



# Mit MATRIX die „Sprache der Roboter“ kennenlernen

Beispiel mit MATRIX Essential Set(MR0001)

Erstellt von Water Xu & MATRIX Robotics am 24.09.2025

# Benötigte Materialien



## Stelle Sicher, dass die "MATRIXblock" Software installiert ist

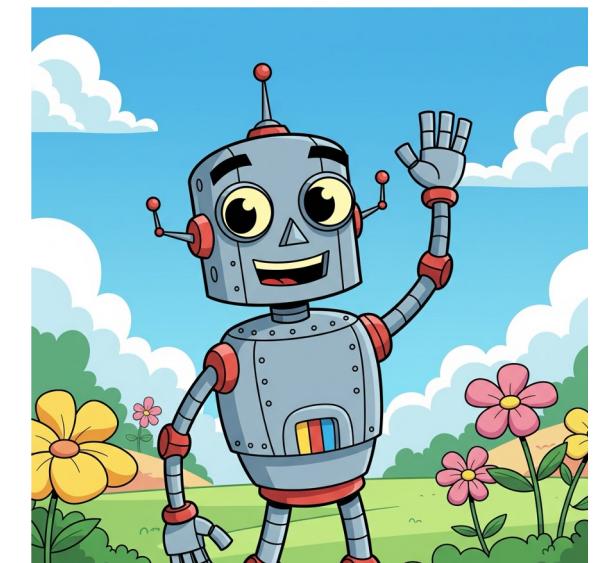
# In welcher Sprache sollten wir am besten mit Robotern sprechen?



# Die Sprache der Roboter - Programmiersprache

Genau wie wir Sprache verwenden, um mit Freunden zu sprechen,  
Ist Programmiersprache ein Weg, um mit Robotern zu sprechen.

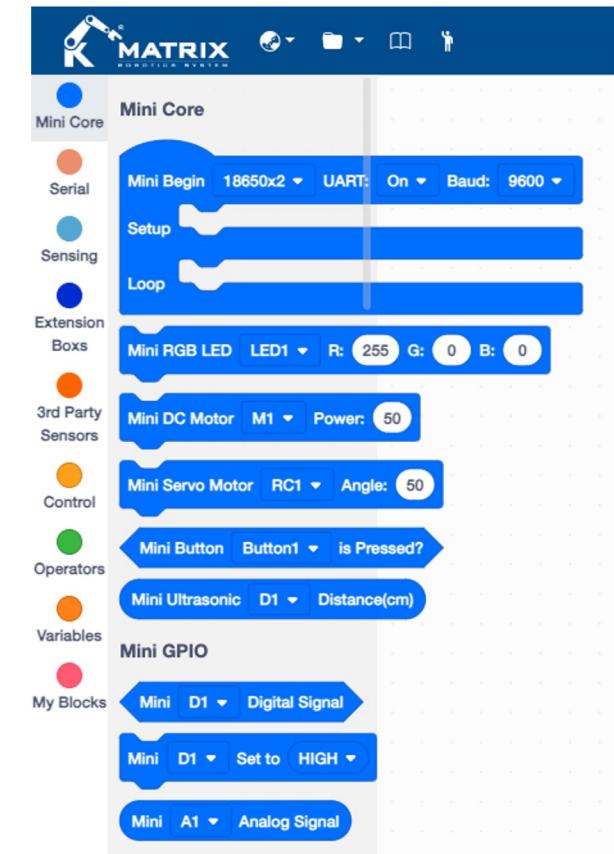
Wir geben Anweisungen vor,  
→ der Computer befolgt sie!



