

TikZ Codeblocks v.0.12

Adrian Salamon

2018-04-03

Inhaltsverzeichnis

1 Präambel	2
1.1 Editoren mit graphischer Programmierung	2
1.2 Alternativpakete	3
1.3 Installation und Benutzung	3
2 Beispielcode	3
2.1 English Codeexample	5
3 Bausteine und Befehle	6
3.1 Bausteinklassen nach PXT-Calliope Editor	6
3.2 Skalierung	7
3.3 Farben	7
3.3.1 Lokale Farbänderung	7
3.3.2 Globale Farbänderung	8
3.3.3 Keine Farben (print)	8
3.4 Boxen	8
3.5 Bilder/LED-Matrix	9
3.6 Strukturen	10
3.6.1 Verzweigungen	10
3.6.2 Schleifen	11
3.6.3 Branches, loops and the english language	12
4 Positionierung der Nodes	12
4.1 Manuelles Positionieren	12
4.1.1 Verschieben der Zacken bei manueller Einrückung	13
4.2 Automatisches Einrücken	13
5 Dekorationen	14
5.1 Puzzleoptik	14
5.2 Symbole und kleine Elemente	14
6 FAQ	15
6.1 Wie kann ich für ein komplettes Dokument eine Farbe umdefinieren?	15
6.2 Eine nested Box ist zu hoch/zu niedrig	15
6.3 Ich will einen Node innerhalb eines Nodes setzen	15
6.4 Mein Block ist sehr klein und deswegen verformt	15

7 Mehr Beispielcode	16
7.1 Bsp: Calliope Smart Home	16

Versionshistorie

Version	Datum	Autor(en)	Änderungen
0.1	2017-07-06	A. Salamon	published
0.11	2017-07-18	A. Salamon	update: shapes, translations and fix typos
0.12	2018-04-03	A. Salamon	update: LED colors

- Translations
 - via **babel** for structures (if/wenn, then/dann...)
 - marcos and styles are now (partly) also in english¹

- new Shapes



no colors

- Package option **print** gets rid of all colors:
- Open-Roberta colors are now more accurate and usable next to the standard colors. (see 3.1.)

- fixed LED-Color for NEPO-Editor.



1 Präambel

Diese Sammlung von TikZ Stilen und Kommandos soll helfen, grafische Codeblöcke, wie beim Calliope mini Editor, der Sprache NEPO von Open Roberta oder der Programmierumgebung Scratch zu setzen. Sie ersetzt **nicht** die Auseinandersetzung mit TikZ und der entsprechenden Syntax. Die Positionierung, Benennung und Referenzierung der Elemente muss weiterhin manuell gestaltet werden. Dies ist eine Entwicklerversion. Bezeichner und Paketname können in späteren Versionen noch variieren.

Fragen nehme ich gerne per adriansalamon@gmail.com entgegen.

1.1 Editoren mit graphischer Programmierung

tikzcodeblocks wurde erstellt, um Quelltexte auf Blockbasis für den Microcontroller Calliope mini zu setzen. Alle verwendeten Farben und Gestaltungen

¹Feel free to contact me, if you are an english speaker and want to use everything in english.

sind daher standardmäßig vom PXT Editor von Calliope inspiriert. Als alternative Farbgebung ist eine Variation nach Open Roberta implementiert (siehe 3.1). Es lässt sich jedoch auch Code mit Farb- und Formgebungen anderer Editoren setzen.

Editor	URL
PXT - Calliope	http://pxt.calliope.cc/
Open Roberta	https://lab.open-roberta.org/
Scratch	https://scratch.mit.edu/projects/editor

Tabelle 2: Tabelle von Editoren mit graphischen Codeblöcken.

1.2 Alternativpakete

Während der Entwicklung dieses Paketes wurde das Paket `scratch`² veröffentlicht. Damit lässt sich graphischer Code in der Optik von Scratch sehr einfach setzen. Die Dokumentation des Pakets ist zur Zeit auf Französisch verfügbar. Ein Unterschied zwischen den Paketen `scratch` und `tikzcodeblocks` ist m. E. vor allem folgender: `scratch` lässt sich **nicht** in andere TikZ Umgebungen einbinden. Die einzelnen Objekte sind nicht als Nodes³ referenzierbar. Die Syntax ist jedoch deutlich schmäler als bei `tikzcodeblocks`. Ein Blick lohnt sich bestimmt für die meisten interessierten Nutzer.

1.3 Installation und Benutzung

- `tikzcodeblocks` Das Paket wird über die üblichen L^AT_EX-Paketmanager installiert (z. B. T_EXLive) und per `\usepackage{tikzcodeblocks}` in die Präambel des gewünschten Dokuments eingebunden.
- `codeblocks` Mit `\begin{tikzpicture}[codeblocks]` werden die Codeblock-Stile in die jeweilige tikzpicture-Umgebung geladen. Es setzt das Verhalten für `every Node`.

```

1 \documentclass{standalone}
2 \usepackage{tikzcodeblocks}
3 \begin{document}
4   \begin{tikzpicture}[codeblocks] Hello World;
5     \node[mathe]{Hello World;};
6   \end{tikzpicture}
7 \end{document}

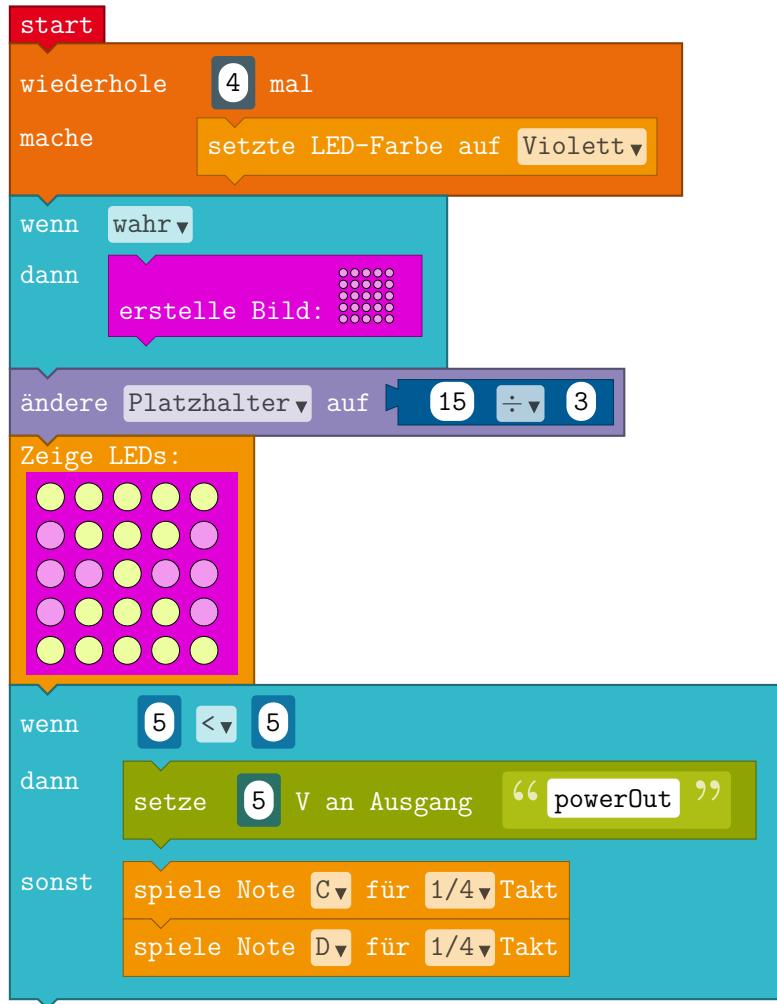
```

