

# Inhaltsverzeichnis

Das senseBox:edu Buch	1.1
Übersicht	1.2

## Erste Schritte

Schritt 1: Software Installation	2.1
Schritt 2: Board-Support-Packages installieren	2.2
Schritt 3: Anschluss und Verkabelung	2.3
Schritt 4: Komponenten testen	2.4

## Grundlagen

Arduino IDE und Sketches	3.1
Kommentare im Quelltext	3.1.1
if/else - Bedingung	3.1.2
Schleifen	3.1.3
Variablen	3.1.4
Der Serielle Monitor	3.1.5
Digitale Signale	3.2
Der serielle Datenbus(I <sup>2</sup> C)	3.3
Analoge Signale	3.4
Bees	3.5
Breadboard	3.6
Registrierung der senseBox:edu auf der oSeM	3.7

## Projekte

DIY-Umweltstation	4.1
Temperatur und Luftfeuchtigkeit	4.1.1
Experimente mit Licht	4.1.2
UV-Sensor	4.1.3
Luftdruck	4.1.4
Datenupload zur OSeM	4.1.5
Ampelschaltung	4.2
Verkehrszähler	4.3
Mobile Station	4.4
Lauschangriff	4.5
Auf die Töne fertig los!	4.6

---

Mehrfarbige LED	4.7
Kaminfeuer	4.8
Umweltstation mit Solar	4.9
Thermometer Hütte für die senseBox	4.10

---

## Komponenten

Inhalt	5.1
senseBox MCU	5.1.1
Bees	5.1.2
Wifi-Bee	5.1.2.1
LAN-Bee	5.1.2.2
mSD-Bee	5.1.2.3
LoRa-Bee	5.1.2.4
Sensoren	5.1.3
Temperatur & Luftfeuchte (HDC1080)	5.1.3.1
Luftdruck & Temperatur	5.1.3.2
Belichtung und UV	5.1.3.3
Feinstaub	5.1.3.4
Zubehör	5.1.4
Strahlenschutz	5.1.4.1
Gehäuse	5.1.4.2
Netzteil und USB-Kabel	5.1.4.3
LED-Display	5.1.4.4
Expander	5.1.4.5
Micro-SD Karte	5.1.4.6
GPS	5.1.4.7

---

## Hilfe & Weiteres

FAQ	6.1
Downloads	6.2
Externe Libraries hinzufügen	6.3
Firmware Update Wifi-Bee	6.4
Windows USB-Bootloader Treiber aktualisieren	6.5
Aktualisierung des Board-Support-Package & senseBox-Libraries	6.6
Contributing	6.7
Hint-Box	6.7.1
Bildunterschriften	6.7.2
Tabs	6.7.3
Neue Seite	6.7.4

---

---

[Collapsible](#)

6.7.5

# Das senseBox:edu Buch

Hier findest Du alle Informationen, die du brauchst um mit Deiner senseBox:edu durchzustarten!

## Über dieses Buch

Dieses Buch begleitet dich auf deinem Weg zur eigenen Umweltmessstation, vom Aufbau bis zur Programmierung. Dabei kannst du spannende Experimente durchführen und gleichzeitig eine Menge lernen über Sensoren, das Programmieren, Elektrotechnik und vieles mehr.

Damit du ohne weiteres starten kannst, lies dir einmal kurz aber gründlich die Einleitung durch. Sie hilft dir das Buch besser zu verstehen und gibt dir Einsicht in das, was dich erwartet.

## Einleitung

Das senseBox:edu Buch ist ein Buch für alle Tüftler und Interessierte, die im Besitz einer senseBox:edu sind. Also für alle, die nicht nur einfach eine Umweltmessstation zusammenbauen und aufstellen wollen, sondern die Technik, Hardware und Software dahinter verstehen und beherrschen möchten. Mit der senseBox:edu sind dir keine Limits gesetzt, eigene Anwendungsfälle zu erfinden und dazu passende Programme zu schreiben. Die senseBox:edu wird zusammen mit einem umfassenden Setup an Sensoren, LEDs, Widerständen, Kabeln und weiteren spannenden Komponenten geliefert. Sie kann in der Schule zum Lernen und Verstehen von Informatik, Physik und Technik eingesetzt werden, aber auch zuhause zum Experimentieren benutzt werden.

