Controller darf nur mit den zugehörigen Motortypen betrieben werden (EV3-Controller mit EV3-Motoren, NXT-Controller mit NXT-Motoren), da die interne Übersetzung der Motoren verschieden ist.

### 2.1 Gesamtziel (bis zum Wettbewerb)

**Der Roboter soll in der Lage sein:**

- einer schwarzen Linie auf einer weißen Fahrbahn zu folgen, siehe Abschnitt 2.2.
- im Bereich zwischen der ersten Lichtschranke, beim Wettbewerb markiert durch einen Laserlinie, und der Wand selbständig zu wenden, siehe Abschnitt 2.3.
- vor einer Schranke anzuhalten, während er mit einem LEGO-Ball beladen wird, siehe Abschnitt 2.4.
- den Ball in seiner Auffangvorrichtung über den ganzen Parkour hinweg sicher zu transportieren. Dies schließt Steigungen und Gefälle ein. Wenn der Ball bei der Aufnahme oder während der Fahrt herausfällt oder abgeworfen wird, kann der letzte Checkpoint nicht erreicht werden, siehe Abschnitt 2.4.
- ein Objekt (Klotz) von der Fahrbahn schieben, siehe Abschnitt 2.5.
- Tunnel, Gefälle und Steigungen zu meistern, siehe Abschnitt 2.6.
- den Ball am Ende in einen Korb zu befördern, siehe Abschnitt 2.7.

**Generelle Informationen:**

- Die Aufgaben können in beliebiger Reihenfolge programmiert, müssen jedoch in einem ununterbrochenen Fahrzyklus ausgeführt werden.
- Die Aufgaben können im Code in einer festen Reihenfolge abgearbeitet werden.
- Die Streckenparameter und Umgebungsbedingungen sind festgelegt, werden jedoch nicht exakt identisch in jedem Durchlauf sein, der Roboter muss somit robust gegenüber Variationen sein.

## 2.2 Challenge 1 – Linienverfolgung

**Ziel:**

Der Roboter muss eine schwarze Linie auf weißem Untergrund selbstständig erkennt und ihr über den gesamten Parkour sicher folgt. Die Linienerkennung erfolgt dabei gemäß den in Kapitel 3 definierten Regeln zur dynamischen Threshold-Bestimmung und Toleranzhandhabung.

**Aufbau der Bahn und Startbedingungen:**

Die Wettbewerbsbahn besteht aus einer weißen Oberfläche mit einer schwarzen Linie als Fahrspur.

Die Bahn ist so gestaltet, dass unterschiedliche Kurvenradien, Helligkeitsbedingungen und Unterbrechungen die Stabilität der Regelung prüfen.

**Bahnparameter:**

- Fahrbahnbreite: mindestens 40 cm mit mittiger schwarzer Linie
- Linienbreite:  $5,0 \pm 0,5$  cm
- Kleinster Kurvenradius: 10 cm
- Streckenlänge: ca. 15 m
- Unterbrechungen: mehrmals Lücken auf Geraden in der Linie (ca. 6 cm lang)
- Umgebungsbedingungen: wechselnde Lichtverhältnisse (Sonne, Tunnel, Reflexionen)

**Bewertungskriterien:**

- Der Roboter darf die Linie kurzzeitig verlassen oder Wände berühren, sofern er selbstständig zur Linie zurückfindet.
- Die Bewertung erfolgt anhand der Stabilität der Linienverfolgung, der Zuverlässigkeit bei Unterbrechungen und der Robustheit gegenüber wechselnden Lichtverhältnissen.
- Die korrekte Anwendung der dynamischen Threshold-Methode und der methodischen Toleranzbereiche aus Kapitel 3 ist Teil der Bewertung.

**2.3 Challenge 2 – Fahrtrichtung ändern****Ziel:**

Nach Überqueren der Startlichtschranke soll der Roboter die Fahrtrichtung umkehren.