2 Beispielcode

Eine Beispiel zeigt Ergebnisse der Nutzung des Pakets. Der Quellcode zur Erstellung der Vektorgrafik folgt unter dem Beispiel.

²<https://www.ctan.org/pkg/scratch?lang=de>.

³In dieser Dokumentation wird die Bezeichnung **Node** für die Knoten in TikZ verwendet.



```

1 \begin{tikzpicture}[codeblocks,openroberta,scale=.90]
2
3 \node [start,pinlow](start){start};
4 \schleife[unter={start}{0}{0}]{wiederhole}{\intbox{4} mal}{\node[
  aktion]{setzte LED-Farbe auf \dropdown{Violett}};}{sch1}
5 \wenndann[unter={sch1}{0}{0}]{\dropdown{wahr}}{\node[bild]{
  erstelle Bild: \bild[0.3]{\emptyled}};}{verz1}
6 \node [variablen,unter={verz1}{0}{0}](plz1){\ändere \dropdown{
  Platzhalter} auf \tikz\node[mathe,boden,keinezacken,puzzleteil
  ]{\intbox{15}\dropdown{$\div$}\intbox{3}};};
7 \node [aktion,unter={plz1}{0}{0}](bild1){Zeige LEDs:\\
  \bild{
    \X \X \X \X \X \\
    \O \X \X \X \O \\
    \O \O \X \O \O \\
    \O \X \X \X \O \\
    \X \X \X \X \X \\
  }
  };
16
17 \wenndannsonst[unter={bild1}{0}{0}]
18 {\intbox{5}\dropdown{<}\intbox{5}}
19 {\node[sensor]{setze \intbox{5}V an Ausgang \stringbox{powerOut
  }}};

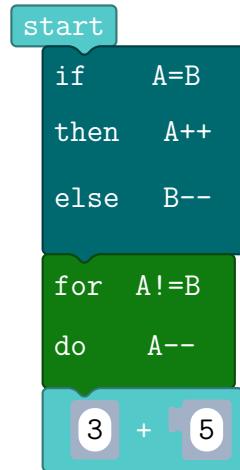
```

```

20  || {\node [aktion,draw,] (akt1){spiele Note \dropdown{C} für \dropdown
21    {1/4}\, Takt};}
22  || \node [aktion,draw,pinhigh,unter={akt1}{0}{0}](akt2){spiele Note
23    \dropdown{D} für \dropdown{1/4}\, Takt};}
24  || {wds1}
24  || \end{tikzpicture}

```

2.1 English Codeexample



```

1  || \begin{otherlanguage}{english}
2  || \begin{tikzpicture}[codeblocks]
3  || \moveindent{
4  ||   \node[start,pinlow](one){start};
5  || }
6  || \ifthenelseblocks[under={one}{1}{0}]{A=B}{\node{A++};}{\node{B
7  ||   -{}-};}{two}
7  || \loopblocks[under={two}{0}{0}]{for}{A!=B}{\node{A-{}-};}{three}
8  || \node[under={three}{0}{0},aktion,pinhigh]{\intbox{3}+\intbox[
8  ||   puzzlepiece]}{5};
9  || \end{tikzpicture}
10 || \end{otherlanguage}

```

3 Bausteine und Befehle

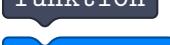
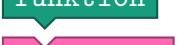
3.1 Bausteinklassen nach PXT-Calliope Editor

Die Klassen der Programmierbausteine werden nach folgendem Muster als TikZ-Style angegeben.

```
\node[STYLE] {Inhalt};
```

Die Defaulteinstellung orientiert sich an den Farben und Formen des Calliope-PXT-Editors. Mithilfe des Stils `openroberta` können alternative Farb- und Formdefinitionen geladen werden, die sich an der NEPO-Umgebung von Open-Roberta orientieren. Der Stil kann auch für eine ganze TikZ-Umgebung verwendet werden.

Stilname	pxt (Standard)	openroberta
grundlage	 grundlage	 grundlage
eingabe	 eingabe	 eingabe
schleife	 schleife	 schleife
logik	 logik	 logik
musik	 musik	 musik
led	 led	 led
platzhalter	 platzhalter	 platzhalter
mathe	 mathe	 mathe
funk	 funk	 funk
motor	 motor	 motor
zeichenkette	 zeichenkette	 zeichenkette
spiel	 spiel	 spiel
bild	 bild	 bild
pins	 pins	 pins
konsole	 konsole	 konsole
steuerung	 steuerung	 steuerung
bluetooth	 bluetooth	 bluetooth
start	 start	 start
aktion	 aktion	 aktion
sensor	 sensor	 sensor

Stilname	pxt (Standard)	openroberta
kontrolle	 kontrolle	 kontrolle
liste	 liste	 liste
farbe	 farbe	 farbe
bild	 bild	 bild
variablen	 variablen	 variablen
funktion	 funktion	 funktion
nachricht	 nachricht	 nachricht

! **Hinweis:** In PXT und im Open-Roberta-Editor heißen die entsprechenden Einträge für Zeichenketten „Texte“. Der Style „Text“ ist von TikZ jedoch bereits intern belegt und wird hier deswegen als „zeichenkette“ verwendet.
 Standardmäßig wird das Paket mit der PXT-Farbinformation geladen, um die Farben des PXT-Editors für Calliope zu verwenden.⁴
 Die Stile unterscheiden sich voneinander nur durch ihre Farben. openroberta lädt standardmäßig noch den Stil eckig.