Dieses Buch hilft dir bei den [Ersten Schritten](#)<sup>1</sup> mit deiner senseBox:edu, erklärt dir Grundlagen, stellt dir Projekte zum selber nachbauen vor und erklärt dir die einzelnen Komponenten der senseBox. Du solltest mit den ersten Schritten starten und diese lesen und befolgen, bevor du dich an die vorgestellten DIY-Projekte oder eigene Projekte wagst.

Auf dem Weg durch das Buch wirst Du dich mithilfe der Seitenleiste links oder mit den grünen Pfeiltasten auf den Seiten durch die Anleitungen klicken. Jede Seite hat eine Überschrift und darunter eine grau hinterlegte Box mit grüner Schrift, in der eine kleine Beschreibung der Seite verfasst wurde. Das Ganze sieht in etwa so aus:

## Überschrift

Hier steht eine kleine Beschreibung von dem, was dich auf dieser Seite erwartet.

---

Außerdem werden dir mit Sicherheit einige unserer vier verschiedenen Arten von Hinweis-Boxen begegnen. Hier siehst du, wofür die Boxen stehen und was sie dir vermitteln:

Diese Box ziegt dir Info-Hinweise. Zum Beispiel Links zu weiteren Informationsquellen oder weiteres

## Hintergrundwissen.

Diese Hinweis-Box ist eine Art Checkliste. In ihr stehen zusammenfassend Sachen, die du im Laufe der Seite erledigt hast. An ihr kannst du abschließend abhaken, ob du keinen wichtigen Zwischenschritt vergessen hast.

Dies ist ein Warn-Hinweis! Er weist dich auf potentielle Fehlerquellen hin oder zeigt dir Stellen, an denen du besonders vorsichtig arbeiten musst oder besonders genau hinschauen solltest!

Diese Box weist dich auf Fehler hin. Sie zeigt dir z.B. was du machen kannst, um einen bereits geschehenen Fehler zu korrigieren.

Weiterhin gibt es einige Absätze, die dir nicht direkt angezeigt werden. Sie enthalten Informationen über weitergehende Themen oder Aufgaben und Beispiele. Damit du eine größere Übersicht erhältst, sind sie eingeklappt. Möchtest du den Inhalt sehen, klicke einfach auf die Überschrift des Absatzes. Du erkennst eingeklappte Absätze an der grauen Box mit grüner Schrift, ähnlich wie bei der Box unter der Überschrift der Seite. Zusätzlich befindet sich ein kleiner grüner Pfeil, der dir vermittelt, ob der Absatz ein- oder ausgeklappt ist. Schau dir hier doch einfach mal so einen versteckten Absatz an:

### ▼ Ein versteckter Absatz

Super, du hast alles richtig verstanden! Klicke noch einmal auf die grüne Überschrift des Absatzes und ich verschwinde wieder!

Als letztes wichtiges Element, dass du kennen solltest bevor Du mit diesem Buch durchstarten kannst, haben wir sogenannte "Tabs". Du kennst sie vermutlich aus deinem Internet-Browser und wenn Du dich aktuell im Internet befindest, hast du auch bestimmt mehrere Tabs geöffnet. Wir haben ähnlich wie im Browser auch Tabs, diese sind allerdings innerhalb der Seite eingebaut. Sie helfen uns, das was du siehst, auf das was du auch wirklich benötigst, zu reduzieren. Wenn wir zum Beispiel eine Anleitung beschreiben, um ein Programm zu installieren, tun wir dies für verschiedene Betriebssysteme (Windows, OSX(Mac) und Linux). Da Du im Normalfall nur an einem Computer arbeitest, ist für dich nur das darauf installierte Betriebssystem von Interesse. Daher integrieren wir die Installationsanleitungen in drei Tabs. In jedem dieser drei Tabs wird die passende Anleitung für ein Betriebssystem angezeigt. Du wählst das zu deinem System passende Tab aus und bekommst so nur die passende Anleitung angezeigt und nicht die anderen beiden. Probier es einfach mal aus und schau dir an, was passiert wenn du hier die verschiedenen Tabs anklickst.