**Ablauf:**

- Der Roboter startet entgegen der Fahrtrichtung.
- Die Zeitmessung beginnt, sobald die erste Lichtschranke durchfahren wird.
- Nach der Lichtschranke folgt ein gerades Stück, das 20 cm vor einem Hindernis endet.
- Der Roboter muss hinter der Lichtschranke selbstständig wenden oder rückwärtsfahren, um die Strecke in Gegenrichtung fortzusetzen.
- Es wird nicht empfohlen, das Drehen zeit- oder distanzbasiert zu realisieren (z. B. „10 cm fahren, dann drehen“), da die Startposition im Wettbewerb variieren kann.  
→ Stattdessen soll eine sichere Erkennung des Referenzbalkens mithilfe eines Sensors erfolgen.
- Das Hindernis darf berührt, aber nicht verschoben werden.

**2.4 Challenge 3 – Warten und Beladen****Ziel:**

Der Roboter muss vor einer Schranke anhalten, bis sie sich öffnet, und währenddessen beladen werden. Nachdem sich die Schranke öffnet, soll der Roboter selbstständig weiterfahren und den Ball sicher transportieren.

**Details:**

- Schrankenhöhe:
  - Oberkante: 10 cm
  - Unterkante: 1 cm
  - Öffnung: 9 cm
- Der Roboter darf die Schranke nicht berühren.
- Während der Wartezeit wird er mit einem LEGO-Ball ( $\varnothing = 5,2$  cm) beladen.
- Der Abwurfpunkt für den Ball liegt 20 cm hinter der Schranke, in 30 cm Höhe.

- Die Wartezeit, vor der geschlossenen Schranke, ist zufällig und zählt nicht in die Zeitmessung mit rein.

## 2.5 Challenge 4 – Objekt wegschieben

### Ziel:

Der Roboter soll ein Objekt (Holzklotz) von der Bahn schieben, bis es vollständig in einem Behälter im Boden verschwindet.

### Details:

- **Klotzgröße:**  $20 \times 7 \times 7 \text{ cm}^3$
- **Rampe:** maximal  $15^\circ$  Steigung
- **Positionierung:** Der Klotz befindet sich **immer** rechts der Hauptstrecke.
- **Erkennung:** Die T-Kreuzung ist durch drei weiße Lücken (je 2 cm breit) gekennzeichnet.

### Bewertung:

Der Roboter darf den Klotz schieben, aber nicht werfen oder fallen lassen. Das Objekt gilt als „erfolgreich bewegt“, wenn es vollständig im Behälter liegt.

## 2.6 Challenge 5 – Tunnel und Steigung

### Ziel:

Der Roboter soll sicher durch einen Tunnel fahren und Steigungen sowie Gefälle bewältigen.

### Rahmenbedingungen:

- **Tunneleingang:**
  - Breite: mindestens 40 cm
  - Höhe: mindestens 25 cm
  - Mit Gefälle am Eingang
- **Steigung / Gefälle:**
  - maximal  $\pm 30^\circ$

### Bewertung:

Der Roboter muss stabil fahren, darf weder kippen noch stehen bleiben. Ein kurzzeitiges Abweichen von der Linie ist erlaubt, sofern die Orientierung wiederhergestellt wird.

## 2.7 Challenge 6 – Ball in Korb werfen

### Ziel:

Der Ball muss am Ziel von dem Roboter in einen Korb befördert werden.

### Details:

- **Korbposition:** Mittig auf der Fahrbahn

- **Markierung:** Schwarzer Querbalken 20 cm vor dem Korb
- **Korbabmessungen:**  $16 \times 16 \times 12 \text{ cm}^3$
- **Kontakt:** Der Roboter darf den Korb berühren, jedoch nicht verschieben.

**Bewertung:**

Der Ball muss durch den Roboter im Korb landen. Ein Fehlschuss gilt als unvollständig, führt aber nicht zur Disqualifikation.

## 2.8 Meilenstein-Aufgaben (Zwischenprüfung)

Vor dem Abschlusswettbewerb müssen die folgenden Grundfunktionen im Meilenstein auf der Testbahn nachgewiesen werden:

1. Zuverlässige Linienverfolgung. Dies wird gezeigt, in dem der Roboter eigenständig den vorgegebenen Parkour, gemäß den Vorgaben in Abschnitt 2.2, vollständig abfährt.
2. 180°-Wendung am Start (nach der Laserlinie / Lichtschranke) gemäß den Vorgaben in Abschnitt 2.3.
3. Durchfahren eines Tunnels, gemäß den Vorgaben in Abschnitt 2.6.
4. Robuster Betrieb bei wechselnden Lichtverhältnissen, gemäß den Vorgaben in Abschnitt 2.2.
5. Vorhandene, aber noch nicht verpflichtend funktionsfähige Ballaufnahmevorrichtung, gemäß den Vorgaben in Abschnitt 2.1.
6. Bewältigung von Steigung und Gefälle, gemäß den Vorgaben in Abschnitt 2.6.

