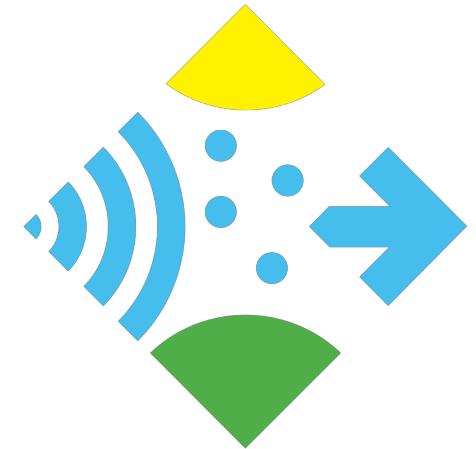


# senseBox

## Lernkarten



# Die senseBox Lernkarten

Die senseBox Lernkarten helfen dir beim Experimentieren mit der senseBox. Neben den Grundlagen der Informatik, erhältst du wichtige Informationen zum Verwenden und Aufbau der Komponenten der senseBox:edu.

Die Programmieroberfläche findest du unter  
[www.blockly.senseBox.de](http://www.blockly.senseBox.de)

Weitere Information unter: [www.senseBox.de](http://www.senseBox.de)

Björn Guntermann & Mario Pesch  
Institut für Geoinformatik – Universität Münster  
1. Auflage

Die Lernkarte sind als OER Veröffentlicht und auch zum Download auf unserer Website verfügbar.

Die Lernkarten sind in 5 verschiedene Kategorien unterteilt.

Kategorie	Beschreibung
GI	Karten in dieser Kategorie enthalten wichtige Grundlagen und Konzepte aus der Informatik.
SB	Diese Karten beschreiben die wichtigsten Bauteile und die Programmierung der senseBox.
A	Die Aufgabenkarten zu verschiedenen Projekten. Diese Karten geben wichtige Tipps zu den unterschiedlichen Projekten.
H	Manchmal sind die Tipps nicht ausreichend und es wird mehr Hilfe benötigt. Diese Karten geben Hilfestellungen zum Lösen von Probleme ohne direkt die komplette Lösung zu verraten.
L	Diese Karten sollten erst am Ende der Zeit ausgegeben werden da die komplette Lösung verraten wird.



# senseBox MCU – Gesamtübersicht der Anschlüsse

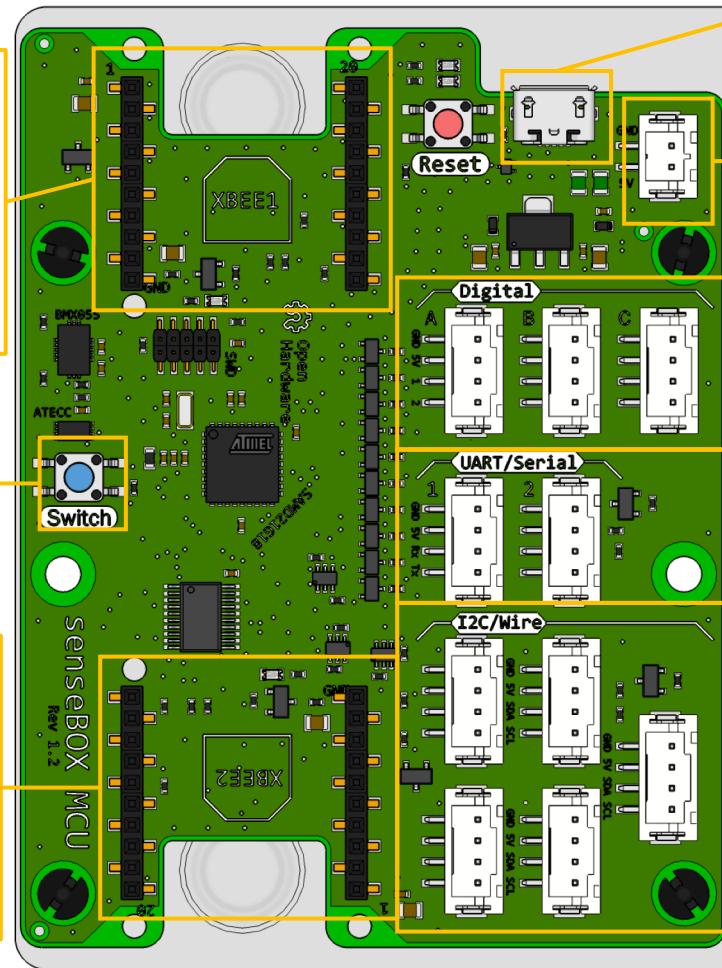
## XBEE-Steckplatz 1

XBEEs sind kleine Zusatzmodule um die senseBox um Funktionen wie WiFi oder eine SD-Karte zu erweitern. Schließe hier das WLAN-Modul an

## Button

## XBEE-Steckplatz 2

XBEEs sind kleine Zusatzmodule um die senseBox um Funktionen wie WiFi oder eine SD-Karte zu erweitern. Schließe hier das SD-Modul an



## USB-Anschluss

## Akku-Anschluss

## Digital/Analog-Ports

Hier werden Sensoren und Aktoren über das Breadboard angeschlossen.

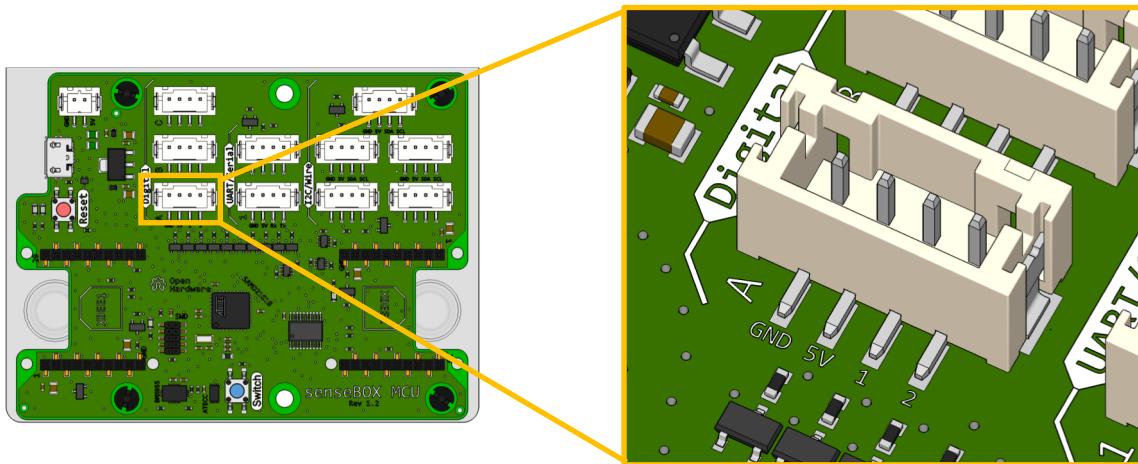
## UART/Serial-Ports

Hier wird der Feinstaubsensor angeschlossen

## I2C/Wire-Ports

Hier werden alle Umweltsensoren sowie das Display angeschlossen.

# Die senseBox und JST-Adapterkabel



Jeder **Digital/Analog-Port** auf der senseBox MCU verfügt über vier verschiedene Pins.

Der **GND-Pin** ist der Minuspol und ist immer mit dem schwarzen Kabel verbunden.

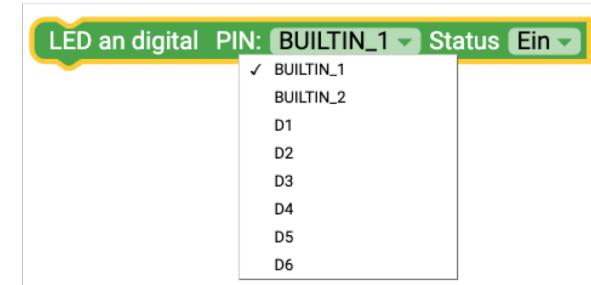
Der **5V-Pin** dient zur Stromversorgung der Sensoren und hat immer das rote Kabel.

Die mit **1** und **2** beschrifteten Pins sind die digitalen bzw. analogen Pins 1 und 2. Du wirst sehen, dass diese Nummerierung fortlaufend bis zum Pin 6 an Port Digital C zählt.

## Pins in den Blöcken einstellen

Damit deine eigenen Programme richtig funktionieren können, musst du in einigen Blöcken den Pin auswählen an dem dein Verbraucher (also z.B. eine LED) angeschlossen ist.