3.2 Skalierung

Bsp: Skalierung Die Skalierung des gesamten Bildes ist mit dem TikZ-Boardmittel `scale` möglich.

```

1 \begin{tikzpicture}[codeblocks, scale=0.7]
2   \node[variablen]{Hello World;};
3 \end{tikzpicture}
4 %
5 \begin{tikzpicture}[codeblocks, scale=1.3]
6   \node[variablen]{Hello World;};
7 \end{tikzpicture}
```



3.3 Farben

3.3.1 Lokale Farbänderung

Lokale Überschreibungen sind – wie üblich – durch Angabe einer Farbe bei den entsprechenden Node-Attributen möglich.

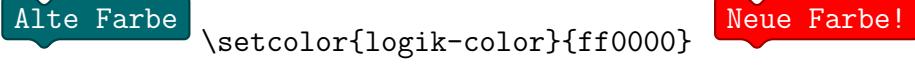
Bsp: Lokale Farbänderung 

```
\node[fill=black,draw=red]{schwarzer Hintergrund - roter Rahmen}
```

⁴Damit beide Stile problemlos ineinander überführt werden können, ohne dass bestimmte Stile in anderen Kontexten undefiniert sind, wurden einige Stile doppelt Zugeordnet: pxt-grundlagen = openroberta-start, pxt-musik = openroberta-aktion, etc.

3.3.2 Globale Farbänderung

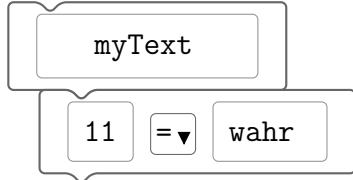
\setcolor Der Befehl \setcolor{<farbreferenz>}{<hexfarbcode>} lässt zu, Farben dokumentenweit umzudefinieren.

Bsp: Globale Farbänderung 

3.3.3 Keine Farben (print)

Mit der Paketoption \usepackage[print]{tikzcodeblocks} lassen sich alle Farben entfernen. Es werden lediglich Umrisse, sowie ausgefüllte LEDs gezeichnet. Die Schriftfarbe ist schwarz.

Bsp: print



3.4 Boxen

Boxen werden innerhalb von Codeblöcken verwendet, um bestimmte Platzhalter und Datentypen zu kennzeichnen. Die Farben werden dabei teilweise in Abhängigkeit zum Parent (durch Durchsichtigkeit) gesetzt.

\dropdown
\intbox
\stringbox
\boolbox

Code	PXT	Open Roberta
\dropdown{Dropdown}	Dropdown▼	Dropdown▼
\intbox{5}	5	5
\stringbox{Text}	“ Text ”	“ Text ”
\boolbox{wahr}	wahr	wahr

\intbox, \stringbox, \boolbox haben alle als optionales Argument die Möglichkeit Stile hinzuzufügen.

Bsp: optionales Argument bei Boxen

42 \intbox{42}

42 \intbox[puzzleteil]{42}

Bsp: Boxen mit Kapselung

Meine Dropdown-Box mit Wert 5

\node[eingabe]{Meine \dropdown{Dropdown}-Box mit Wert \intbox{5}};

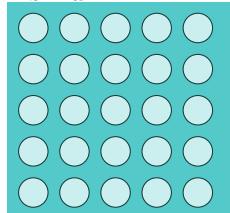
! Hinweis: Bei der Verwendung des Stils `openroberta` ändern sich auch entsprechend die Farben der int-, string- und boolboxen.

3.5 Bilder/LED-Matrix

`\bild` Mit `\bild[<skalierungsfaktor>]{<Inhalt>}` lassen sich LED-Matrizen setzen. Es erwartet einen Tabelleninhalt. Jede Zeile muss entsprechend per `\\"` beendet werden. Zeilen und Spalten können dabei unbegrenzt sein. Dabei gelten weiterhin folgende Befehle:

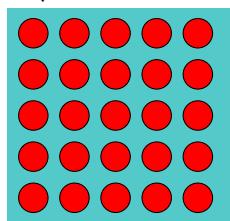
`\emptyled`

- `\emptyled` setzt eine 5×5 LED-Matrix, bei der alle LEDs ausgeschaltet sind.



`\fullled`

- `\fullled` setzt eine 5×5 LED-Matrix, bei der alle LEDs angeschaltet sind.



`\X`

- `\X` repräsentiert darin eine angeschaltete LED



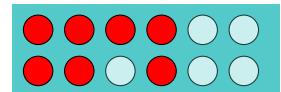
`\0`

- `\0` repräsentiert darin eine ausgeschaltete LED
- Die Kombination von `\bild{}`, `\X`, und `\0` ergibt schließlich alle möglichen LED-Matrizen:

Bsp: Beliebige Matrix

```

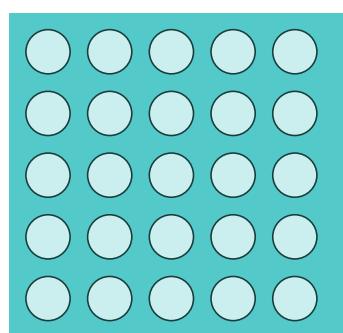
1 |   \bild{
2 |     \X \X \X \X \0 \0 \\
3 |     \X \X \0 \X \0 \0 \\
4 | }
```



`\bild`

- Mit dem optionalen Argument kann ein Skalierungsfaktor angegeben werden.

Bsp: Bildskalierung



`\bild[0.4]{\emptyled}` `\bild[1.5]{\emptyled}`

! Hinweis: \x, \0 sowie \emptyled und \fullled können nur innerhalb des \bild-Kommandos verwendet werden.

3.6 Strukturen

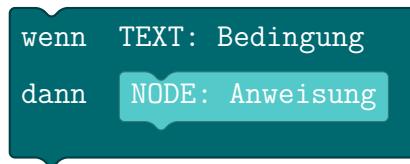
Strukturen helfen, die Positionierung von Nodes zu vereinfachen. Dafür können Verzweigungen und Schleifen verwendet werden. Damit muss nur noch in seltenen Fällen eine manuelle Positionierung von Nodes vorgenommen werden. Intern werden Tabellen verwendet.