Die Ballwurffunktion muss erst zum Abschlusswettbewerb funktionsfähig sein.

## 3. Sensorkalibrierung

### 3.1 Grundprinzip

- Sensoren dürfen kalibriert werden, um Offset-Fehler zwischen den Sensoren zu kompensieren. Dabei kann eine sogenannte Offset-Korrektur angewendet werden. Sie bedeutet, dass von jedem Messwert eines Sensors eine feste Zahl addiert oder subtrahiert wird, um systematische Unterschiede zwischen den Sensoren auszugleichen.

**Beispiel:** Wenn alle Sensoren auf dieselbe weiße Fläche zeigen, aber ein Sensor 255 misst und ein anderer 250, kann der zweite Sensor durch eine Offset-Korrektur von +5 angeglichen werden, sodass beide denselben Referenzwert liefern.

- Diese Kalibrierung darf zu Beginn des Laufs durchgeführt werden (z. B. während einer Kalibrierungsphase vor dem Start).

### 3.2 Dynamische Abweichung

- Auch nach der Kalibrierung können kleine Restabweichungen zwischen den Sensoren bestehen.

- Diese dürfen durch ein statisches Toleranzintervall (z. B.  $\pm 5\%$ ) ausgeglichen werden, um geringfügige Unterschiede in der Sensorempfindlichkeit zu kompensieren.

### 3.3 Erlaubte und verbotene Methoden

- **Nicht erlaubt:** Speichern fester „Schwarz-“ oder „Weißwerte“ (Hardcoding fester Sensorwerte, z. B.  $\text{threshold} = 45$ ).
- **Erlaubt:** Dynamische oder adaptive Ermittlung der Referenzwerte während der Fahrt. Da sich während des Wettbewerbs die Lichtverhältnisse stark verändern können (z. B. Tunnel, Sonnenflecken, Reflexionen, tiefstehende Sonne), müssen die Verfahren zur Schwellenbestimmung robust gegenüber kurzfristigen Änderungen sein.

### 3.4 Mögliche Verfahren zur dynamischen Bestimmung von Schwarz-/Weiß-Grenzen

#### (a) Laufende Mittelwertbildung (zeitlich begrenzt)

Jeder Sensor speichert ein gleitendes Mittel über die letzten  $n$  Messwerte (z. B. der letzten 20 – 50 ms).

→ Vorteil: Kurzzeitige Schwankungen (z. B. durch Flackern oder Glanz) werden geglättet.

→ Nachteil: Langsame, dauerhafte Lichtänderungen werden nur verzögert erkannt.

#### (b) Zeitlich begrenzte Min-Max-Kalibrierung

Während der Fahrt werden pro Sensor lokale Minimal- und Maximalwerte gespeichert, die sich über ein definiertes Zeitfenster oder Wegsegment hinweg aktualisieren (z. B. letzte 3 Sekunden).

→ Vorteil: Passt sich langsam verändernden Lichtverhältnissen an (z. B. Einfahrt in Schatten).

→ Nachteil: Bei extrem schnellen Wechseln (z. B. Tunnel-Einfahrt) kann kurzzeitig Fehlinterpretation entstehen.

#### (c) Dynamischer Threshold mit Begrenzung

Der aktuelle Schwellenwert wird als:

$$\text{threshold} = \frac{\text{aktuelles\_Maximum} + \text{aktuelles\_Minimum}}{2}$$

Die Maximal- und Minimalwerte müssen dabei zeitlich oder ereignisbasiert begrenzt werden (z. B. nur die letzten  $x$  gültigen Schwarz- und Weißmessungen).

→ So wird verhindert, dass einzelne Ausreißer (Reflexion, Schatten) die zukünftige Berechnung verfälschen.