# MATRIXblock

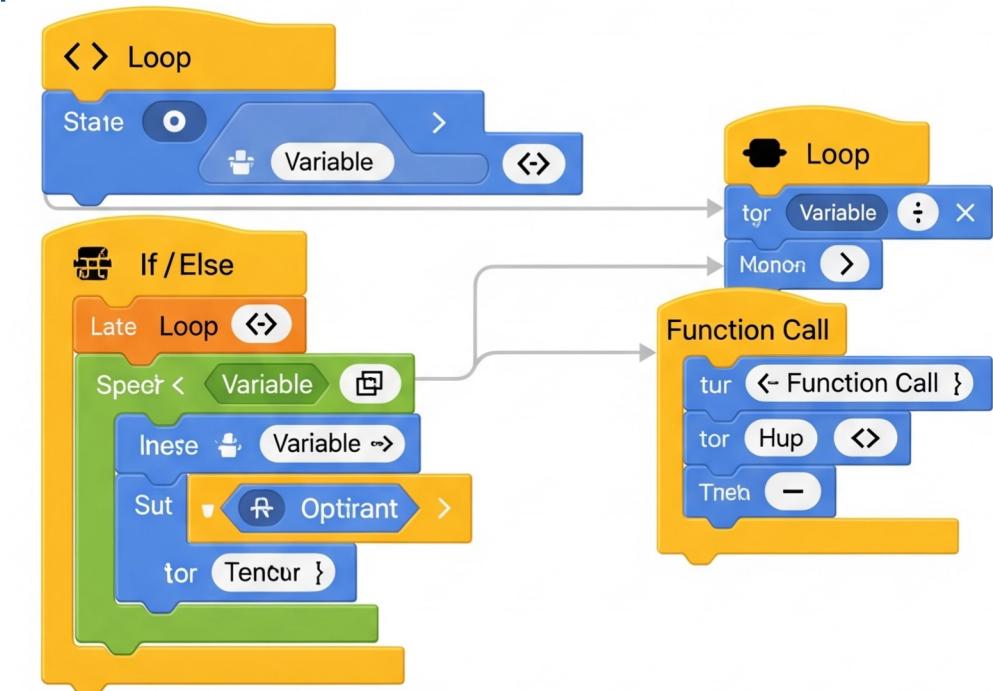
Eine Programmiersprache. Du brauchst keine Befehle zu schreiben oder zu merken; zieh einfach blocks mit deiner Maus dahin wo du sie möchtest und der Computer oder Roboter wird sie ausführen!

Jeder Block steht für eine Aktion oder einen Befehl, wie bspw. "move forward" , "make a sound" , "wait one second" oder "repeat 10 times" . Wenn du sie untereinander stapelst wird der Computer sie Schritt für Schritt abarbeiten.



# Lernziele:

1. Versteh das Konzept einer Programmiersprache
2. Schreib dein erste Programm



# Praktisches Beispiel

(30 Minuten)

1. Einführung in das MATRIXblock Interface (10 Minuten)
2. Programmieren (20 Minuten)



# Einführung in das MATRIXblock Interface

## Öffne MATRIXblock auf deinem Computer



The image shows a laptop screen displaying the MATRIXblock software interface. The interface includes a graphical programming workspace with blocks for motors, sensors, and logic, a text-based code editor with C++ code, and a serial port monitor. A red box highlights the MATRIXblock logo in the top right corner of the software window. To the left of the laptop, a physical MATRIX Mini Controller board is shown, featuring various pins, a digital display, and an Arduino logo. Below the laptop, a small blue button says "Learn more >".

**MATRIXblock**  
*Bridging Blocks to Code, Unleashing Creativity*

A graphical programming tool based on Scratch, offering block to C++ previews and a serial port monitor for easy data debugging. MATRIXBlock bridges the gap between block-based and text-based coding, ideal for beginners and educators.

**MATRIX Mini Controller**   **Software Interface Guide**   **Assembly Techniques**

**Basic Car**   **Obstacle Avoiding Robot**   **MATRIX Joystick 2**



File Edit View Tools Help

\*Example6.mbn2

⌘ ⌘ ⌘ ⌘ ⌘ ⌘ /dev/tty.usbs...



