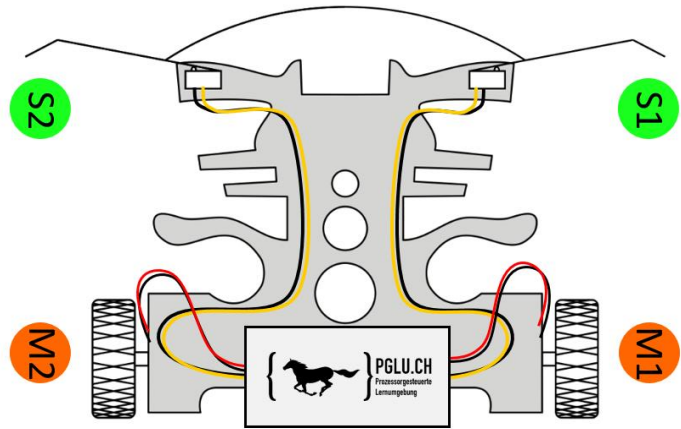


# 1



**Variante** Rückwärts bei Berührung und Vereinfachung mit Funktionen  
Jede Fahrrichtung wird in eine Funktion ausgelagert. Das macht das Programm übersichtlicher und einfacher zu erweitern.

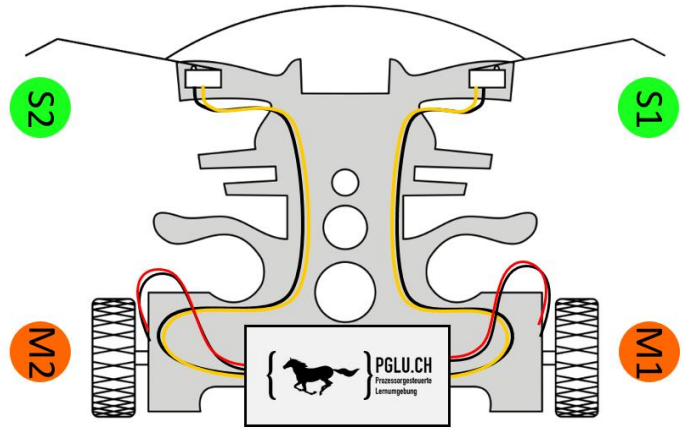
The screenshot shows a block-based programming environment. The main loop is set to 100,000 iterations per second. The logic is as follows:

- Wenn Sensor 1 = EIN (prüfe standard):**
  - rueck500
  - links500
- Sonst:** vorwärts
- Wenn Sensor 2 = EIN (prüfe standard):**
  - rueck500
  - rechts500
- Sonst:** vorwärts

The functions are defined as follows:

- vorwärts:** setze Motor 1 auf 100 %, setze Motor 2 auf 100 %
- rueck500:** setze Motor 1 auf -100 %, setze Motor 2 auf -100 %, pausiere 500 ms
- links500:** setze Motor 1 auf 100 %, setze Motor 2 auf -100 %, pausiere 500 ms
- rechts500:** setze Motor 1 auf -100 %, setze Motor 2 auf 100 %, pausiere 500 ms

# 2

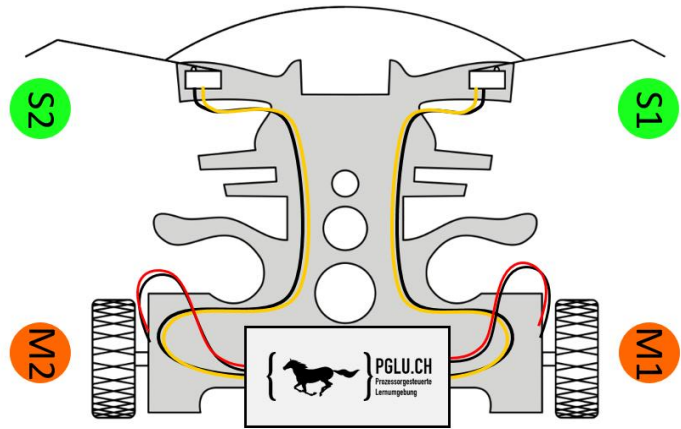


**Variante** mit Lauflicht bei Drehung – wie im Video auf Webseite

Was nach viel Arbeit aussieht, ist schnell erledigt: du kannst Blöcke und Gruppen mit einem Rechtsklick (lange Berührung) kopieren!

The screenshot displays the PGLU.CH programming environment. The interface includes a top navigation bar with 'Programmieren', 'Testen', 'Organisieren', and 'Hilfe'. Below this are tabs for 'Grafik', 'Code', 'Simulator', and 'Programme'. A left sidebar lists various components: Agieren, Sensor, Motor, LED, Steuern, Frage, Ablauf, Dimensionieren, Variable, Mathe, Vereinfachen, Funktion, Erweitern, and Ablauf II. The main workspace shows a block-based program. The 'Hauptloop' is set to '100'000 mal pro Sekunde'. The code consists of several conditional blocks: 'wenn Sensor 1 = EIN', 'wenn Sensor 2 = EIN', and two 'wiederhole' loops for 'Lauflicht Links' and 'Lauflicht Rechts'. Each 'wiederhole' loop contains a sequence of 'setze' blocks for LEDs and 'pausiere' blocks. On the right, there are several sub-blocks: 'Vor Hauptloop: 1x' with 'schreibe x = 25', 'Name Fahre Rechts', 'Name Fahre Links', 'Name Fahre Rück', and 'Name LEDs Touch'. The 'Name LEDs Touch' block has a 'zähle j von 100 bis 0 alle 1' block circled in yellow. A note at the bottom right says 'Setze für die Simulation hier 10 ein!'.

# 3



## Variante ohne Programmstopp durch Pausieren-Block

So würde ein Profi das Selbstfahrende Auto programmieren. Da das Programm nie pausieren muss, können die Sensoren immer abgefragt werden und sind dadurch permanent «scharf»

The screenshot shows a programming environment with a dark theme. The top navigation bar includes 'Programmieren', 'Testen', 'Organisieren', and 'Hilfe'. The main workspace contains the following code blocks:

- Vor Hauptloop: 1x** (highlighted in red):
  - schreibe Timer 1 = 1000
  - schreibe Timer 2 = 1000
- Hauptloop: 100'000 mal pro Sekunde** (highlighted in blue):
  - wenn Timer 1 < 1000:
    - setze Motor 1 auf -100 %
  - sonst:
    - setze Motor 1 auf 100 %
  - wenn Timer 2 < 1000:
    - setze Motor 2 auf -100 %
  - sonst:
    - setze Motor 2 auf 100 %
- Parallel zu Hauptloop: alle 1ms** (highlighted in red):
  - schreibe Timer 1 = Timer 1 + 1
  - schreibe Timer 2 = Timer 2 + 1
  - wenn Sensor 1 = EIN prüfe standard:
    - schreibe Timer 1 = 0
    - schreibe Timer 2 = 500
  - wenn Sensor 2 = EIN prüfe standard:
    - schreibe Timer 1 = 500
    - schreibe Timer 2 = 0

A yellow callout bubble points to the '1' in the 'schreibe Timer 1 = Timer 1 + 1' block with the text: 'Setze für die Simulation hier 100 ein!'.