#### (d) Verhältnisbasierte Auswertung (robust bei Lichtwechsel)

Statt absolute Werte zu vergleichen, werden die Sensorwerte relativ zum Durchschnitt aller Sensoren bewertet:

$$\text{Abweichung}_i = \frac{\text{Wert}_i - \text{Mittelwert}_{\text{aller Sensoren}}}{\text{Mittelwert}_{\text{aller Sensoren}}}$$

Um Messrauschen zu vermeiden, wird eine Toleranzschwelle  $T_{tol}$  definiert (z. B.  $\pm 5\%$ ). Nur wenn  $|\text{Abweichung}_i| > T_{tol}$ , gilt der Sensor als deutlich heller oder dunkler. Dadurch bleibt der Algorithmus unempfindlich gegenüber schwankenden Lichtverhältnissen und Sensorrauschen.

→ Vorteil: Automatische Anpassung an globale Helligkeitsänderungen.

→ Nachteil: Bei sehr ungleichmäßiger Beleuchtung über die Sensoren hinweg (z. B. eine Seite im Schatten) kann Fehlinterpretation auftreten.

### 3.5 Schwellen- und Toleranzdefinition

Auch wenn feste („hardcodierte“) Werte grundsätzlich zu vermeiden sind, dürfen in bestimmten Fällen feste Start- oder Sicherheitswerte für Thresholds oder Toleranzbereiche verwendet werden. Diese müssen jedoch methodisch begründet sein und sich während der Laufzeit automatisch anpassen können.

#### Begriffserklärungen

- **Threshold (Schwellenwert):**

Ein Grenzwert, ab dem ein Sensorwert als „schwarz“ oder „weiß“ interpretiert wird.

*Beispiel:* Wenn der Threshold 50 % beträgt, werden alle Messwerte unterhalb von 50 % als „schwarz“ gewertet, alle darüber als „weiß“.

- **Toleranzbereich:**

Ein erlaubter Abweichungsbereich, der kleine Messfehler oder Unterschiede zwischen den Sensoren ausgleicht.

*Beispiel:* Wenn der Toleranzbereich  $\pm 5\%$  beträgt, werden Werte zwischen 45 % und 55 % als gleich interpretiert.

#### Zulässige feste Werte

- **Initiale Thresholds:**

Ein fester Startwert (z. B. 50 %) darf zu Beginn der Fahrt gesetzt werden, muss aber im weiteren Verlauf regelmäßig aktualisiert oder überprüft werden.

Das reine Setzen eines festen Thresholds ohne spätere Anpassung oder Nachführung gilt als unzulässig (Hardcoding).

- **Kalibrier-Toleranzen (statische Toleranzen):**

Diese Toleranzen dienen dazu, Sensorrauschen oder kleine Restabweichungen nach der Kalibrierung zu kompensieren.

Sie sind immer zulässig, müssen nicht dynamisch angepasst werden und gelten nicht als Hardcoding.

- **Methodische Toleranzen (z. B. in 3.4 d):**  
Innerhalb dynamischer Verfahren (wie der verhältnisbasierten Auswertung) dürfen Toleranzen ebenfalls definiert werden, um Bewertungsgrenzen zwischen Sensoren festzulegen. Wichtig ist, dass die Methode den Threshold innerhalb des Toleranzbereichs dynamisch verschieben kann, um sich an veränderte Sensorbedingungen anzupassen, wie in Methode 3.4d. Diese Toleranzwerte können statisch oder dynamisch sein, müssen aber methodisch nachvollziehbar eingesetzt werden.
- **Kalibrierungskonstanten:**  
Werte, die im Rahmen einer Offset-Kalibrierung gewonnen wurden, gelten nicht als Hardcoding, da sie auf realen Messdaten beruhen.

### Dynamische Bestimmung von Thresholds oder Vertrauensbereichen

Um sich ändernde Lichtverhältnisse (z. B. Tunnel, Reflexionen, Sonnenflecken) auszugleichen, sollte der Threshold oder Vertrauensbereich **laufend an aktuelle Messbedingungen angepasst** werden.