Werkzeugleiste

```
1 #include "MatrixMini.h"
2
3 void setup()
4 {
5
6 }
7
8 void loop()
9 {
10
11 }
```

No Device Selected

Send

The screenshot shows the MATRIX IDE interface with three main sections highlighted by red boxes:

- Blockpalette (Left):** Contains a sidebar with categories: Mini Core, Serial, Sensing, Extension Boxes, 3rd Party Sensors, Control, Operators, and Variables. Below this is a list of blocks categorized by function: Mini, Serial, Sensing, Extension Boxes, 3rd Party Sensors, Control, Operators, and Variables. A yellow box labeled "Blockpalette" is overlaid on the sidebar.
- Code Bereich (Center):** A large central workspace for writing code. A yellow box labeled "Code Bereich" is overlaid on the center area.
- Text Code Bereich & Upload Bereich (Right):** A sidebar on the right containing a text code editor with the following content:

```
1 #include "MatrixMini.h"
2
3 void setup()
4 {
5
6 }
7
8 void loop()
9 {
10
11 }
```

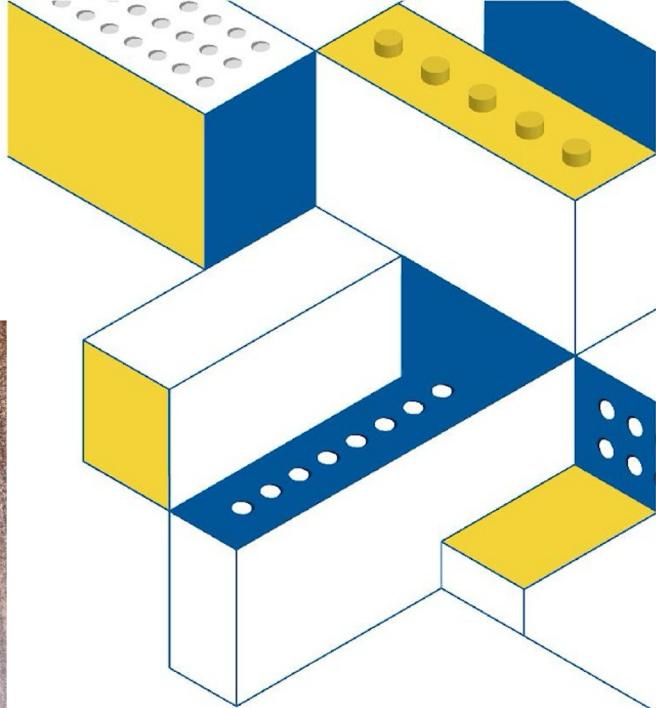
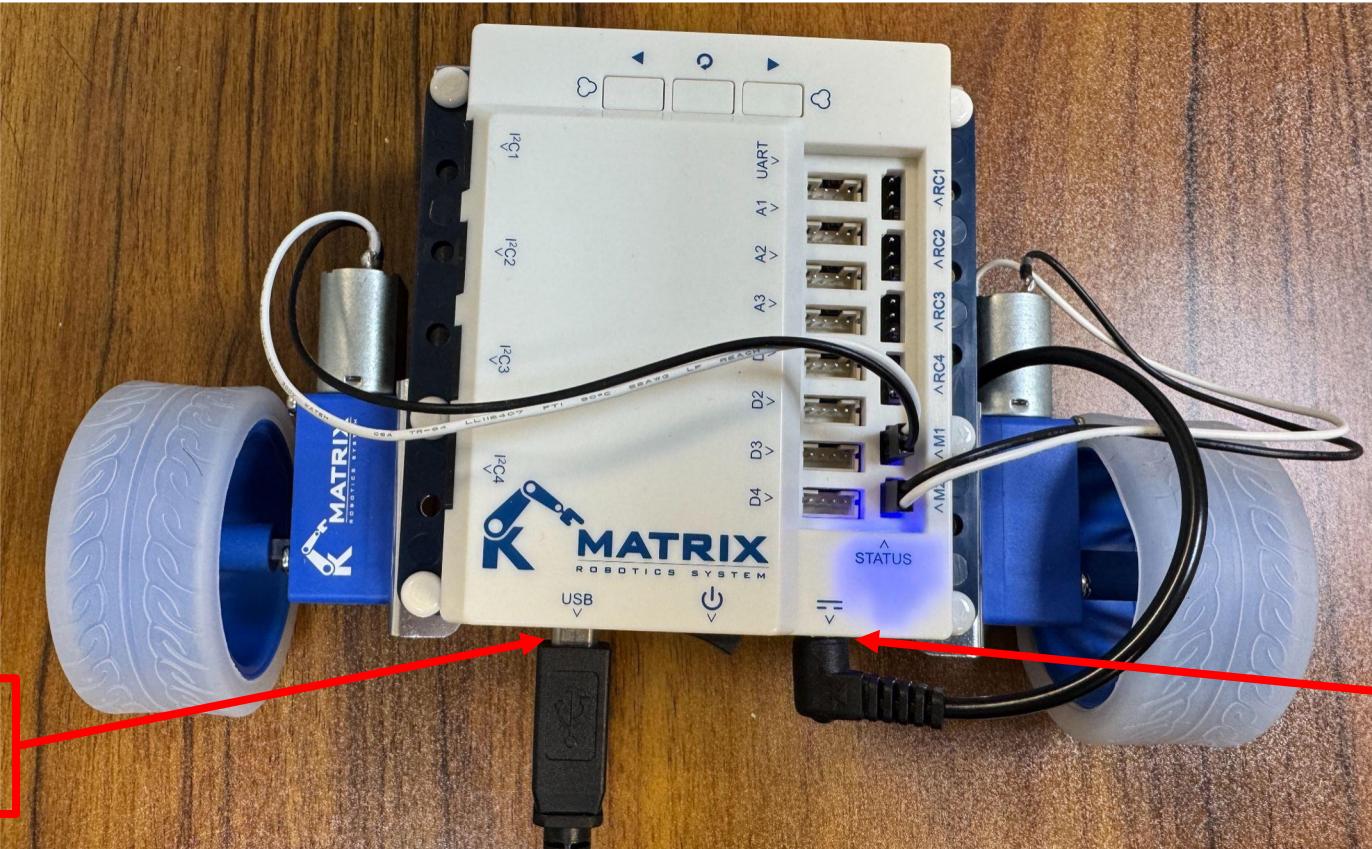
The sidebar also includes a "No Device Selected" message and a toolbar with icons for upload, delete, and search.