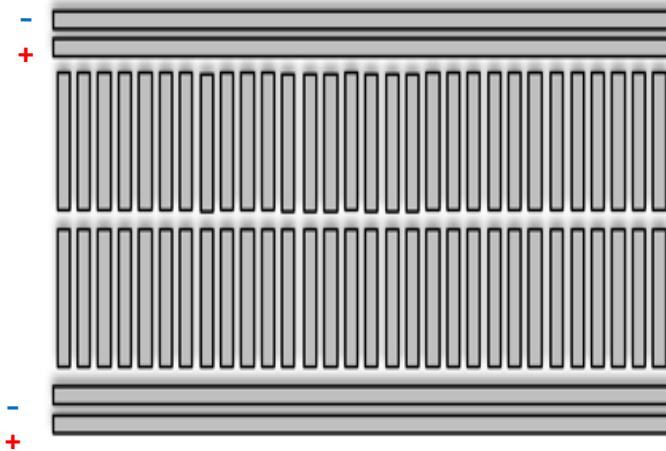
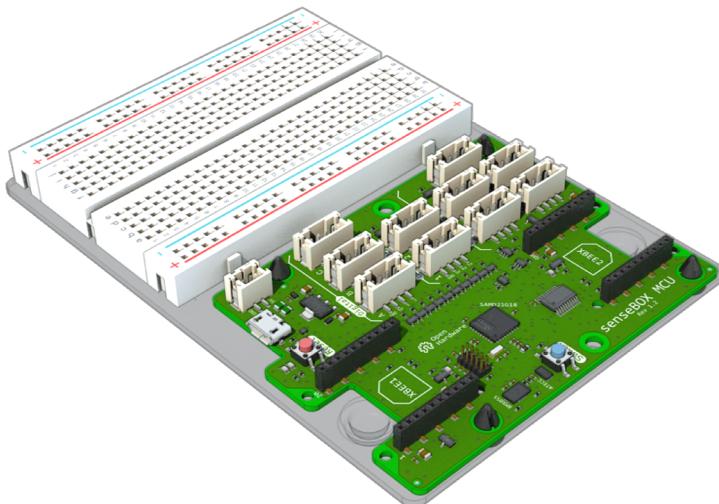
Für eine LED an Digital 1 würde der Block also wie folgt aussehen:



# Das Breadboard

Das Breadboard, oder auch Steckbrett, hilft dir Schaltungen auch ohne Löten sicher zu verbinden. Die elektronischen Bauteile werden einfach in die Federkontakte gesteckt, so kann eine Schaltung durch Umstecken schnell geändert werden.

Das Breadboard besteht aus zwei gespiegelten, nicht leitend verbundenen Seiten.



Diese bestehen jeweils aus zwei langen Reihen für die **Plus**- und **Minus** anschlüsse sowie zweimal 30 Reihen mit je fünf Federkontakten, die mit a bis e bzw. f bis j beschriftet sind.

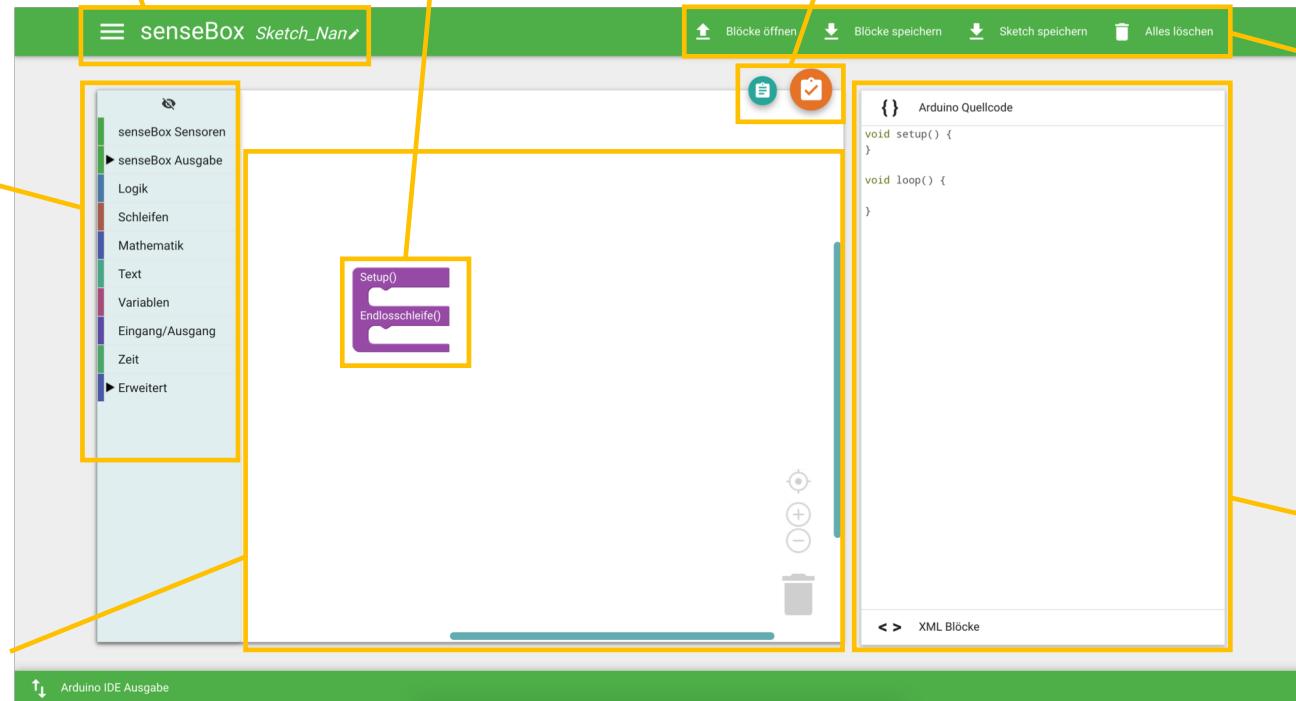
Die Plus- und Minusanschlüsse sowie die fünf Federkontakte einer Reihe sind, wie oben dargestellt, leitend miteinander verbunden.

# Die Programmieroberfläche

Hier kannst du zum einen auf das Einstellungsmenü von Blockly für senseBox zugreifen sowie den Namen deines Programmes (Sketches) ändern.

Hinter diesen Schaltflächen verbergen sich alle Blöcke die du zum Programmieren der senseBox benötigst.

Dies ist dein Arbeitsbereich. Hier setzt du dein Programm aus Blöcken zusammen



Dieser Block wird automatisch beim Starten der Oberfläche geladen. Die Setup-Funktion wird einmalig zum Programmstart ausgeführt. Die Endlosschleife wird durchgehend wiederholt.

Mit dieser Schaltfläche kannst du dein Programm komplizieren (in Maschinensprache umwandeln) sowie herunterladen um es auf deine senseBox-MCU zu übertragen.

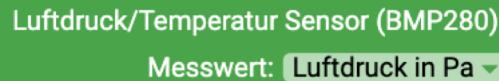
Diese Schaltfläche kopiert deinen Programmcode in die Zwischenablage.

Mit diesen Schaltflächen kannst du Projekte öffnen, speichern sowie dein aktuelles Projekt löschen.

In diesem Fenster wird dir dein Programmquellcode.

# Umweltsensoren auslesen

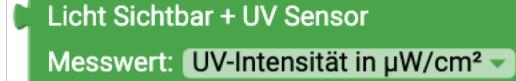
## Luftdrucksensor



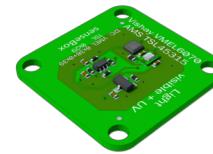
Dieser Block gibt dir den Messwert des Luftdrucksensors aus. Der Sensor kann neben dem Luftdruck auch die Temperatur messen. Den gewünschten Messwert kannst du im Dropdown Menü auswählen.



## Lichtsensor



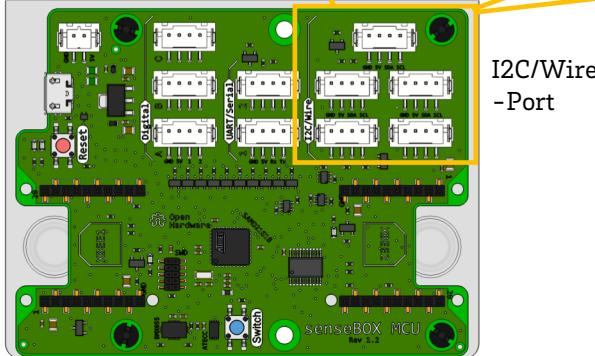
Dieser Block gibt dir den Messwert des Lichtsensors aus. Im Dropdown Menü kannst du den gewünschten Messwert auswählen. Der Sensor kann das sichtbare Licht (lux) und das UV-Licht messen.



## Temperatur- & Luftfeuchtigkeitssensor



Dieser Block gibt dir den Messwert des Temperatur- & Luftfeuchtigkeitssensors aus. Im Dropdown Menü kannst du den gewünschten Messwert auswählen.