Mögliche Verfahren:

1. **Gleitendes Fenster (Rolling Window):**  
Nur die letzten  $n$  erkannten Weiß- und Schwarzwerte werden zur Berechnung des aktuellen Thresholds verwendet.  
*Beispiel:*
  - Speichere die letzten 10 Werte, die als „weiß“ klassifiziert wurden.
  - $\text{Threshold} = (\text{Mittelwert Weiß} - \text{Mittelwert Schwarz}) / 2$
2. **Adaptive Mittelwertbildung:**  
$$\text{Threshold} = \alpha \times \text{alter Threshold} + (1 - \alpha) \times \text{neuer Messwert}$$
  
mit  $0 < \alpha < 1$  als Glättungsfaktor (z. B. 0,9).  
→ Der Threshold passt sich langsam an neue Lichtbedingungen an und vermeidet sprunghafte Änderungen.
3. **Lokaler Vergleich:**  
Statt eines globalen Schwellenwerts pro Roboter kann pro Sensormodul oder Sensorreihe ein eigener lokaler Threshold berechnet werden, um Teilabschattungen oder Reflexionen besser auszugleichen.

## 4. Regelung und Programmstruktur

### 4.1 PID-Regler

- Es ist ein Reglerprinzip (PID oder ähnliche Regler) zu verwenden.

- Parameter sollen so eingestellt werden, dass der Roboter stabil, schnell und nicht schwingend fährt.
- Eine reine On-Off-Steuerung ist nicht zulässig.

## 4.2 Code-Struktur

- Der Code muss klar strukturiert und kommentiert sein. Zum Beispiel müssen alle Funktionen einen Kommentar besitzen, der den groben Funktionsumfang einer Funktion beschreibt.
- Funktionen sollen logisch getrennt sein (z. B. `readSensors()`, `calculateError()`, `updatePID()`, `setMotorSpeed()`).

## 5. Hardcoding

### 5.1 Definition

Hardcoding bezeichnet das feste Eintragen konkreter Werte oder Entscheidungen im Programmcode, anstatt diese während der Laufzeit durch Sensoren, Berechnungen oder Parameter zu bestimmen.

### 5.2 Grundsatz

Das Hardcoding fester Werte für Sensorgrenzen, Farbschwellen oder Entscheidungslogik ist nicht zulässig, da es die Anpassungsfähigkeit des Roboters an wechselnde Umgebungsbedingungen einschränkt.

### 5.3 Zulässige Ausnahmen (erlaubtes Hardcoding)

- **Kalibrierungskonstanten**  
Offsets oder Skalierungsfaktoren, die nach einer initialen Sensor-Kalibrierung ermittelt werden, dürfen als feste Werte gespeichert werden.  
Beispiel: Korrektur eines Sensors um +12 zur Kompensation eines systematischen Messversatzes.
- **Toleranzbereichsdefinition**  
Ein fester Toleranzwert (z. B.  $\pm 5\%$ ) zur Bewertung der zulässigen Abweichung zwischen Sensoren nach erfolgter Kalibrierung darf hardcodiert werden.
- **Threshold-Methoden (siehe Abschnitt 3.4)**  
Es ist erlaubt, Schwellenwerte (z. B. dynamischer Threshold, Min-Max-Kalibrierung oder Verhältnisbewertung) im Code algorithmisch zu definieren, auch wenn dabei feste Parameter wie Zeitfenster oder Messanzahl (z. B. „letzte n Werte“) verwendet werden.
- **Bewegungsparameter**  
Werte wie Geschwindigkeit oder Drehgeschwindigkeit dürfen ebenfalls hardcodiert werden, auch wenn empfohlen wird, diese anpassbar zu gestalten.
- **Abstandsmessung mit Ultraschallsensor**  
Für die Messwerte des Ultraschallsensors werden feste Schwellenwerte festgelegt, um beispielsweise Hindernisse zu erkennen (z. B. `if (sensor < 10) → Wand`).

#### 5.4 Nicht zulässig (nicht erlaubtes Hardcoding):

Feste, unveränderliche „Schwarz-“ oder „Weißwerte“ (z. B. *if (sensor > 100) → Linie*) oder das manuelle Einstellen spezifischer Sensorschwellen, die nicht dynamisch aus Messwerten abgeleitet werden, sind nicht erlaubt.