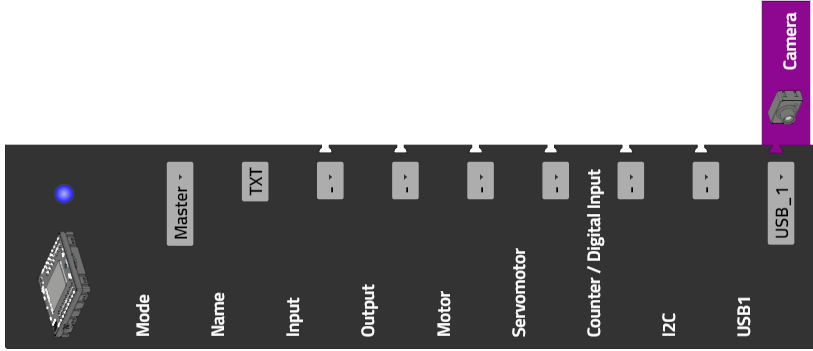
Lösungsblatt

# Aufgabe 4: Kodierung

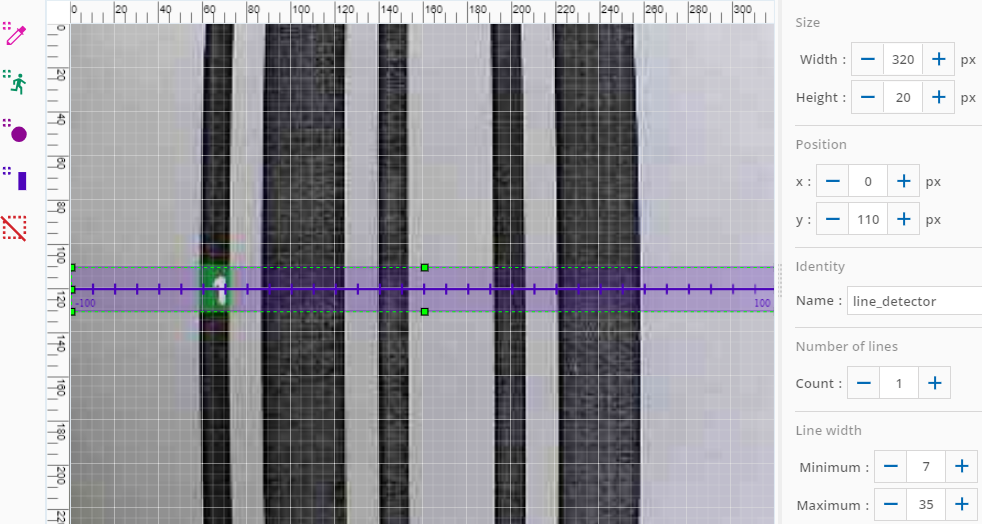
## Programmieraufgaben

**1. Balkenerkennung**

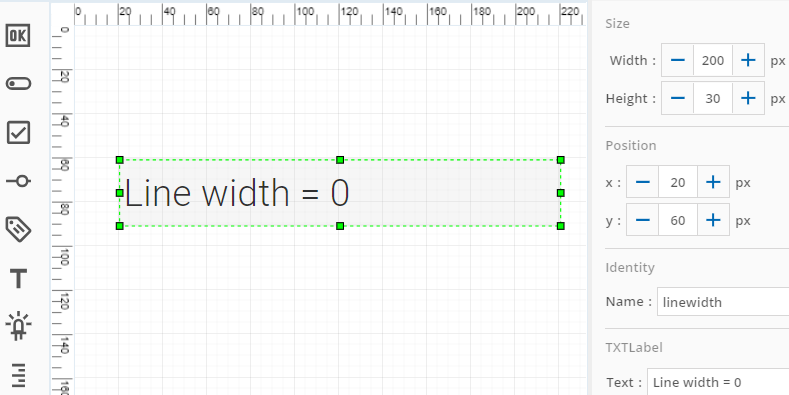
Konfiguration der USB-Kamera:



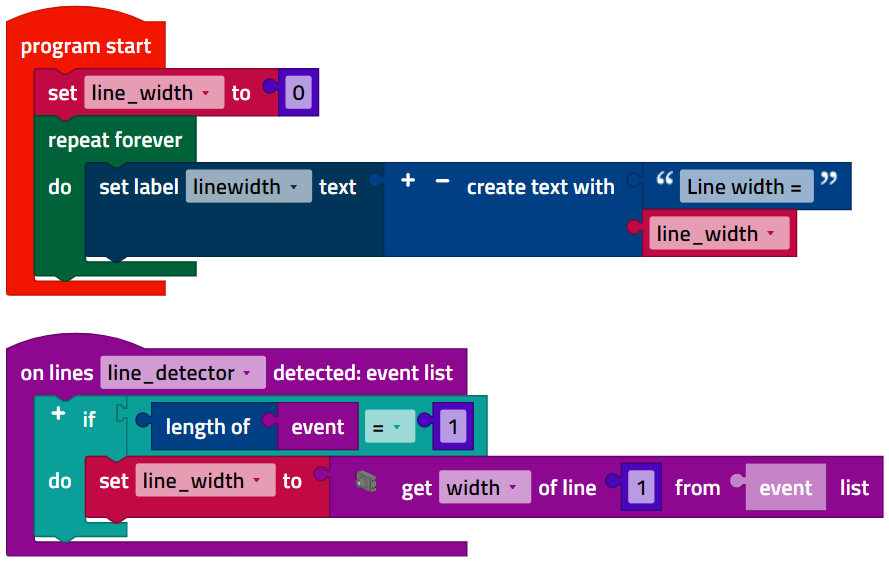
1a. Konfiguration der Linienerkennung:



Display-Konfiguration:

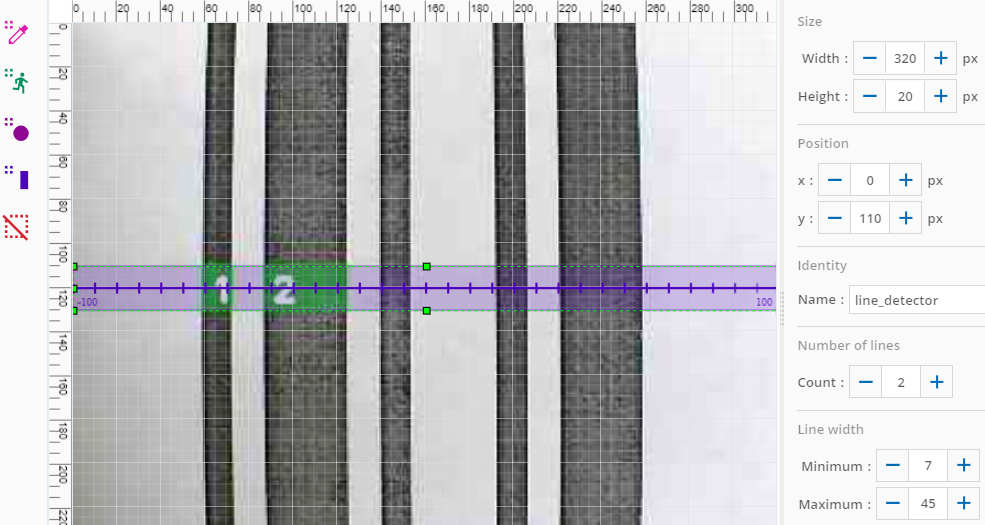


Programm (Beispiel): Bestimmung der Breite eines senkrechten schwarzen Balkens

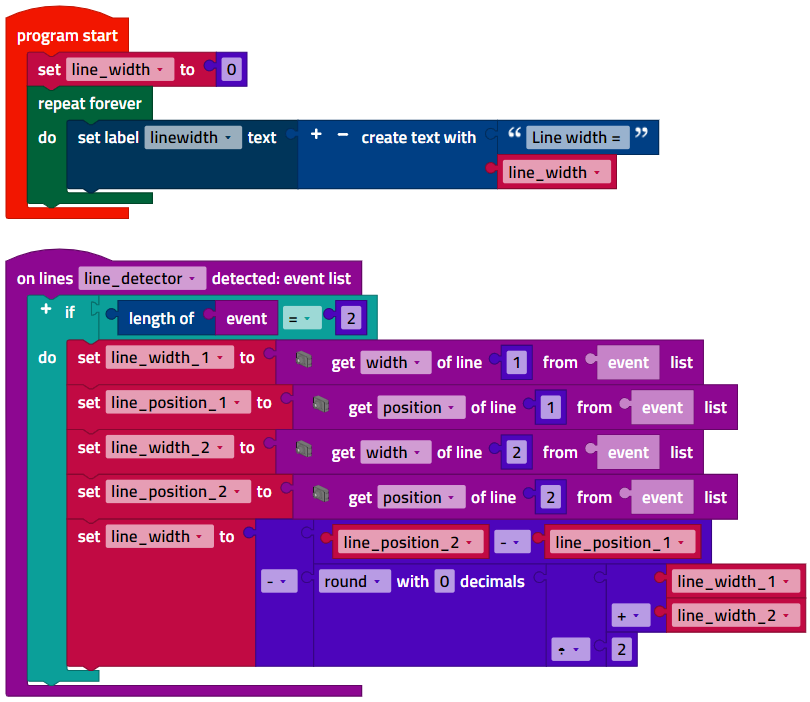


*Line\_Detection.ft*

1b. Konfiguration der Linienerkennung:



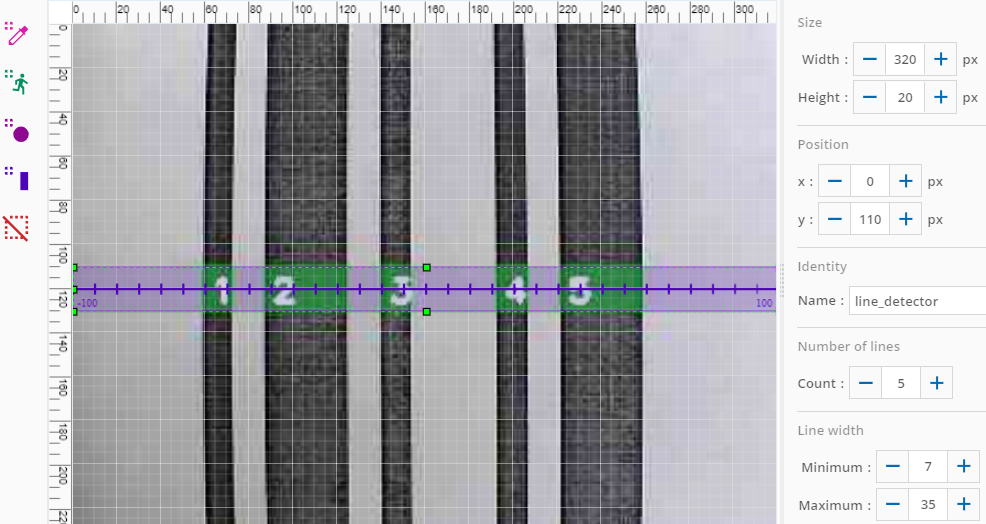
Programm (Beispiel): Bestimmung der Breite des ersten weißen Zwischenraums:



*Bar\_Width\_Recognition.ft*

**2. 9-stelliger Barcode**

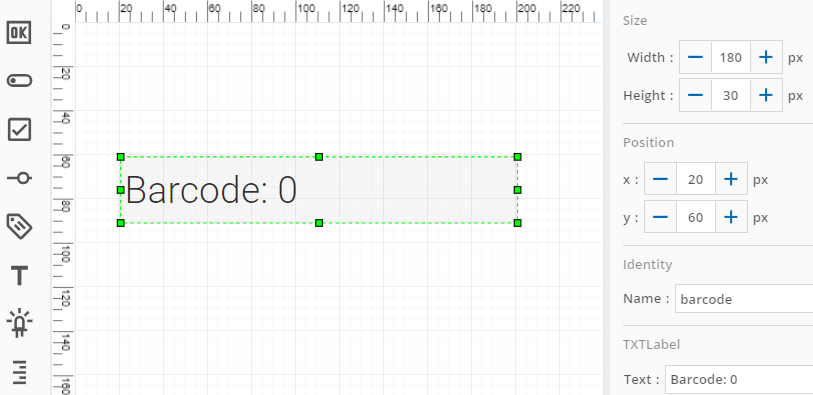
Konfiguration der Erkennung von fünf Linien:



Die erkannten (schwarzen) Balken des Codes werden von links nach rechts durchgezählt. Zur Auswertung wird der Code zunächst als Binärcode interpretiert: Ein breiter (schwarzer oder weißer) Balken steht für eine 1, ein schmaler für eine 0.

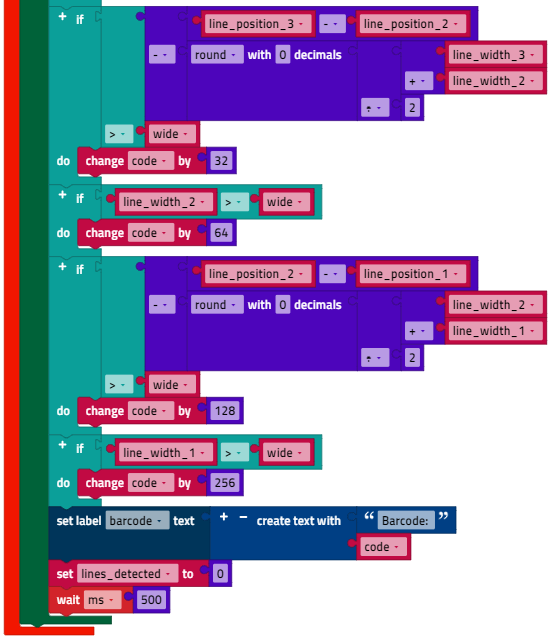
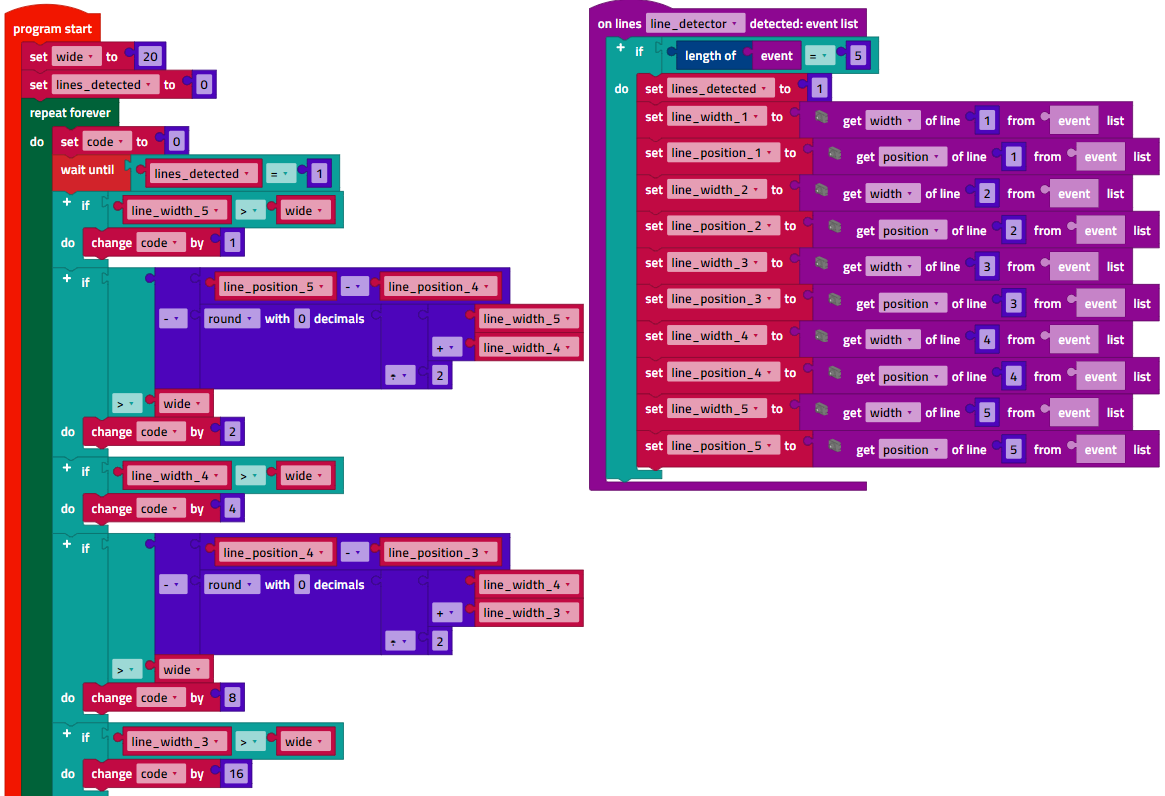
Der in der Abbildung gezeigte Code entspricht also der neunstelligen Binärzahl 001001001b = 73.