I2C/Wire -Port

## Anschließen der Sensoren

Alle Umweltsensoren der senseBox werden über ein JST-Kabel an die I2C/Wire-Port angeschlossen. Mehr zu den JST-Kabeln findest du auf der Karte **Die senseBox und JST-Kabel**

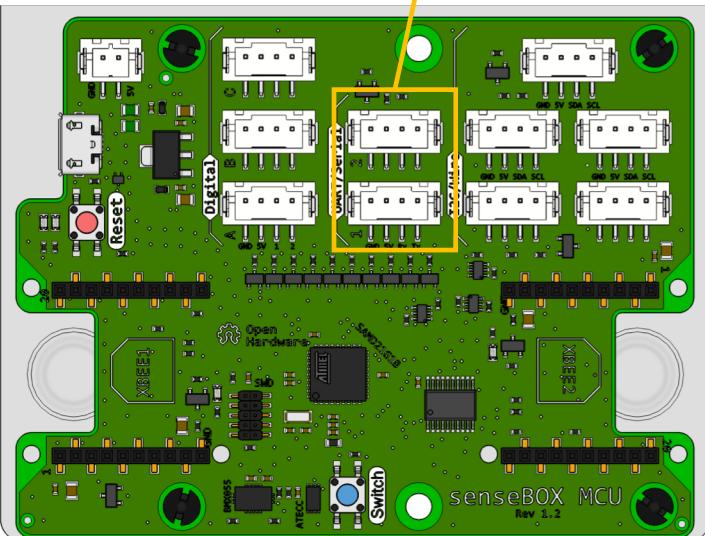
# Feinstaubsensor auslesen

## Anschluss des Sensors

Der Feinstaubsensor wird über das Feinstaubsensor-JST Kabel an einen der beiden UART/Serial Ports angeschlossen.



UART/Serial



## Feinstaubsensor auslesen

Feinstaubsensor

Messwert: PM2.5 in  $\mu\text{g}/\text{m}^3$  an Serial1

Dieser Block gibt dir den Messwert des Feinstaubensors. Im Dropdown Menü kannst du den gewünschte Messwert auswählen.

Der Feinstaubsensor kann Feinstaub in zwei verschiedenen Partikelgrößen gemessen werden.

**PM2.5:** Gibt dir den Anzahl der Feinstaubpartikel  $<2.5 \mu\text{m}$  in  $\mu\text{g}/\text{m}^3$  an

**PM10:** Gibt dir den Anzahl der Feinstaubpartikel  $<10 \mu\text{m}$  in  $\mu\text{g}/\text{m}^3$  an

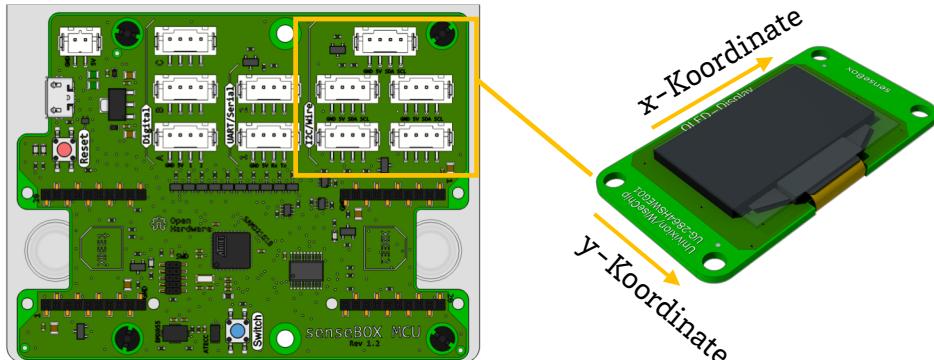
# Das Display

## Text anzeigen

Um das Display verwenden zu können muss es im *Setup()* initialisiert werden. Anschließend kannst du mit ein paar Blöcken in der *Endlosschleife()* Text oder auch Messwerte auf dem Display anzeigen lassen. Hier ein Beispiel für einen kurzen Text:

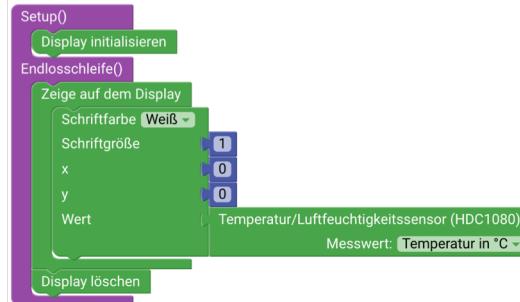


I2C/Wire-Stecker

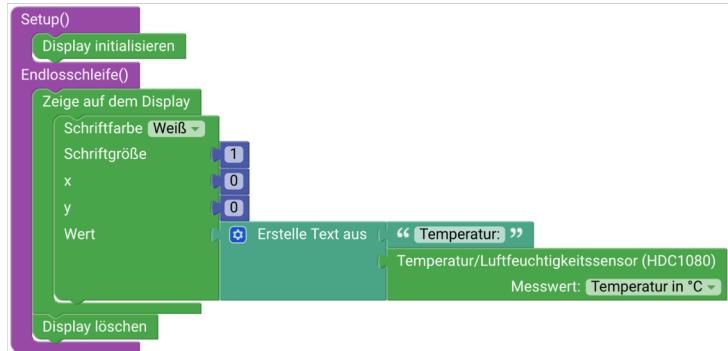


## Messwerte anzeigen und beschreiben

Um Messwerte auf dem Display anzeigen zu lassen verknüpfe entweder den Sensorblock direkt mit den Displayblöcken,

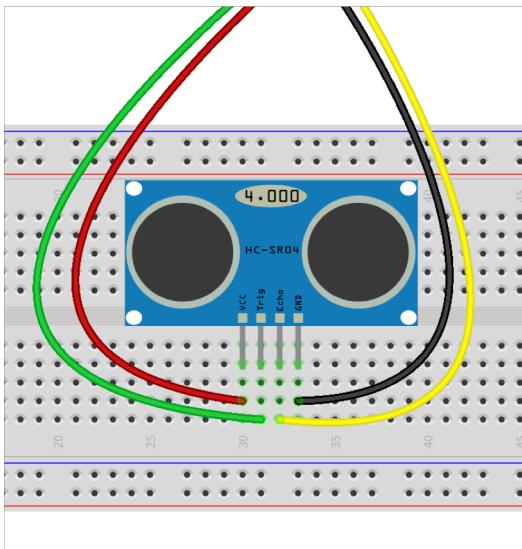
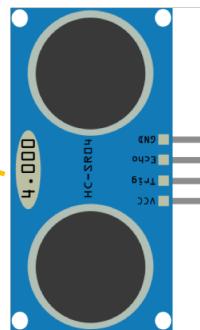
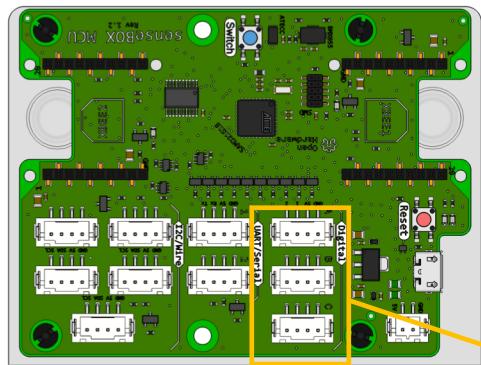


oder erstelle einen Text mit einer Beschreibung und dem Messwert.



**INFO:** Das Display hat eine Auflösung von 128x64 Pixel.

# Der Ultraschall-Distanzsensor

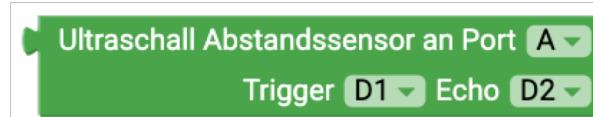


Zum Anschließen des Ultraschall-Distanzsensors benötigst du ein JST-Adapterkabel. Der Sensor selbst hat vier verschiedene Anschlüsse (Pins): VCC, Trig, Echo und GND. Diese vier Stecker müssen mit den vier Kabeln des JST-Adapterkabels verbunden werden.

GND	GND	(schwarzes Kabel)
Echo	2	(gelbes Kabel)
Trig	1	(grünes Kabel)
VCC	5V	(rotes Kabel)

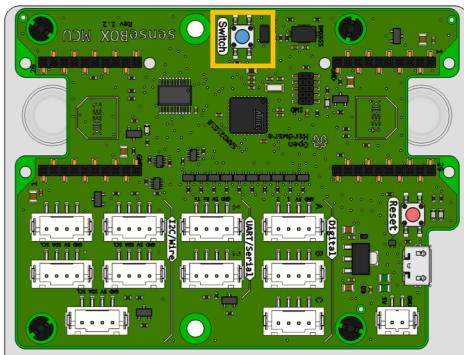
**Beachte:** Wenn du den Sensor an einen anderen Port anschließt ändert sich auch die Belegung für Trigger und Echo.  
Mehr zur Funktionsweise der JST-Adapterkabel findest du auf der Karte [Die senseBox und JST-Adapterkabel](#).