3.6.1 Verzweigungen

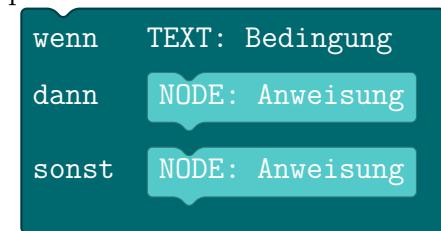
\wenndann Über den Befehl \wenndann[⟨TikZ-Stil⟩]{⟨TEXT: Bedingung⟩}{⟨NODE: Anweisung⟩}{⟨TEXT: Nodename⟩} wird die Kontrollstruktur Verzweigung abgebildet. Der logik-/bzw. kontrolle-Stil wird automatisch gesetzt. Das letzte Argument ist die Bezeichnung des eigenen Nodennamen, damit nachfolgende Codeblöcke dies bei ihrerer Positionierung referenzieren können.

\wenndannsonst \wenndannsonst[⟨TikZ-Stil⟩]{⟨TEXT: Bedingung⟩}{⟨NODE: Dann-Anweisung⟩}{⟨NODE: Sonst-Anweisung⟩}{⟨TEXT: Nodename⟩} verhält sich analog zu \wenndann, ist jedoch um einen Sonst-Block, der mit Nodes gefüllt wird, erweitert. Automatisch ergänzt werden die Wörter „wenn“, „dann“ und „sonst“.

! Hinweis: Das optionale Argument ist oft notwendig, um die Verzweigung richtig in Relation zu vorherstehenden Blöcken zu positionieren. Siehe hierfür 4.1.

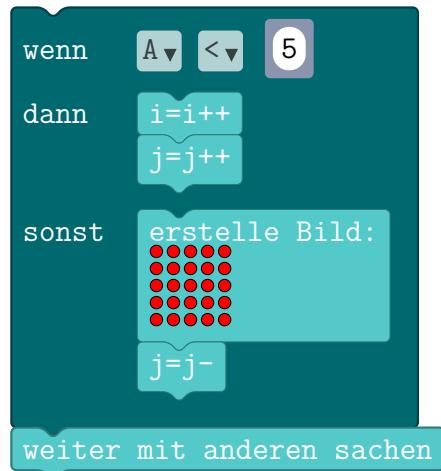


Deutsch: \wenndann
English: \ifthenblocks



\wenndannsonst
\ifthenelseblocks

Bsp: Verzweigungen



¹ \begin{tikzpicture}[codeblocks]
² \wenndannsonst [draw]

```

3   {\dropdown{A}~\dropdown{<}~, \intbox{5}} %wenn
4   { \node[aktion](akt1){i=i++};
5     \node[farbe, unter={akt1}{0}{0}](akt2){j=j++}; } %
6     dann
7   { \node[aktion, ](akt1){erstelle Bild:\\
8     \bild[0.4]{\fullled}
9   };
10    \node[farbe, unter={akt1}{0}{0}](akt2){j=j--}; }
11  } % sonst -ende
12  {eins}; % eigener Name
13 \node[aktion, unter={eins}{0}{0}]{weiter mit anderen
14   sachen};
15 \end{tikzpicture}

```

! **Hinweis:** Sollen Nodes in Textfeldern gesetzt werden, so muss `\tikz` vorgeschenben werden.

3.6.2 Schleifen

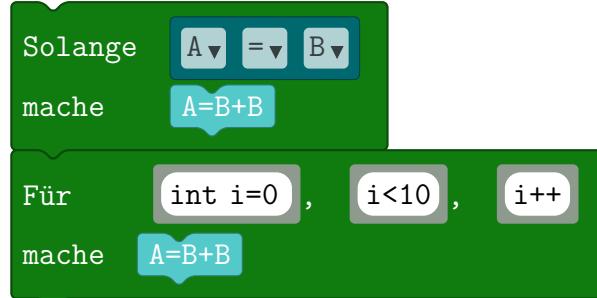
\schleife Die Schleife nach dem Muster `\schleife[⟨TikZ-Stil⟩]{⟨TEXT: Für/solange/etc.⟩}{⟨TEXT: Bedingung⟩}{⟨NODE: Anweisung⟩}{⟨TEXT: Nodename⟩}` ist eine weitere vordefinierte Struktur. Automatisch ergänzt wird der Begriff „mache“.

Bsp: Muster: Schleifen

TEXT: Für/solange/etc	TEXT: Bedingung
mache	NODE: Anweisung

deutsch `\schleife[⟨STIL⟩]{⟨PRE⟩}{⟨BED⟩}{⟨ANW⟩}{⟨NAME⟩}`
 english `\loopblocks[⟨STYLE⟩]{⟨PRE⟩}{⟨COND⟩}{⟨INST⟩}{⟨NAME⟩}`

Bsp: Schleifen



```

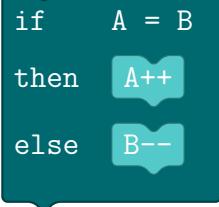
1 \schleife[draw]{Solange}{\tikz\node[logik,keinezacken
2   ]{\dropdown{A}~\dropdown{=}~\dropdown{B}};}{%
3   \node[aktion]{A=B+B};}{schl1}
4 \schleife[draw, unter={schl1}{0}{0}]{Für}{\intbox{int i
5   =0}, \intbox{i<10}, \intbox{i++}}{%
6   \node[aktion]{A=B+B};}{schl2}

```

3.6.3 Branches, loops and the english language

If you want to use this package in an english document just load `\usepackage[english]{babel}` in the preamble. It will automatically set the outer words for branches and loops in english.