Display-Konfiguration:



Programm (Beispiel): Decodieren eines neunstelligen Barcodes

Die Variable „lines\_detected“ dient hier als Semaphor: Sie signalisiert dem Hauptpro­gramm, dass die Linienerkennung fünf Linien erkannt hat:



*Barcode\_Decoder.ft*

**3. Code-39-Dekodierer**

Mithilfe der Listen aus dem Programm-Template „Code39\_Template.ft“ lässt sich der binäre Wert des Codes nun mit einer zusätzlichen Programmzeile und einem modifi­zierten Ausgabebefehl in das kodierte Zeichen „übersetzen“.

Programmauszug (Beispiel):



*Code39\_Decoder.ft*

**4. Fehlererkennung**

Mit dem Code 39 lassen sich 40 verschiedene Zeichen kodieren:

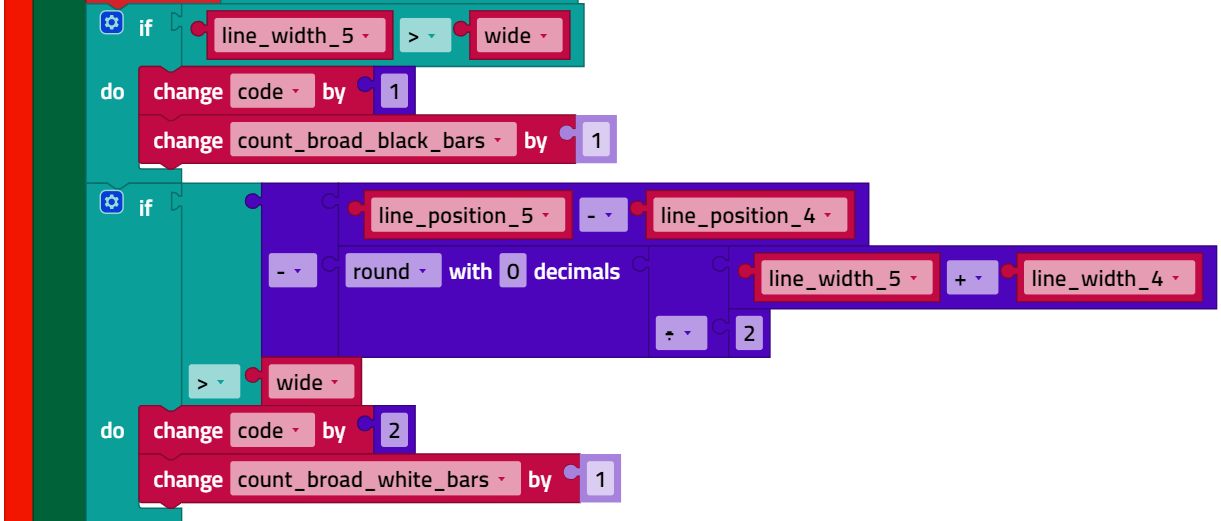
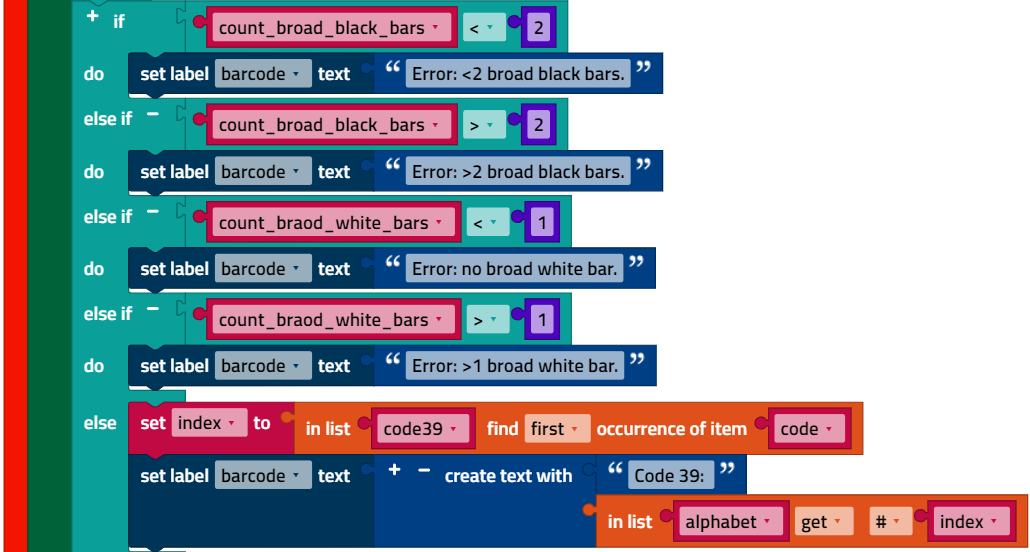
* zwei breite und drei schmale schwarze Linien erlauben 10 verschiedene Kombinationen, und bei jeder dieser Kombinationen
* kann die breite weiße Linie an einer von vier Positionen stehen.