Der Ultraschall-Distanzsensor hat, genau wie alle anderen Sensoren, einen eigenen Block in dem definiert wird wo und wie der Sensor angeschlossen ist.

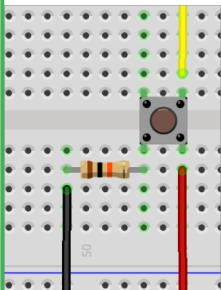


# Der Button

Die senseBox MCU hat eine Knopf aufgelötet, den du über einen Block abfragen kannst.



Du kannst auch noch weitere Knöpfe über das Breadboard anschließen.



**"ist gedrückt"**

Mit diesem Block kannst du abfragen ob der Block gerade gedrückt wird. Du erhältst entweder den Wert TRUE oder FALSE.

Drucktaster ist gedrückt ▾ PIN: on Board ▾

**"wurde gedrückt"**

Mit diesem Block kannst du abfragen ob der Block gedrückt wurde. Erst wenn der Knopf gedrückt und wieder losgelassen wurde erhältst du TRUE zurück

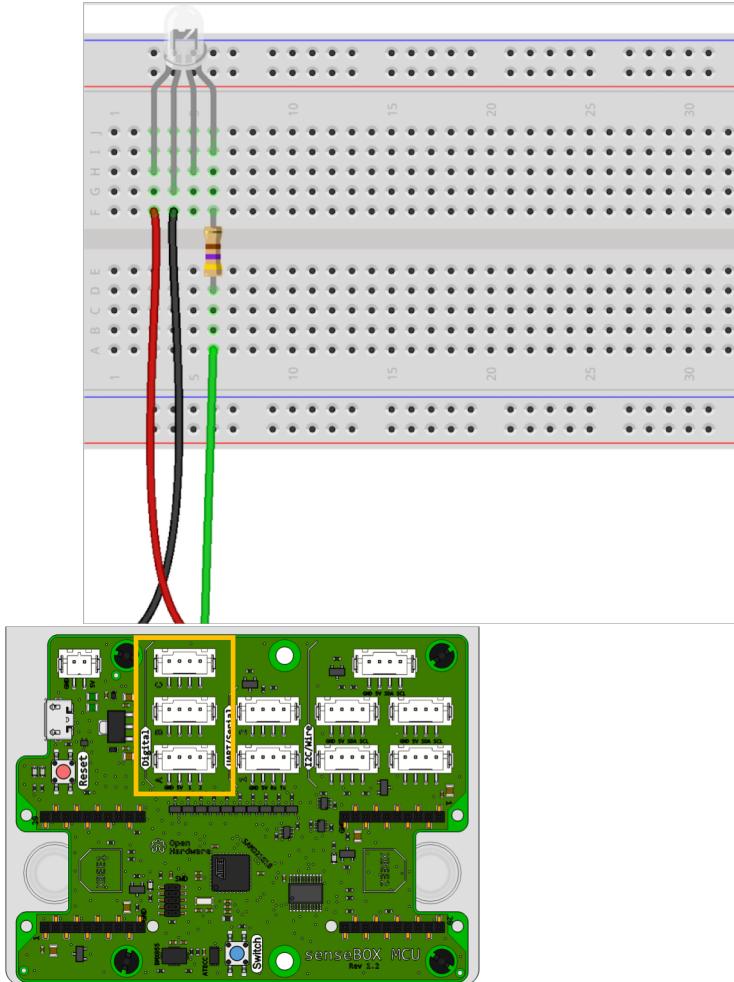
Drucktaster wurde gedrückt ▾ PIN: on Board ▾

**"als Schalter"**

Wenn du diesen Block verwendest kannst du den Knopf wie ein Lichtschalter verwenden. Der Status wird gespeichert bis der Button erneut gedrückt wird

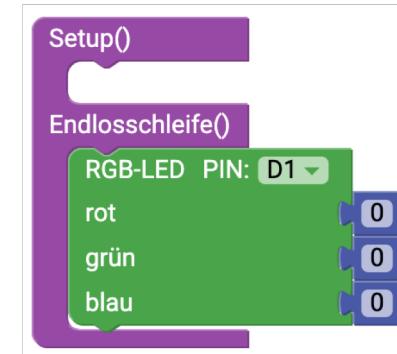
Drucktaster als Schalter ▾ PIN: on Board ▾

# Die RGB-LED



## RGB-LED ansteuern

Mit diesem Block kannst du die RGB-LED ansteuern. Wähle den korrekten PIN aus, über den du die RGB-LED angeschlossen hast.



RGB, steht für Rot, Grün, Blau und alle weiteren Farben kannst du durch mischen erstellen. Die Werte liegen zwischen 0 und 255. Möchtest du die LED vollständig ausschalten weißt du jeder Farbe den Wert 0 zu.

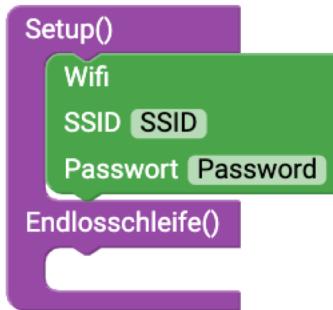
# Übertragen an die openSenseMap

## Wifi Verbindung herstellen

Verbinde das Wifi Bee mit dem Xbee-Steckplatz 1

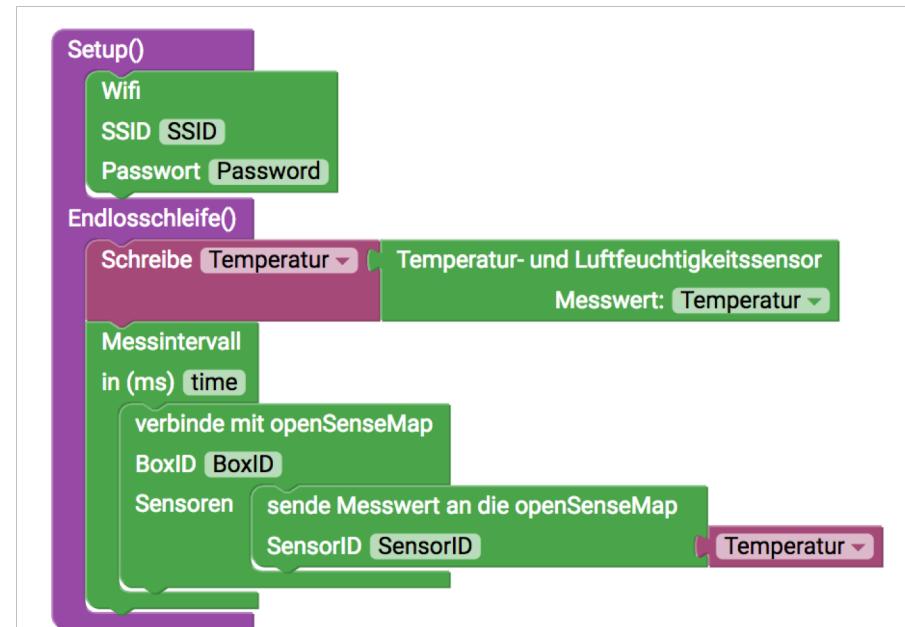


Anschließend musst du den WiFi Block ins Setup ziehen und deinen Netzwerknamen (SSID) und das Passwort angeben.



## Senden an die openSenseMap

Nach der Registrierung deiner senseBox auf der openSenseMap erhältst du eine BoxID und für jeden Sensor eine SensorID. Trage nun die BoxID in den „verbinde mit openSenseMap Block“ und die SensorID in den „sende Messwert an die openSenseMap“-Block ein. Das Messintervall legt fest wie häufig gemessen werden soll.



# Speichern auf SD-Karte

## Datei erstellen

Verbinde zuerst das SD-Bee mit dem Xbee-**Steckplatz 2**.