# MATRIXblock Programmierung

STEAM EDUCATION, FUTURE TECHNOLOGY.

# Verkabelung



MATRIX  
EDUCATION SYSTEM

\*Example6.mbn2

No Device

The interface includes a sidebar with categories: Mini Core, Serial, Sensing, Extension Boxes, 3rd Party Sensors, Control, Operators, Variables, and My Blocks. The main workspace shows a block-based code structure with a red callout box containing the German text: "Ziehe den Programm Hauptblock in den Code Bereich". The code preview panel on the right shows the generated C++ code:

```
1 #include "MatrixMini.h"
2
3 void setup()
4 {
5     Mini.begin(LI_2, 0, 9600);
6     Serial.begin(9600);
7 }
8
9 void loop()
10 {
11 }
```

No Device Selected

Block-based code structure:

- Mini Core:
  - Mini Begin: 18650x2, UART: On, Baud: 9600
  - Setup
  - Loop
- Extension Boxes:
  - Mini RGB LED: LED1, R: 255, G: 0, B: 0
  - Mini DC Motor: M1, Power: 50
  - Mini Servo Motor: RC1, Angle: 50
  - Mini Button: Button1, is Pressed?
  - Mini Ultrasonic: D1, Distance(cm)
- Variables
- My Blocks:
  - Mini: D1, Digital Signal
  - Mini: D1, Set to: HIGH
  - Mini: A1, Analog Signal
- Serial:
  - Serial Print: Hello
  - Serial Print: World, with New Line
  - Serial Write (ASCII): 65
  - Serial Chart DataSet: 32, 64, 128, Interval: 500 ms
  - Is Serial Available?
  - Serial Received Data (ASCII)