Damit hat der Code einen Informationsgehalt (= die minimale Anzahl an Bits, die benötigt werden, um 40 Zeichen zu kodieren) von 6 bit (26 = 64).

Mit einem 9-bit-Code könnte man die 8-fache Anzahl Zeichen darstellen (29 = 512). Der Code hat somit einen relativ geringen Informations­gehalt und eine sehr hohe Redundanz.

Die große Redundanz des Codes macht ihn sehr robust. Ungültige Codeworte lassen sich sehr leicht durch Zählen der breiten schwarzen und der breiten weißen Balken erkennen.

Programmauszüge (Beispiel):

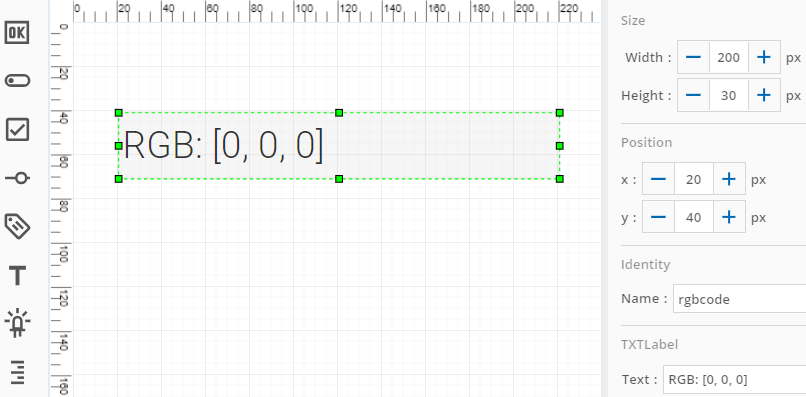
…  
  
…  


*Code39\_Decoder\_with\_Error\_Detection.ft*

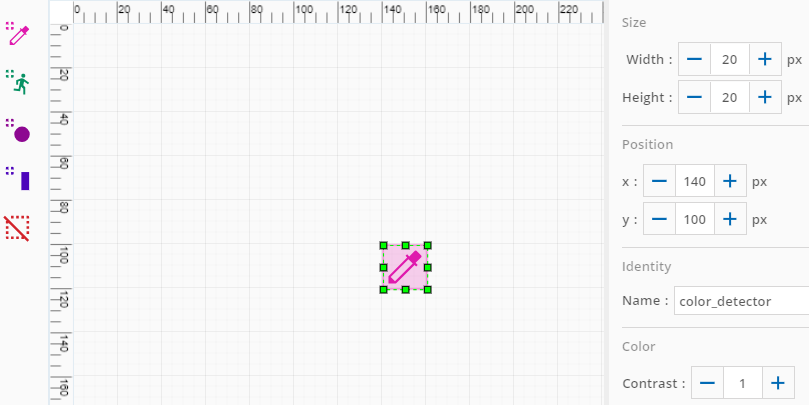
## Experimentieraufgaben

**1. RGB-Code**

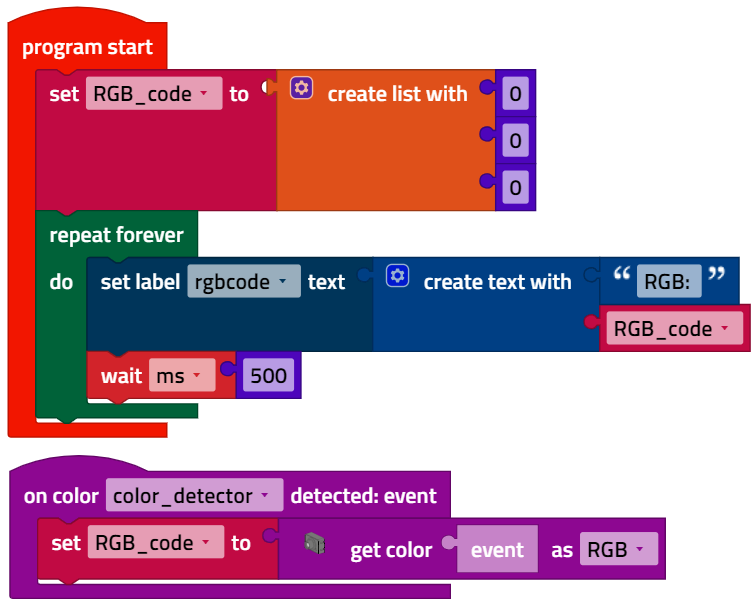
1a. Display-Konfiguration:



Konfiguration Farberkennung:

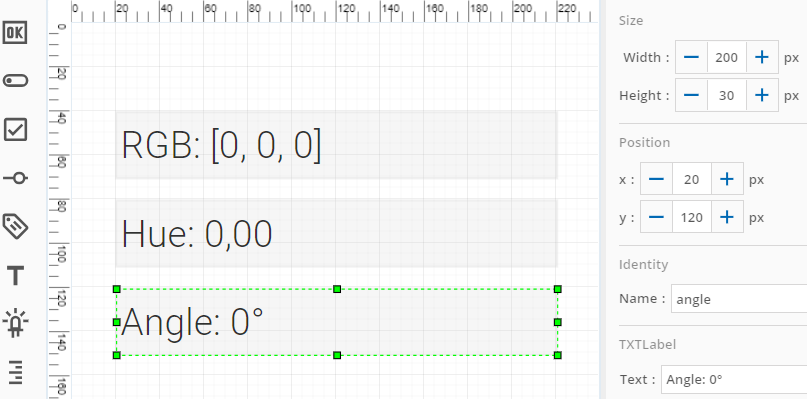


Programm (Beispiel):

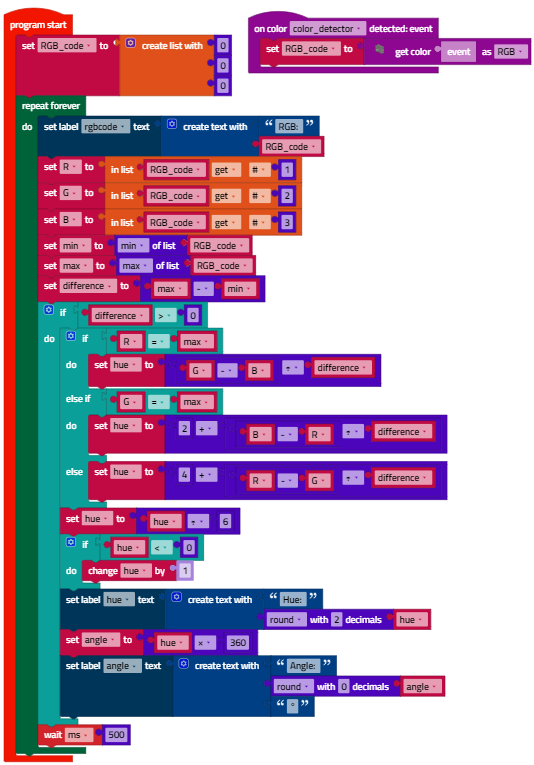


*RGB\_Colorcode.ft*

1b. Display-Konfiguration:



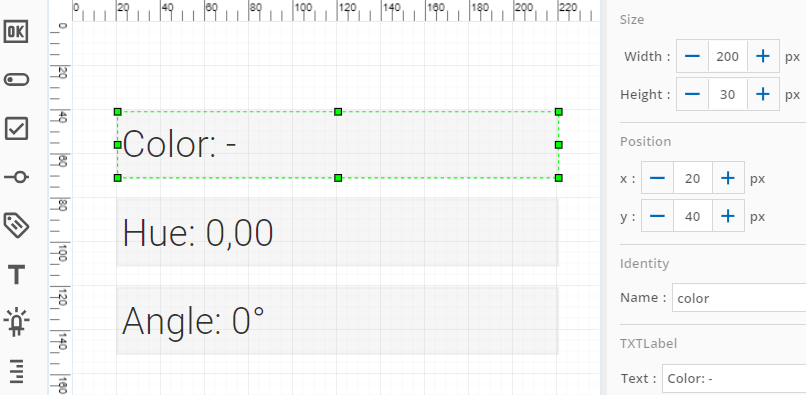
Programm (Beispiel):