Anschließend muss in der Setup( ) Schleife eine Datei erstellt werden. Den Block hierzu findest du in der Kategorie senseBox Output – SD

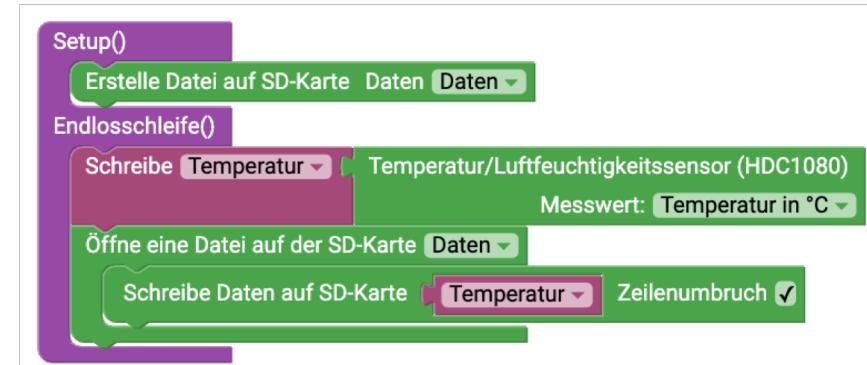


Um Daten auf die SD-Karte zu speichern muss die Datei immer geöffnet werden. Nach dem Schreiben der Messwerte kann die Datei wieder geschlossen werden.

## Messwerte in die Datei schreiben

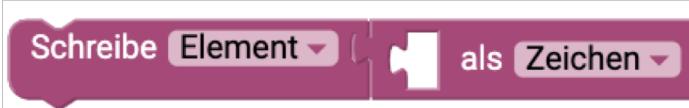
Zuerst muss der gemessene Temperaturwert in eine Variable geschrieben werden. Um mehr über Variablen zu erfahren schaue dir die Karte **GI1** an.

Um nun die Variable in der Datei zu speichern muss diese zuerst mit dem „Öffne-Datei auf SD-Karte“-Block geöffnet werden und anschließend die Variable mit dem „Schreibe Daten auf SD-Karte“-Block in die Datei geschrieben werden. Der „Öffne-Datei auf SD-Karte“-Block schließt nach dem Schreiben die Datei automatisch.



# Variablen – Platzhalter

**Variablen** oder auch Platzhalter genannt werden in der Informatik für ganz viele verschiedene Dinge genutzt. Sie sind eine Art Kiste, die mit einem Name versehen ist, in diese Kiste kannst du nun verschiedene Dinge hinterlegen, z.B. Zahlen oder auch Texte, und diese später erneut abrufen.



## Variablen – Datentypen

Je nachdem, was du in einer Variable speichern möchtest musst du den richtigen Datentyp auswählen.

**Zeichen:** Für einzelne Textzeichen

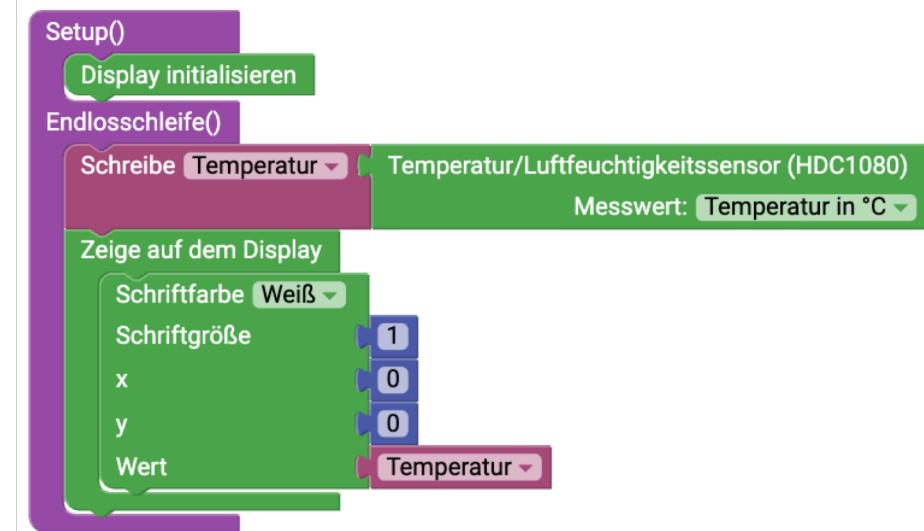
**Text:** Für komplette Sätze oder Wörter

**Zahl:** Für Zahlen zwischen -32768 und 32768

**Große Zahl:** Für große Zahlen zwischen -2147483648 und 2147483647

**Dezimalzahl:** Für Kommazahlen (z.B. 25,56)

**Variablen** können ihren Wert im Laufe des Programms verändern, so dass du zum Beispiel der Variable „Temperatur“ immer die aktuell gemessene Temperatur zuweist.



# Wenn-Dann-was?

Die „Wenn-Dann Bedingung“ ist beim Programmieren eine der wichtigsten Kontrollstrukturen, die du kennenzulernen wirst.

Mithilfe des Wenn-Dann Bedingung kann die senseBox bestimmte Aktionen ausführen, wenn etwas bestimmtes (z.B. ein Knopf gedrückt wurde) passiert ist.



Hier siehst du den Programmierbefehl zur „Wenn-Dann-Bedingung“. Damit kannst du programmieren, was genau gemacht werden soll wenn eine Bedingung erfüllt ist.

## Logischer Vergleich



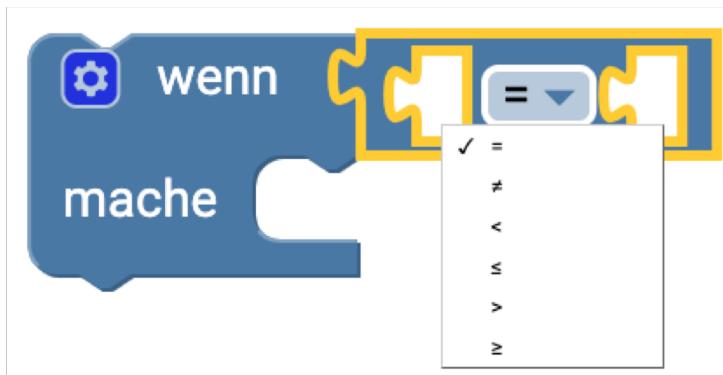
Mit diesem Block kannst du immer zwei Sachen miteinander vergleichen. Eine Erklärung für die verschiedenen Symbole findest du auf der Karte **Operatoren**.



Beispiel: **Wenn** die Temperatur größer als 20°C ist, **dann** wird die eingebaute LED angeschaltet.

# Operatoren

Operatoren werden in vielen Situationen beim Programmieren benötigt. Mithilfe der Operatoren können Bedingungen überprüft oder auch Werte verglichen werden.



Die folgenden Operatoren finden sich unter [Logik](#):

Das **=-Zeichen**: Hiermit kannst du die senseBox einen Programmierbefehl ausführen lassen, wenn zwei Werte gleich groß sind.

Das **≠-Zeichen**: Hiermit kannst du die senseBox einen Programmierbefehl ausführen lassen, wenn zwei Werte unterschiedlich groß sind.

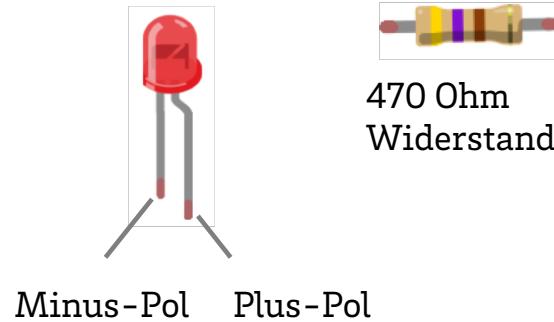
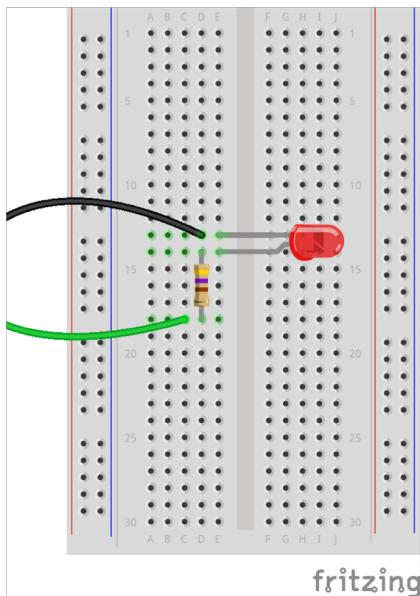
Das **<-Zeichen**: Mithilfe dieses Zeichens kannst du zwei Werte vergleichen lassen. Wenn die Spitze des Symbols auf den kleineren Wert zeigt, führt die senseBox den nächsten Befehl aus.

Das **≤-Zeichen**: Dieses Zeichen ist eine Erweiterung des »kleiner«-Zeichens ( $\leq$ ) und schließt auch Werte ein, die kleiner und gleich groß sind.

Das **>-Zeichen**: Mithilfe dieses Zeichens kannst du den senseBox zwei Werte vergleichen lassen. Die Öffnung des Symbols zeigt diesmal auf den größeren Wert.

Das **≥-Zeichen**: Dieses Zeichen ist eine Erweiterung des »größer«-Zeichens ( $\geq$ ) und schließt auch Werte ein, die größer oder gleich groß sind.