1. Ziehe 2 Motorkontrollblocks heraus.

2. Setze M1 power auf 50, M2 power auf 50.

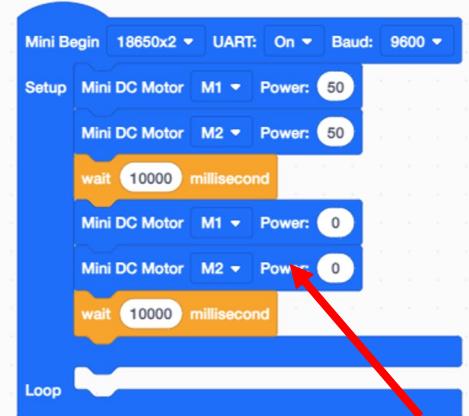
```
1 #include "MatrixMini.h"
2
3 void setup()
4 {
5     Mini.begin(LI_2, 0, 9600);
6     Serial.begin(9600);
7     Mini.M1.set(50);
8     Mini.M2.set(50);
9 }
10
11 void loop()
12 {
13 }
```

The image shows the MATRIX Scratch IDE interface with a script for two DC motors. The script consists of a **Setup** block and a **Loop** block. In the **Setup** block, two motors are initialized: **Mini DC Motor M1** and **Mini DC Motor M2**, both set to **Power: 50**. In the **Loop** block, the motors move forward for 10 seconds, then stop for 10 seconds, and repeat this cycle. The Scratch interface includes a block palette on the left and a script editor on the right. German instructions are overlaid on the interface:

1. Wähle Kontrollen aus
2. Ziehe den Warteblock heraus
3. Setze Warten auf 10000 Millisekunden (fahre für 10 Sekunden geradeaus)

Code Editor (Right Side):

```
1 #include "MatrixMini.h"
2
3 void setup()
4 {
5     Mini.begin(LI_2, 0, 9600);
6     Serial.begin(9600);
7     Mini.M1.set(50);
8     Mini.M2.set(50);
9     delay(10000);
10 }
11
12 void loop()
13 {
14 }
```



1 #include "MatrixMini.h"  
2  
3 void setup()  
4 {  
5 Mini.begin(LI\_2, 0, 9600);  
6 Serial.begin(9600);  
7 Mini.M1.set(50);  
8 Mini.M2.set(50);  
9 delay(10000);  
10 Mini.M1.set(0);  
11 Mini.M2.set(0);  
12 delay(10000);  
13 }  
14  
15 void loop()  
16 {  
17  
18 }

Ziehe ein Paar aus M1 und M2 blocks heraus,  
setze power auf 0, warte für 100  
Millisekunden  
(Dieses Programm stoppt die Bewegung)