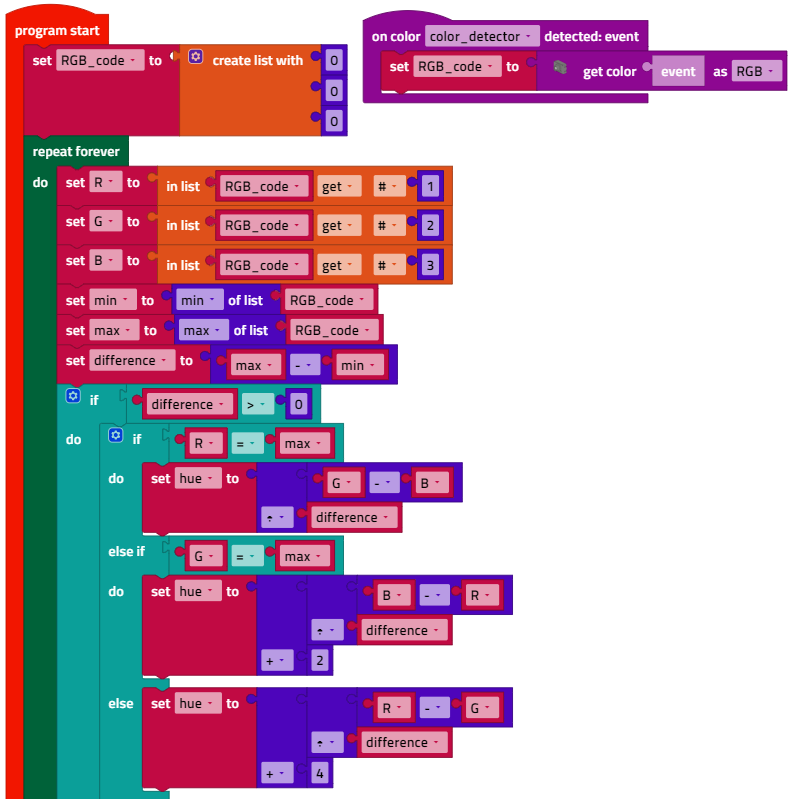
*RGB\_Decoder.ft*

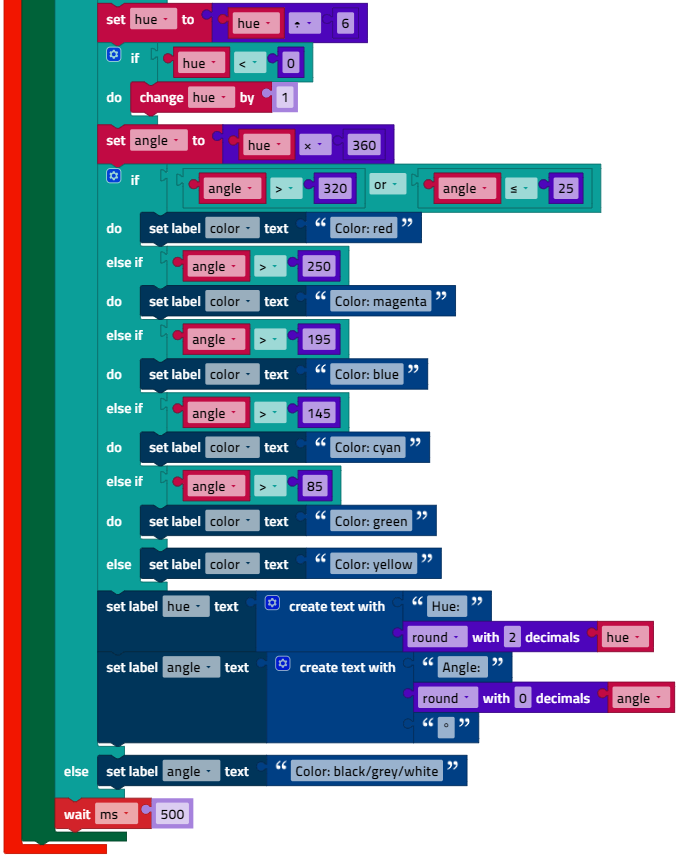
**2. Farberkennung**

Display-Konfiguration:



Programm (Beispiel): Farben des Farbsterns erkennen





*Camera\_Color\_Decoder.ft*

**3. Mehrdimensionaler Code**

Beispiel-Code (Colored Barcode – Color/Width, CW)

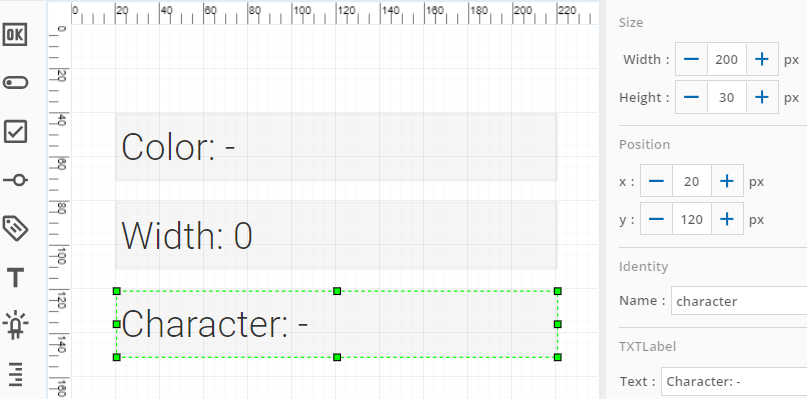
Der CW-Code unterscheidet sechs Farben (rot, gelb, grün, cyan, blau, magenta) und sechs Balkenbreiten (1-6), kann also 36 Zeichen kodieren:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **A** | **B** |
| **Breite** | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 |
| **Farbe** | 1 | 2 | 3 | 4 | 5 | 6 | 1 | 2 | 3 | 4 | 5 | 6 |
|  | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **J** | **K** | **L** | **M** | **N** |
| **Breite** | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 4 |
| **Farbe** | 1 | 2 | 3 | 4 | 5 | 6 | 1 | 2 | 3 | 4 | 5 | 6 |
|  | **O** | **P** | **Q** | **R** | **S** | **T** | **U** | **V** | **W** | **X** | **Y** | **Z** |
| **Breite** | 5 | 5 | 5 | 5 | 5 | 5 | 6 | 6 | 6 | 6 | 6 | 6 |
| **Farbe** | 1 | 2 | 3 | 4 | 5 | 6 | 1 | 2 | 3 | 4 | 5 | 6 |