# Basis-Modul : Hello World



Zum Anschließen einer LED benötigst du einen **Widerstand** (470 Ohm) und ein Adapterkabel. Das Adapterkabel steckst du in den **Digital A**-Port auf der senseBox-MCU. Danach verbindest du das **schwarze** und das **grüne** Kabel wie oben abgebildet.

Um die LED zum leuchten zu bringen benötigst du nur diesen einen Block:

```
LED an digital PIN: BUILTIN_1 Status Ein
```

**Aufgabe:** Baue die abgebildete Schaltung und bringe die LED zum blinken.

```
LED an digital PIN: BUILTIN_1 Status Ein
```

```
Warte 1000 Millisekunden
```

# Projekt I: Verkehrszähler

## Verkehrszähler

**Problemstellung:** Es soll ein neuer Fußgängerüberweg gebaut werden. Dafür stehen zwei Straßen zur Auswahl. Es ist jedoch unklar, an welcher der Straßen sich ein Fußgängerüberweg mehr lohnen würde. Um eine Entscheidung zu fällen soll das Verkehrsaufkommen an beiden Straßen gemessen werden.

**Aufgabe:** Baue und Programmiere einen automatischen Verkehrszähler mit Hilfe des Ultraschall-Distanzsensors



### Schritte:

1. Schaue dir zuerst die Karte **SB08 „Der Ultraschall-Distanzsensor“** an. Dort findest du alle wichtigen Informationen, um Distanzen mit dem Ultraschall-Distanzsensor zu messen. Lasse dir die Messwerte des Ultraschalldistanzsensors auf dem Display anzeigen.
2. Überlege dir eine Bedingung, wie du mit Hilfe der Distanzwerte erkennen kannst ob ein Auto vorbeigefahren ist. Schaue dir anschließend die Karten **GI02-03** an.

**Tipp:** Baue dir eine kleine Modellstraße auf dem Tisch und definiere passende Spurbreiten. Weitere Hilfe findest du auf der Hilfekarte **H01**.

3. Lasse dir die Anzahl der gezählten Autos auf dem Display anzeigen, schaue dir dazu die Karte **SB06** an.

## Hilfe

Um zu verhindern, dass Autos doppelt gezählt werden brauchst du eine zweite Bedingung, welche überprüft ob die Spur wieder frei ist. So wird ein vor dem Sensor haltendes Auto nicht mehrfach gezählt.  
Für weitere Hilfen kannst du dir die Hilfekarte **H02** zum Verkehrszähler holen.

# Projekt II: IoT Umweltmessstation

## Umweltmessstation mit Internetanbindung

**Problemstellung:** Umweltmessstationen sind oftmals sehr teuer, weshalb recht wenige und meist nur in Städten von den Behörden aufgestellt werden. In der Zukunft wird es normal sein Bürger in die Erfassung von Umweltdaten einzubinden und die Daten direkt z.B. vom Balkon ins Internet zu übertragen.

**Aufgabe:** Baue eine Umweltmessstation, die ihre Messwerte auf dem Display anzeigen und ins Internet auf die openSenseMap ([www.opensensemap.org](http://www.opensensemap.org)) übertragen kann.

### Schritte:

1. Schaue dir zuerst die Karte **SB05 „Umweltsensoren auslesen“** an. Dort findest du alle wichtigen Informationen um die Umweltsensoren auszulesen.
2. Lasse dir die Messwerte der Umweltsensoren auf dem Display anzeigen. Informationen dazu findest du auf Karte **SB07 „Das Display“**.
3. Die openSenseMap ist eine Web-Applikation für freie Umweltdaten. Registriere eine neue Umweltmessstation mit einer „Manuellen Konfiguration“ und lasse die Messwerte deiner Station über WLAN übertragen. Informationen dazu findest du auf der Karte **SB09 „Übertragen an die openSenseMap“**. Wenn kein offenes WLAN zur Verfügung steht kannst du vielleicht mit deinem Handy einen WLAN-Hotspot erstellen.
4. Suche Deine Station auf der openSenseMap und schaue, ob die Daten live übertragen werden.



# Projekt III: Intelligente Straßenbeleuchtung

## Intelligente Straßenbeleuchtung

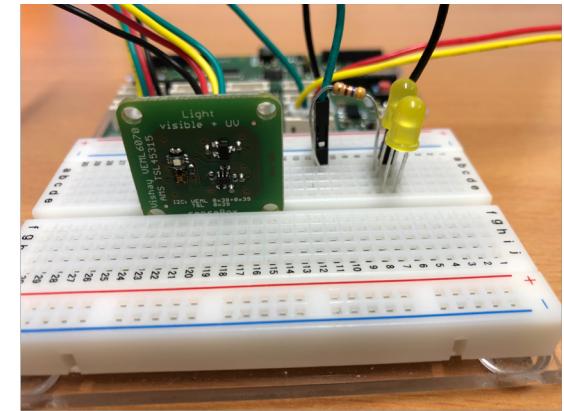
**Problemstellung:** In Deutschland werden jährlich etwa 750 Millionen Euro für die Beleuchtung von Straßen, Plätzen und Brücken ausgegeben. Dabei ist die Beleuchtung gar nicht notwendig, solange niemand die Straße oder den Gehweg benutzt.

**Aufgabe:** Baue und Programmiere eine intelligente Straßenbeleuchtung, die nur dann leuchtet wenn es dunkel ist.

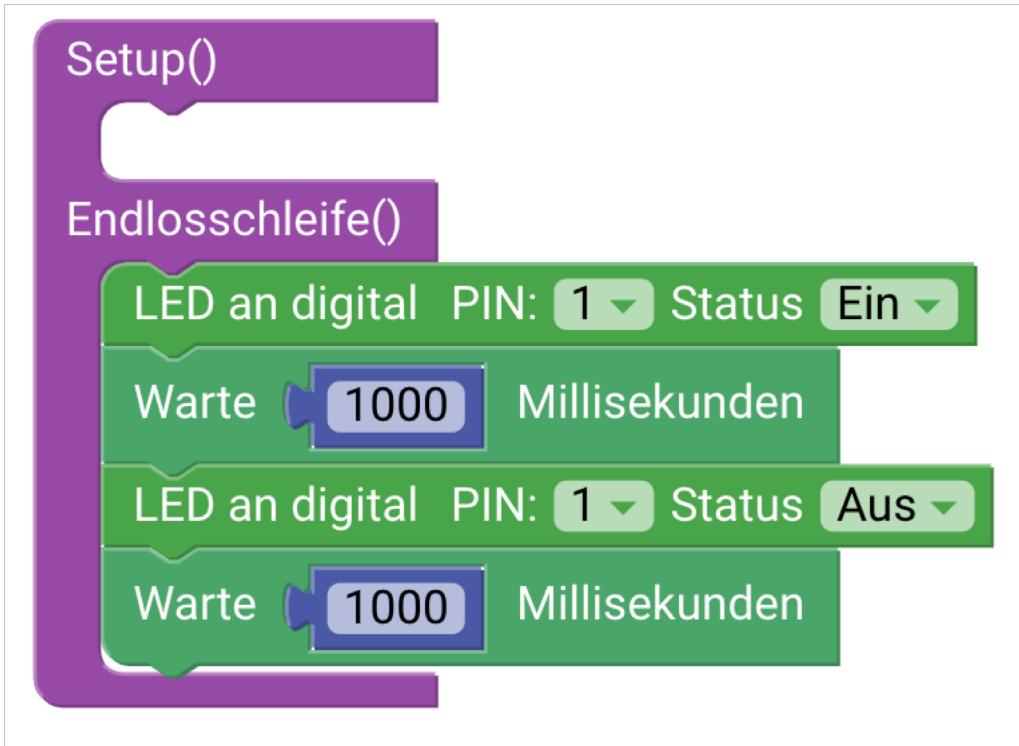
### Schritte:

1. Schaue dir zuerst die Karte **SB05 „Umweltsensoren auslesen“** an. Dort findest du alle wichtigen Informationen um den Helligkeitssensor anzuschließen und zu programmieren.
2. Baue eine Modell-Straßenbeleuchtung aus LEDs auf.
3. Lege einen Helligkeitswert fest, ab dem die Beleuchtung eingeschaltet werden soll. Schaue dir anschließend die Karten **GI02-03** an.

Tipp: Zum Festlegen eines Helligkeitswertes lässt du dir am besten die gemessen Werte auf dem Display anzeigen und überprüfst durch abdecken und beleuchten den Wertebereich. Hilfe zum Display findest du auf der Karte **SB07 „Das Display“**.