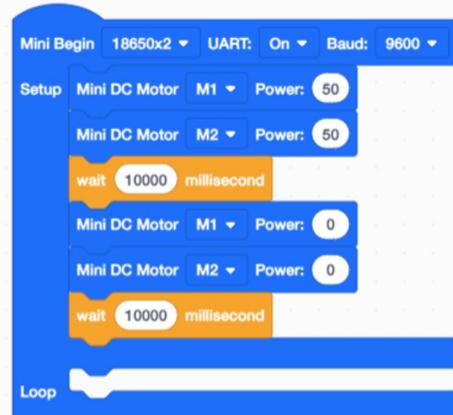
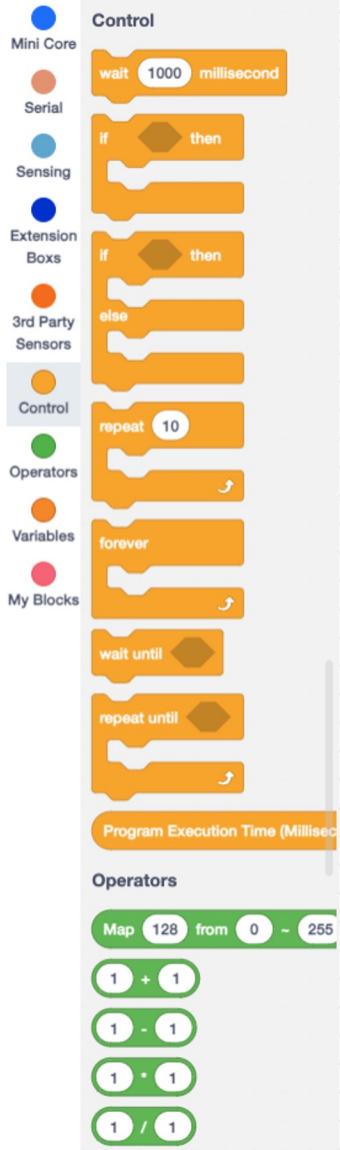


File Edit View Insert Project Help

\*Example6.mbn2

File Edit View Insert Project Help

USB Serial Port: /dev/tty.usb...



**Wähle den kleinen  
Roboter an**

```
1 #include "MatrixMini.h"
2
3 void setup()
4 {
5     Mini.begin(LI_2, 0, 9600);
6     Serial.begin(9600);
7     Mini.M1.set(50);
8     Mini.M2.set(50);
9     delay(10000);
10    Mini.M1.set(0);
11    Mini.M2.set(0);
12    delay(10000);
13
14
15 void loop()
16 {
17
18 }
```

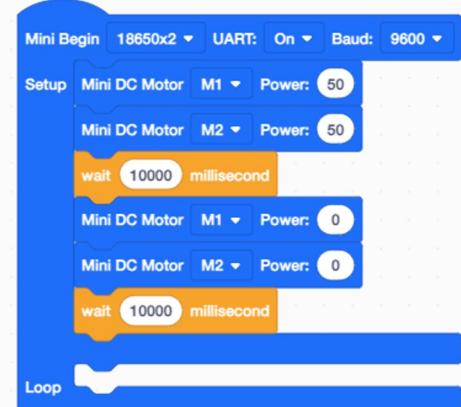
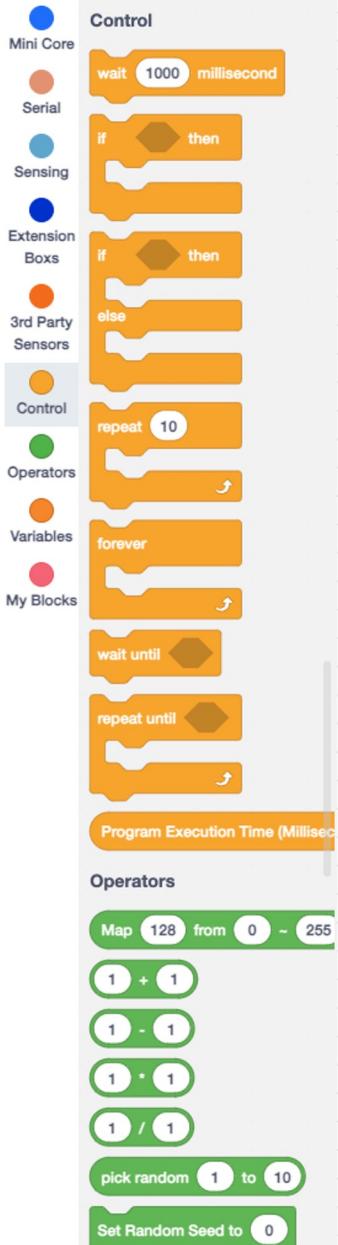