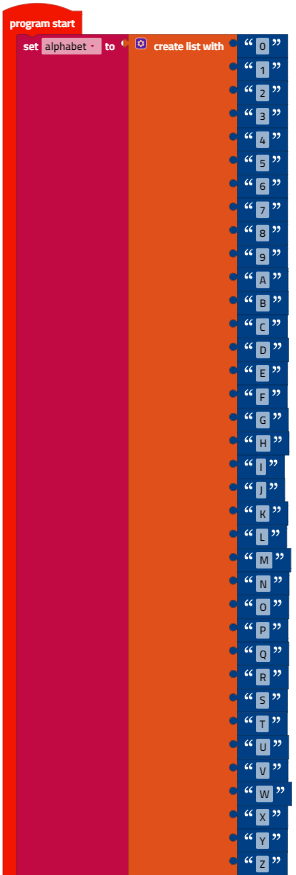
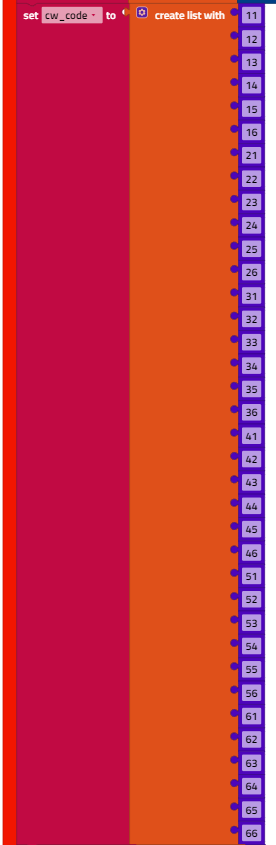
Die sechs Balkenbreiten werden in 5-mm-Schritten gewählt: 0,5 cm (= 1), 1 cm (= 2), …, 3 cm (= 6). Beim Abstand der Kamera im Barcodeleser-Modell entsprechen 0,5 cm etwa 15 Breitenpunkten.

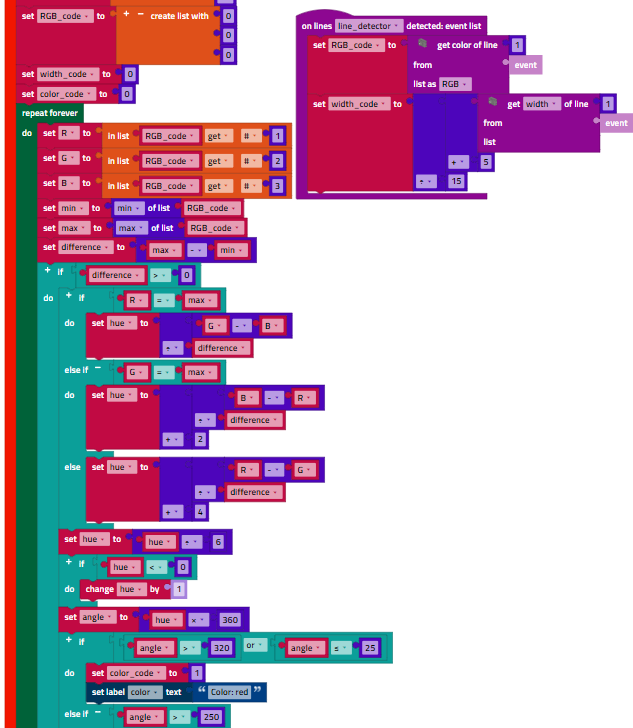
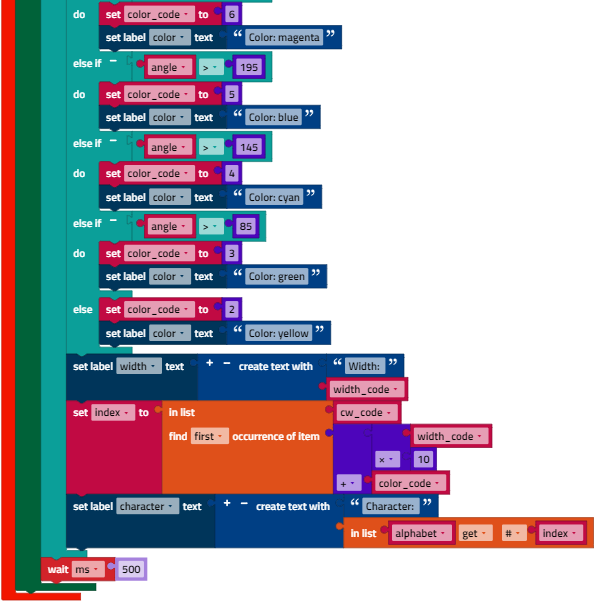
Die Umrechnung der Breitenpunkte in den Wert für die Kodierung gelingt also leicht, indem die Breitenpunkte durch 15 geteilt werden. Um Schwankungen bei der Erkennung auszugleichen, werden zuvor fünf Breitenpunkte hinzuaddiert.

Display-Konfiguration:



Die Darstellung des kodierten Werts er­folgt vereinfacht als Dezimalzahl: Die Bal­kenbreite entspricht der ersten, die Farbe der zweiten Ziffer. Ist einer der zurückge­lieferten Werte „0“ konnte der Balken nicht erkannt werden.

*Colored\_Barcode\_Decoder.ft*

Anlagen

# Aufgabe 4: Kodierung

## Erforderliches Material

* PC für Programmentwicklung, lokal oder über Web-Schnittstelle.
* USB-Kabel oder BLE- bzw. WLAN-Verbindung für die Übertragung des Programms auf den TXT4.0.
* Programm-Template (Code 39): Code39\_Template.ft
* Code-39-Beispiele zum Dekodieren (Breite: 6 cm)
* Farbkreis und Farbstern
* Papier und Farbstifte zum Design eines eigenen Codes

## Weiterführende Informationen

[1] Dominic Welsh: *Codes und Kryptographie*. VCH 1991.

[2] Wikipedia: [*Strichcode*](https://de.wikipedia.org/wiki/Strichcode), [*Code 39*](https://de.wikipedia.org/wiki/Code_39).

[3] RapidTables:[*RGB Color Codes Chart*](https://www.rapidtables.com/web/color/RGB_Color.html)*.*