Bsp: English example `\usepackage[english]{babel}`



```

1 \begin{otherlanguage}{english}
2   \begin{tikzpicture}[codeblocks]
3     \wenndannsonst{A = B}{\node [aktion]{A++};}{\node [
4       aktion]{B--};}{name}
5   \end{tikzpicture}
6 \end{otherlanguage}

```

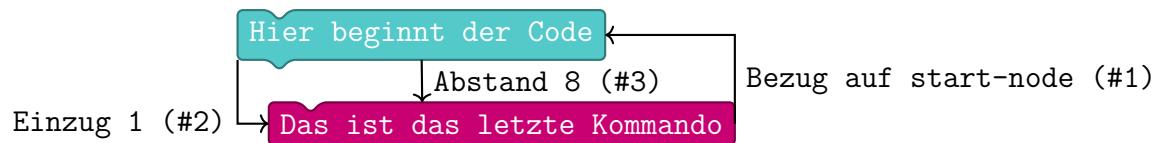
`\ifthenblocks` To fit the commands to the english language please you can use `\ifthenblocks`,
`\ifthenelseblocks` `\ifthenelseblocks` and `\loopblocks`. Notice the added „blocks“, because the
`\loopblocks` L^AT_EXCommand `\ifthenelse` is already used by the ifthen-Package.

	english	german
	<code>\ifthenblocks</code>	<code>\wenndann</code>
	<code>\ifthenelseblocks</code>	<code>\wenndannsonst</code>
	<code>\loopblocks</code>	<code>\schleife</code>

4 Positionierung der Nodes

4.1 Manuelles Positionieren

Mit konsequenter Verwendung der Strukturen ist manuelles Einrücken selten notwendig. Jedoch ist das Aneinanderketten der Nodes unabdinglich. Hierfür wird der Stil `unter={⟨NODE⟩}{⟨X-Einzug-Faktor⟩}{⟨Y-Einzug-Faktor⟩}` verwendet. Hierbei wird der Einzug als Ankerpunkt jeweils relativ zum Vorgänger gesetzt.



```

1 \node[grundlage] (drueber) {Hier beginnt der Code};
2 \node[eingabe, unter={drueber}{1}{8}] (drunter){Das
      ist das letzte Kommando};

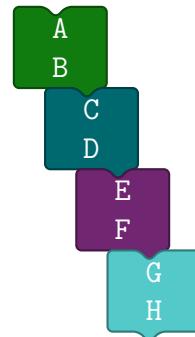
```

! **Hinweis:** Dank der Verwendung von Kontrollstrukturen (siehe 3.6) ist manuelles Einrücken in der Regel nicht notwendig.

4.1.1 Verschieben der Zacken bei manueller Einrückung

- \einruecken Über das Kommando \einruecken{\{nodes\}} lässt sich der untere Zacken eines Nodes um genau einen Einzug verschieben (siehe: 4.1;). Die obere Ausbuchtung bleibt an ihrem normalen Platz.
- \moveindent The english equivalent to \einruecken is \moveindent.

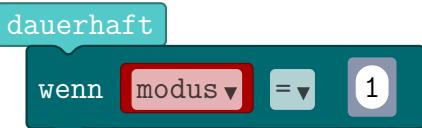
Bsp: Verschachteltes Einrücken



```

1 \begin{tikzpicture}[codeblocks, minimum width=1.2cm]
2 \einruecken{\node[kontrolle]{eins}{A\\ B};
3   \einruecken{\node[logik, unter={eins}{1}{0}]{zwei}{C\\ D};
4     \einruecken{\node[mathe, unter={zwei}{1}{0}]{drei}{E\\ F};}
5       \node[aktion, unter={drei}{1}{0}]{vier}{G\\ H};
6   }
7 }
8 \end{tikzpicture}
```

Bsp: einruecken



```

1 \einruecken{
2   \node[grundlage, pinlow]{grund}{dauerhaft};
3 }
4 \node[logik, unter={grund}{1}{0}]{wenn1}{wenn \tikz\node
  [platzhalter, boden, keinezacken]{\dropdown{modus}}; \
  dropdown{=} \intbox{1} };
```

4.2 Automatisches Einrücken

Siehe Strukturen 3.6!

5 Dekorationen

5.1 Puzzleoptik

- Um die Verzahnung der einzelnen Elemente darzustellen, wird standardmäßig eine **Puzzleoptik** verwendet, welche die *vertikale* Beziehung der Bausteine zueinander verdeutlicht. Sie ist über die Form **robertashape** definiert.
- puzzleteil** Soll die *horizontale* Beziehung von Bausteinen betont werden, so kann die Form **puzzleteil** verwendet werden.
- keinezacken** Mit dem Stil **keinezacken** lassen sich alle vordefinierten Zacken entfernen. Dies ist v. A. bei verschachtelten Nodes notwendig. **keinezacken** ist ein Alias für **rectangle**.
- eckig** Der Stil **eckig** entfernt alle runde Ecken und orientiert sich damit stärker an dem Erscheinungsbild von Open Roberta.
- pinlow** Für Start- und Endbausteine sind die Formen **pinlow** und **pinhigh** definiert.
- pinhigh**

Shape	deutsch	english
	pinlow	pinlow
	pinhigh	pinhigh
	keinezacken	nopins
	eckig	square
	eckig, keinezacken	square, nopins
	puzzleteil	puzzlepiece
	puzzleteil, eckig	puzzlepiece, square

Tabelle 4: Übersicht über Blockformen

Bsp: `keinezacken` Der Stil „keinezacken“ ist z. B. bei verschachtelten Nodes notwendig:



```
1 || \node [aktion] {mache \tikz\node [mathe, keinezacken] {A=B};};
```

5.2 Symbole und kleine Elemente

- \usb** `\usb` setzt ein .
- \farbe** `\farbe{color}` setzt ein Quadrat mit der angegeben Farbe . Zu verwenden auch in Nodes: 

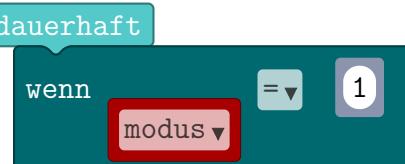
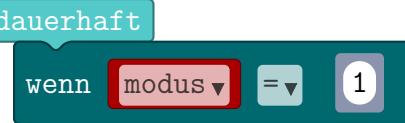
6 FAQ

6.1 Wie kann ich für ein komplettes Dokument eine Farbe umdefinieren?

\setcolor

Ist: 
Soll: 
Lösung: \setcolor{logik-color}{ff0000}

6.2 Eine nested Box ist zu hoch/zu niedrig

Ist: 
Soll: 
Lösung: Verwende den Hilfstyle boden:

```
1 || \einruecken{  
2   \node[grundlage,pinlow](grund){dauerhaft};  
3 }  
4 \node[logik,unter={grund}{1}{0}](wenn1){wenn \tikz  
  \node[platzhalter,boden,keinezacken]{\dropdown{  
    modus}}; \dropdown{=} \intbox{1} };
```

6.3 Ich will einen Node innerhalb eines Nodes setzen

Verwende eine verschachtelte TikZ-Umgebung, z. B. mit dem \tikz-Befehl:

außen: 

```
1 || \node[logik]{außen:  
2   \tikz\node[aktion]{innen};  
3 };
```