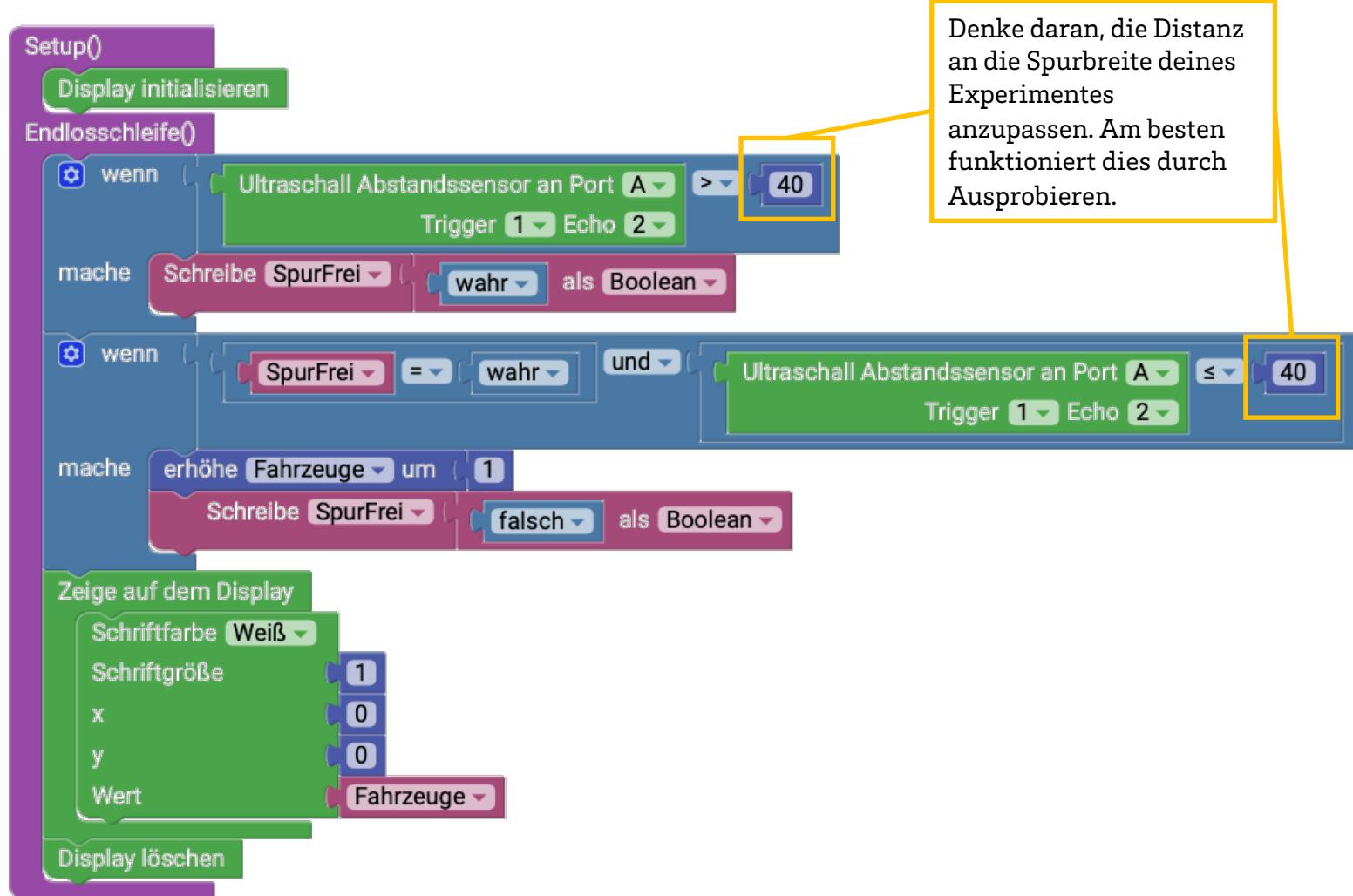


# Lösung: Basis-Modul

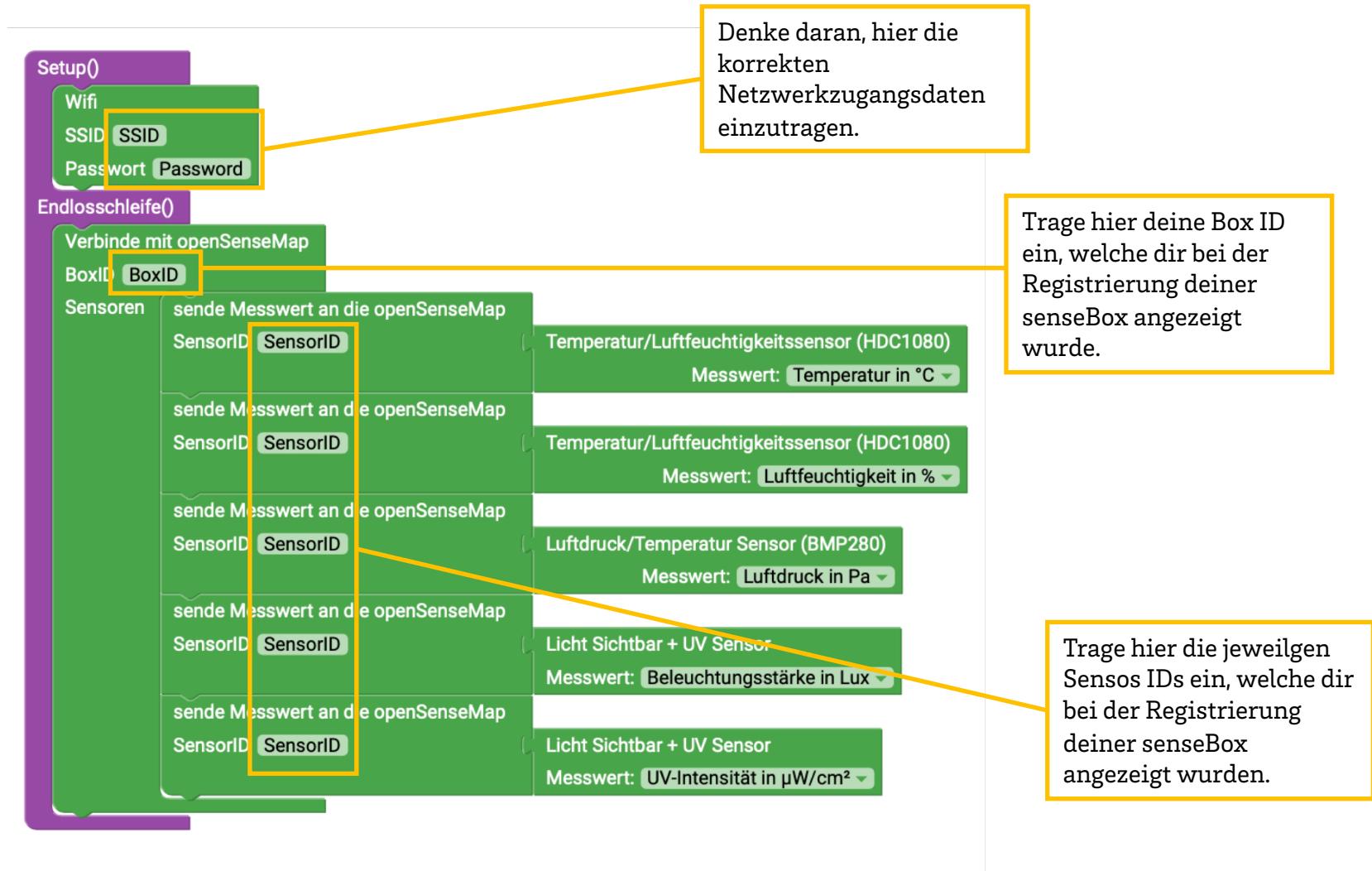


# Lösung: Projekt I Verkehrszähler

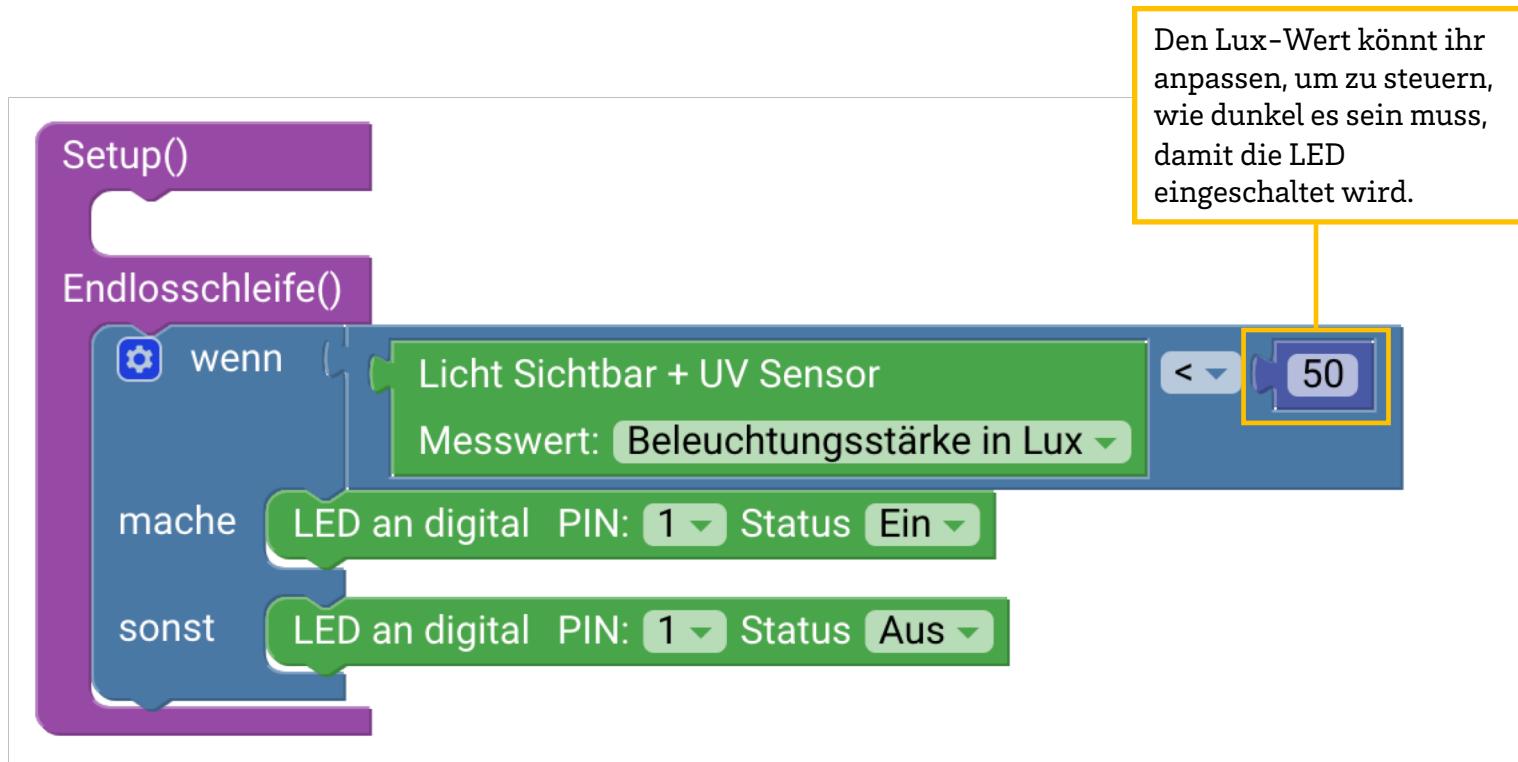
Denkt daran die Spurbreiten auf euer Experiment anzupassen!



# Lösung: Projekt II- IoT Umweltmessstation



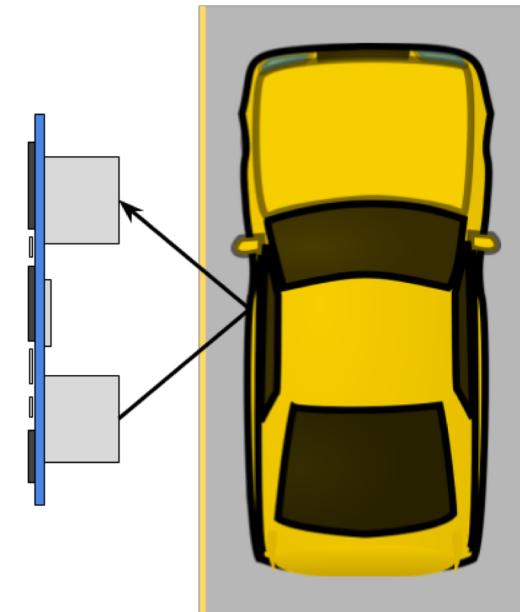
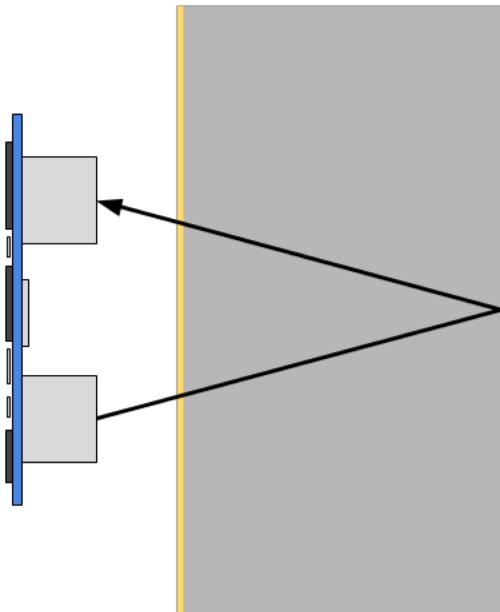
# Lösung: Projekt III – Intelligente Straßenbeleuchtung



# Hilfekarte 1: Autos zählen

Wie kann ich mit einem Distanzsensor Autos zählen?

Die Straße, egal ob im Modell oder in der Realität hat eine bestimmte Breite. Was wird der Distanzsensor messen wenn ein Auto vorbei fährt?

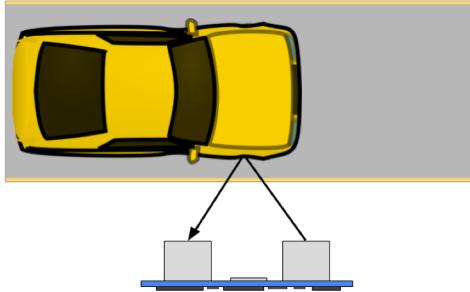


## Hilfekarte 2: Falsche Zählungen

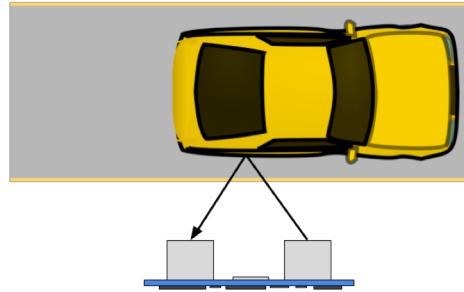
Computer, also auch die senseBox MCU, arbeiten sehr schnell. Für deinen Verkehrszähler bedeutet dies, dass er so oft pro Minute zählt, dass ein langsames Auto es nicht schafft vorbeizufahren, bevor es erneut gemessen wird. Nun wäre vermutlich die erste Idee einfach