File Edit View Help

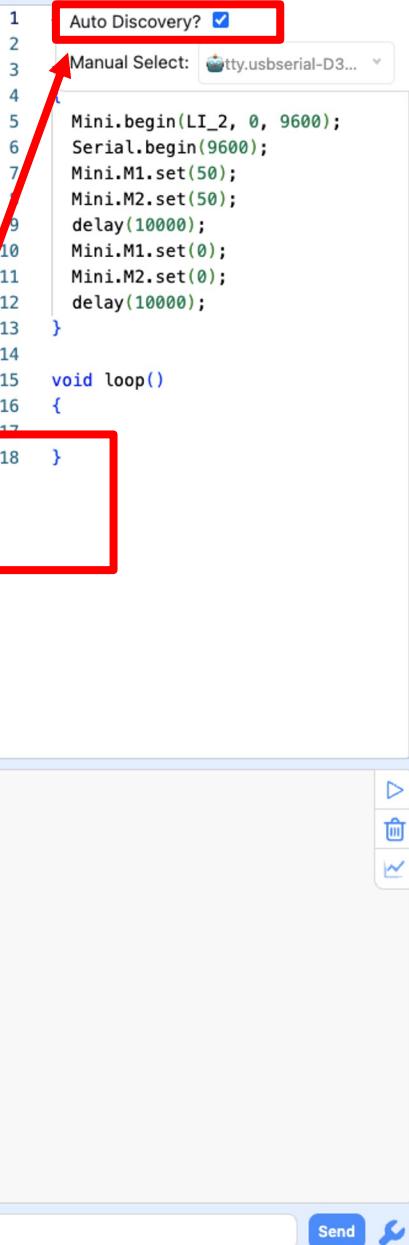
\*Example6.mbn2

File Edit View Help

File Edit View Help



**Wähle "Auto Discovery" an**

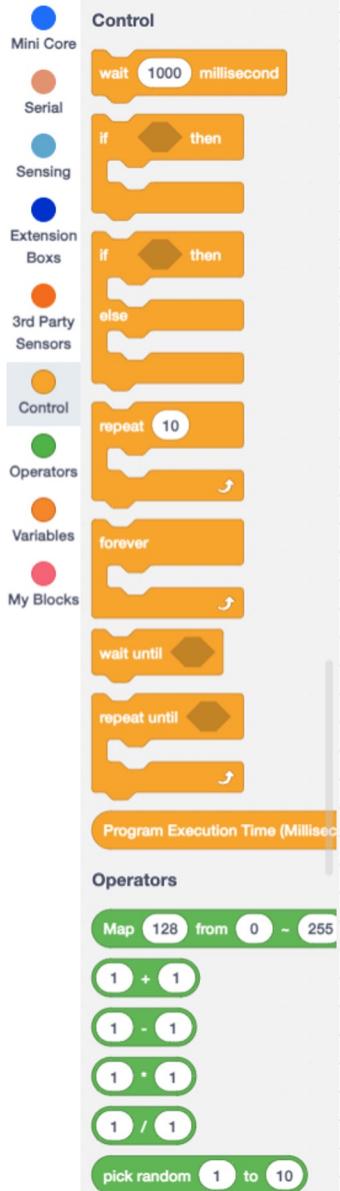




File Edit View Help

\*Example6.mbn2

File Edit View Help /dev/tty.usbs...



Klicke zum  
Starten

```
1 #include "MatrixMini.h"
2
3 void setup()
4 {
5     Mini.begin(LI_2, 0, 9600);
6     Serial.begin(9600);
7     Mini.M1.set(50);
8     Mini.M2.set(50);
9     delay(10000);
10    Mini.M1.set(0);
11    Mini.M2.set(0);
12    delay(10000);
13 }
14
15 void loop()
16 {
17 }
```

# Aufgabe für den Unterricht (10 Minuten )

Probier es aus,  
Kann der Roboter mit deinem  
Programm gleichmäßig fahren?

Schalte den Roboter an und er beginnt, das  
hochgeladene Programm auszuführen.