6.4 Mein Block ist sehr klein und deswegen verformt

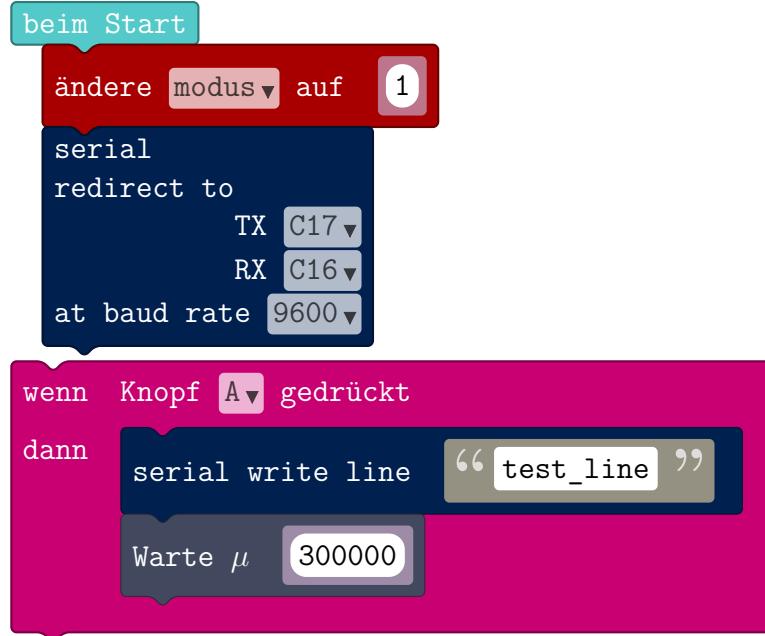
Ist: 
Soll: 

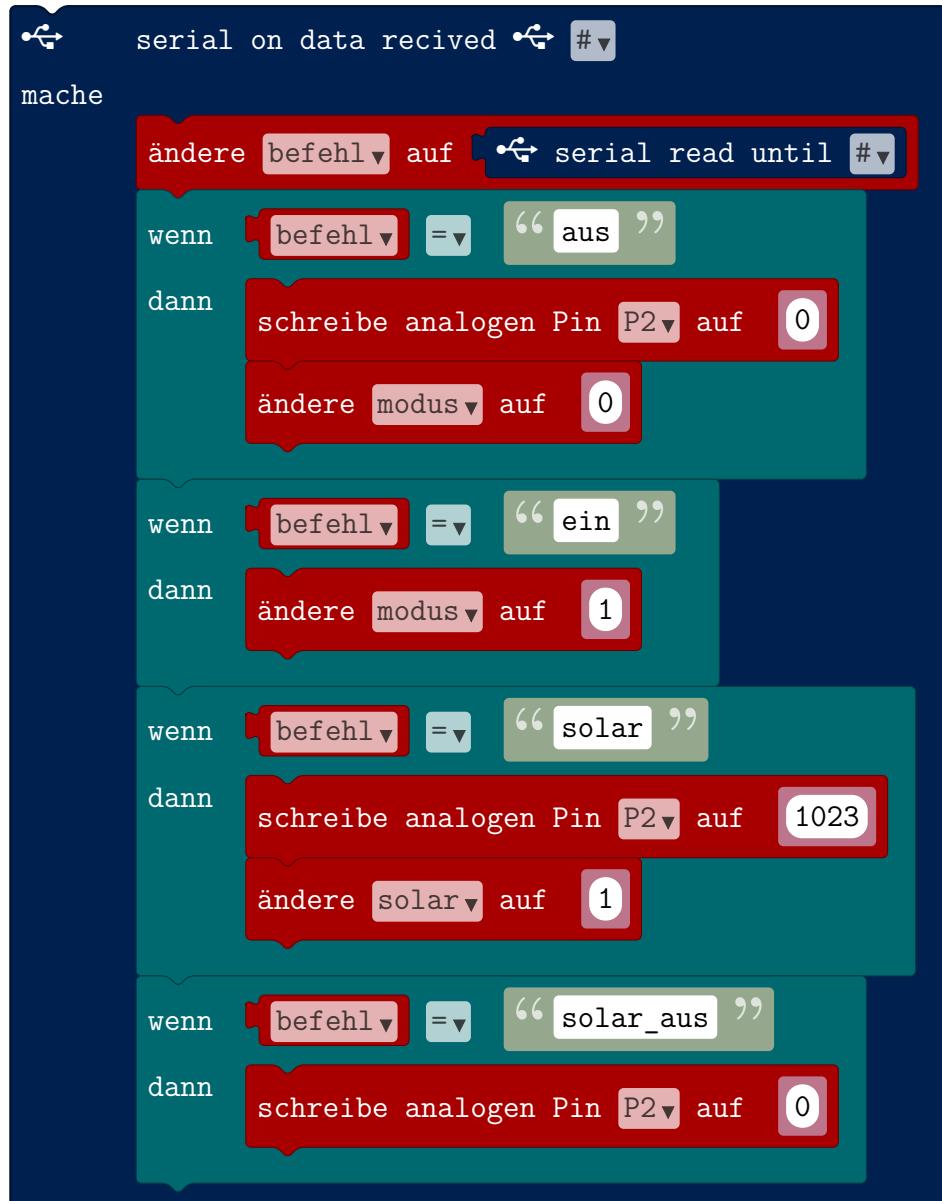
Lösung: Erweitere den Inhalt des Nodes um Whitespace/Phantome oder setzte `minimum width` für den Node.