eine Pause zwischen den Messungen zu programmieren. Aber was ist, wenn ein sehr schnelles Auto vorbeifährt? Genau, das schnelle Auto könnte übersehen werden.

Wie kannst Du also verhindern, dass langsame Autos mehrfach gezählt werden, aber schnelle Autos nicht übersehen werden?

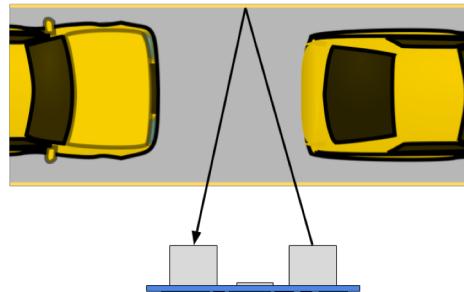
Eine zuverlässige Lösung für dieses Problem ist es, eine zusätzliche Bedingung festzulegen. Diese Bedingung sollte festlegen, dass nachdem ein Auto gezählt wird, zuerst die maximale Spurbreite gemessen werden muss, bevor erneut hochgezählt werden kann.



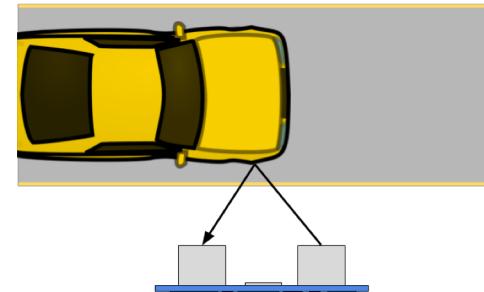
1. Distanz A wird unterschritten, das Auto gezählt und die 2. Bedingung gleichzeitig auf "falsch" gesetzt.



2. Distanz A wird wieder unterschritten, die 2. Bedingung ist aber gleich „falsch“. Es wird nicht gezählt.



3. Es Distanz A wird nicht mehr unterschritten. Die 2. Bedingung wird wieder auf „wahr“ gesetzt.



4. Distanz A wird unterschritten, das Auto gezählt und die 2. Bedingung gleichzeitig auf "falsch" gesetzt.