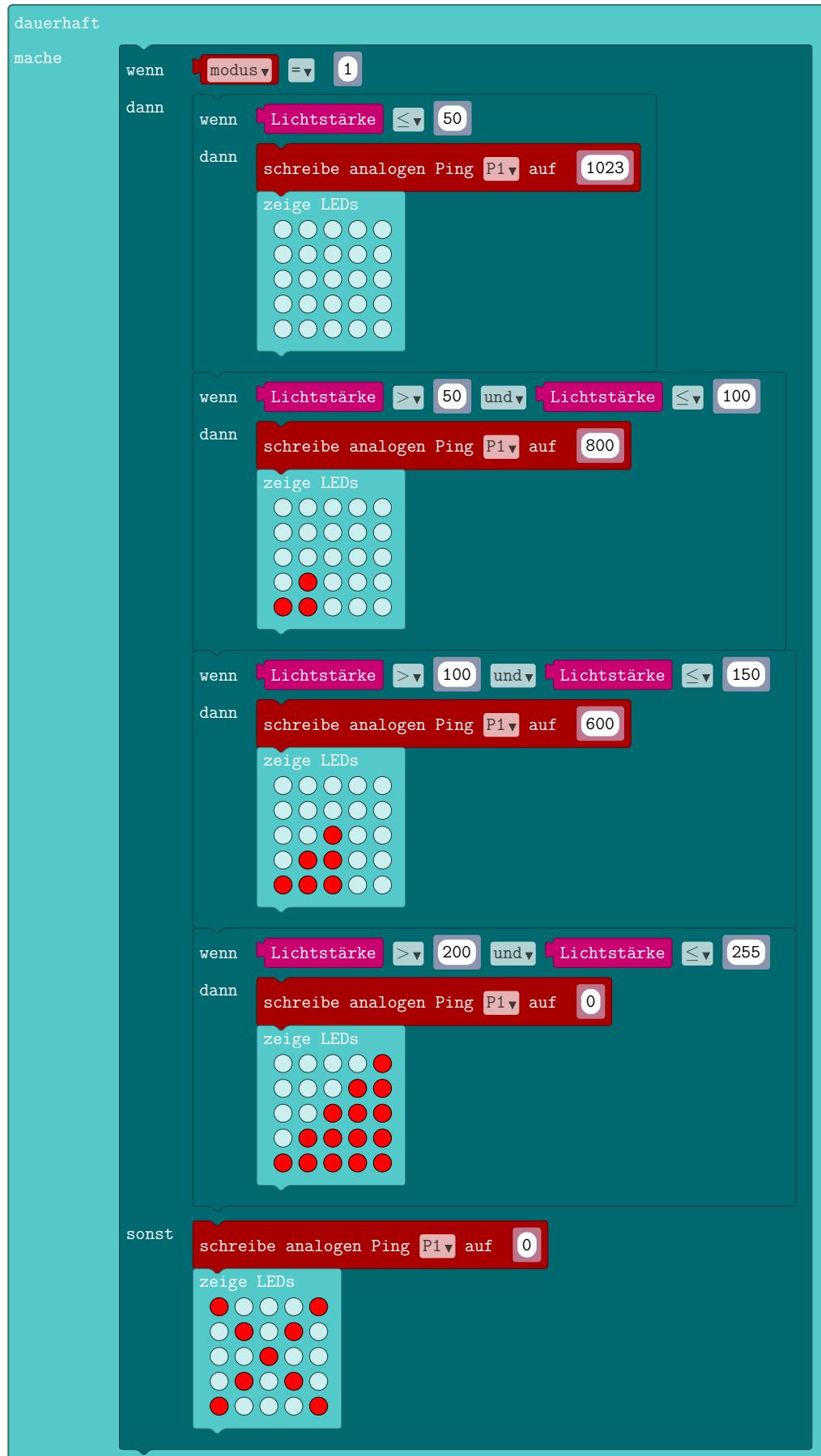
```
1 || \node [logik](one){2~~};  
2 || \node [logik, minimum width=1cm](one){3};
```

7 Mehr Beispielcode

7.1 Bsp: Calliope Smart Home







```

1 || \begin{tikzpicture}[codeblocks]
2 \einruecken{\node[grundlage,pinlow] (start){beim Start};}
3   \node[platzhalter,unter={start}{1}{0}] (plz1){ändere \dropdown{modus} auf \intbox{1}};
4   \node[konsole,unter={plz1}{0}{0}] (ser1) {serial \\\ redirect to
      \\ \hspace{5em} TX \dropdown{C17}\\ \hspace{5em} RX \dropdown{C16}\\ at baud rate \dropdown{9600}~};
5 \end{tikzpicture}
6
7 \begin{tikzpicture}[codeblocks]
8 \wenndann[eingabe]
9   {Knopf \dropdown{A} gedrückt}
10  {
11    \node[konsole] (ser1) {serial write line \stringbox{test\_line}};
12    \node[steuerung, unter={ser1}{0}{0}] (wait1){Warte \$\mu\$ \
      intbox{300000}};
13  }
14  {buttonA}
15 \end{tikzpicture}
16
17
18 \begin{tikzpicture}[codeblocks]
19
20 \schleife[konsole]{\usb{}}{serial on data received \usb{} \
  dropdown{\#}}
21   \node[platzhalter, unter={start}{1}{0}] (plz1) {ändere \
  dropdown{befehl} auf \tikz\node[konsole,boden,keinezacken, \
  puzzleteil,]{\usb{} serial read until \dropdown{\#}};};
22 \wenndann[unter={plz1}{0}{0}]{\tikz\node[pins,boden,keinezacken, \
  puzzleteil]{\dropdown{befehl}}; \dropdown{=} \stringbox{aus}};
23   \node[pins] (pin1) {schreibe analogen Pin \dropdown{P2} auf \
  \intbox{0}};
24   \node[pins,unter={pin1}{0}{0}] (pin2) {ändere \dropdown{modus} auf \intbox{0}};
25 \wenndann[unter={wenn1}{0}{0}]{\tikz\node[pins,boden, \
  keinezacken,puzzleteil]{\dropdown{befehl}}; \dropdown{=} \
  stringbox{ein}};
26   \node[pins] (pin2) {ändere \dropdown{modus} auf \intbox{1}};
27 \wenndann[unter={wenn2}{0}{0}]{\tikz\node[pins,boden, \
  keinezacken,puzzleteil]{\dropdown{befehl}}; \dropdown{=} \
  stringbox{solar}};
28   \node[pins] (pin3) {schreibe analogen Pin \dropdown{P2} auf \
  intbox{1023}};
29   \node[pins,unter={pin3}{0}{0}] (pin2) {ändere \dropdown{solar} \
  auf \intbox{1}};
30 \wenndann[unter={wenn3}{0}{0}]{\tikz\node[pins,boden, \
  keinezacken,puzzleteil]{\dropdown{befehl}}; \dropdown{=} \
  stringbox{solar\_aus}};
31   \node[pins] (pin4) {schreibe analogen Pin \dropdown{P2} auf \
  intbox{0}};
32 \wenndann[unter={wenn4}{0}{0}]{\tikz\node[pins,boden, \
  keinezacken,puzzleteil]{\dropdown{befehl}}; \dropdown{=} \
  stringbox{solar\_aus}};
33   \node[pins] (pin4) {schreibe analogen Pin \dropdown{P2} auf \
  intbox{0}};
34 \wenndann[unter={ende Schleifenrumpf}{0}{0}]{\tikz\node[pins,boden, \
  keinezacken,puzzleteil]{\dropdown{befehl}}; \dropdown{=} \
  stringbox{solar\_aus}};
35
36 }%ende Schleifenrumpf

```

```

37 || {start}
38 \end{tikzpicture}
39
40
41 \begin{tikzpicture}[codeblocks, scale=0.72]
42 \schleife[grundlage, keinezacken]{dauerhaft}{}{
43   \wenndannsonst[robertashape]{\tikz\node[platzhalter, boden,
44     keinezacken, puzzleteil]{\dropdown{modus}}; \dropdown{=} \
45     \intbox{1}} %wenn außen
46   {
47     \wenndann{\tikz\node[eingabe, boden, keinezacken, puzzleteil]{%
48       Lichtstärke}; \dropdown{$\leq$}\intbox{50}}
49     {\node[pins](pin1){schreibe analogen Ping \dropdown{P1} auf \
50       \intbox{1023}};
51       \node[grundlage, unter={pin1}{0}{0}](bild1){zeige LEDs\\\
52         bild{\emptyled}};
53     }{wenn1}
54
55     \wenndann[unter={wenn1}{0}]{\tikz\node[eingabe, boden,
56       keinezacken, puzzleteil]{Lichtstärke}; \dropdown{$>$}\intbox{50} \
57       \dropdown{und} \tikz\node[eingabe, boden,
58       keinezacken, puzzleteil]{Lichtstärke}; \dropdown{$\leq$}\intbox{100}}{
59       \node[pins](pin2){schreibe analogen Ping \dropdown{P1} auf \
60       \intbox{800}};
61       \node[grundlage, unter={pin2}{0}{0}](bild2){zeige LEDs\\\
62         \begin{array}{c} \text{\scriptsize 0} \text{\scriptsize 0} \text{\scriptsize 0} \text{\scriptsize 0} \\ \text{\scriptsize 0} \text{\scriptsize 0} \text{\scriptsize 0} \text{\scriptsize 0} \\ \text{\scriptsize 0} \text{\scriptsize 0} \text{\scriptsize 0} \text{\scriptsize 0} \\ \text{\scriptsize X} \text{\scriptsize 0} \text{\scriptsize 0} \text{\scriptsize 0} \\ \text{\scriptsize X} \text{\scriptsize X} \text{\scriptsize 0} \text{\scriptsize 0} \end{array} \\
63     }{wenn2}
64
65     \wenndann[unter={wenn2}{0}]{\tikz\node[eingabe, boden,
66       keinezacken, puzzleteil]{Lichtstärke}; \dropdown{$>$}\intbox{100} \
67       \dropdown{und} \tikz\node[eingabe, boden,
68       keinezacken, puzzleteil]{Lichtstärke}; \dropdown{$\leq$}\intbox{150}}{%
69       \node[pins](pin3){schreibe analogen Ping \
70       \dropdown{P1} auf \intbox{600}};
71       \node[grundlage, unter={pin3}{0}{0}](bild3){zeige LEDs\\\
72         \begin{array}{c} \text{\scriptsize 0} \text{\scriptsize 0} \text{\scriptsize 0} \text{\scriptsize 0} \\ \text{\scriptsize 0} \text{\scriptsize 0} \text{\scriptsize 0} \text{\scriptsize 0} \\ \text{\scriptsize 0} \text{\scriptsize 0} \text{\scriptsize X} \text{\scriptsize 0} \text{\scriptsize 0} \\ \text{\scriptsize 0} \text{\scriptsize X} \text{\scriptsize X} \text{\scriptsize 0} \text{\scriptsize 0} \\ \text{\scriptsize X} \text{\scriptsize X} \text{\scriptsize X} \text{\scriptsize 0} \text{\scriptsize 0} \end{array} \\
73     }{wenn3}
74
75     \wenndann[unter={wenn3}{0}]{\tikz\node[eingabe, boden,
76       keinezacken, puzzleteil]{Lichtstärke}; \dropdown{$>$}\intbox{150} \
77       \dropdown{und} \tikz\node[eingabe, boden,
78       keinezacken, puzzleteil]{Lichtstärke}; \dropdown{$\leq$}\intbox{200}}{%
79       \node[pins](pin4){schreibe analogen Ping \
80       \dropdown{P1} auf \intbox{200}};
81       \node[grundlage, unter={pin4}{0}{0}](bild4){zeige LEDs\\\
82         \begin{array}{c} \text{\scriptsize 0} \text{\scriptsize 0} \text{\scriptsize 0} \text{\scriptsize 0} \text{\scriptsize 0} \end{array} \\
83     }
84   }
85 }
```

```

74   \o \o \o \x \o \\
75   \o \o \x \x \o \\
76   \o \x \x \x \o \\
77   \x \x \x \x \o \\
78 }};}{wenn4}

79
80 \wenndann[unter={wenn3}{0}{0}]{\tikz\node[eingabe,boden,
81   keinezacken,puzzleteil]{Lichtstärke}; \dropdown{$>$}\intbox{200} \dropdown{und} \tikz\node[eingabe,boden,
82   keinezacken,puzzleteil]{Lichtstärke}; \dropdown{$\leq$}\intbox{255}}{\node[pins](pin5){schreibe analogen Ping \
83   dropdown{P1} auf \intbox{0}};
84 \node[grundlage,unter={pin5}{0}{0}](bild5){zeige LEDs\\
85   \bild{
86     \o \o \o \x \\
87     \o \o \x \x \\
88     \o \o \x \x \x \\
89     \o \x \x \x \x \\
90     \x \x \x \x \x \\
91   }};}{wenn5}
92 \%dann außen
93 {\node[pins](pin6){schreibe analogen Ping \dropdown{P1} auf \
94   \intbox{0}};
95 \node[grundlage,unter={pin6}{0}{0}](bild5){zeige LEDs\\
96   \bild{
97     \x \o \o \o \x \\
98     \o \x \o \x \o \\
99     \o \o \x \o \o \\
100    \o \x \o \x \o \\
101    \x \o \o \o \x \\
102  }};}{sonst außen
103 {wenn}
104 }{schleife1}
105 \end{tikzpicture}

```