- [Erster Tab](#)
- [Zweiter Tab](#)
- [Dritter Tab](#)

## Erster Tab

Ich bin der erste Tab. Ich bin standardmäßig aufgerufen, wenn du diese Seite öffnest.

## Zweiter Tab

Ich bin der zweite Tab. Indem du mich angewählt, hast ersetzt mein Inhalt den Inhalt des ersten Tabs.

## Dritter Tab

Ich bin der dritte Tab. Indem du mich angewählt, hast wird der Inhalt des vorher ausgewählten Tabs durch meinen Inhalt ersetzt. Viel Spaß noch mit dem senseBox:edu Buch!

---

Hast Du dir die Tabs und alle weiteren Elemente angeschaut? Wenn ja, dann kannst Du jetzt loslegen mit diesem Buch und deiner senseBox:edu!

Viel Spaß wünscht dir das Team der [senseBox](#)<sup>1</sup>!

---

<sup>1</sup>. Siehe [2.1 Schritt 1: Software Installation](#) ↵

<sup>1</sup>. <https://sensebox.de/> ↵

# Übersicht

In diesem Buch zeigen wir Dir, was du mit deiner senseBox machen kannst, und erklären dir alles, was du wissen musst, um eine fertige Wetterstation zu bauen. Außerdem geben wir dir Tipps und Anregungen zu weiteren Experimenten.

Damit alles reibungslos läuft, starten wir ganz von vorne und führen dich Schritt für Schritt durch die Installation von Software und Hardware bis zum Bau deiner einsatzbereiten Messstation. Dazu bekommst Du zuerst einmal eine Übersicht wie dieses Buch aufgebaut ist.

1. Erste Schritte - Installiere benötigte Programme, teste deine Sensoren und starte durch mit Deiner senseBox
  - i. [Installation von benötigter Software](#)
  - ii. [Installieren der Board-Support-Packages und senseBox Libraries](#)
  - iii. [Anschließen und Verkabeln von MCU und Sensoren](#)
  - iv. [Programmierung und erste Tests mit deinen Sensoren](#)
2. Grundlagen
3. Projekte
4. Komponenten - Schaue dir alle Teile der senseBox und ihre Funktionen an]
  - i. [Inhalt](#)
    - i. [sensebox MCU](#)
    - ii. [Bees](#)
    - iii. [Sensoren](#)
    - iv. [Weiteres Zubehör](#)
5. Hilfe - Häufig gestellte Fragen, Antworten und Hilfe bei Problemen
  - i. [FAQ](#)
  - ii. [Externe Libraries hinzufügen](#)
  - iii. [Firmware Update Wifi-Bee](#)
  - iv. [Windows USB-Bootloader Treiber aktualisieren](#)
  - v. [Aktualisierung des Board-Support-Package & senseBox-Libraries](#)
  - vi. [Contributing](#)

## Schritt 1: Software Installation

Bevor du mit dem Messen von Phänomenen starten kannst, musst du noch ein paar Dinge beachten. Dazu gehört die Installation von verschiedenen Treibern und Software. Keine Sorge, es handelt sich um Open-Source-Software, du zahlst also nichts dafür. Du solltest dir die Schritte 1 bis 4 allerdings genau anschauen, damit später keine Probleme auftreten.

### Arduino IDE installieren

Bevor die senseBox aktiviert werden kann, musst du Treiber sowie eine Software auf deinem Computer installieren. Außerdem ist es vor Inbetriebnahme der senseBox ratsam einen Testlauf durchzuführen, um zu überprüfen ob die Sensoren korrekt funktionieren und die Kommunikation mit dem Internet reibungslos läuft.

Schau dir die Anleitung für dein Betriebssystem an und folge den angegebenen Schritten.

- [Windows](#)
- [Mac\(OSX\)](#)
- [Linux](#)

#### Arduino Software für Windows herunterladen

Für einen reibungslosen Ablauf bitte Arduino 1.8.5 oder höher nutzen.

Die senseBox ist ein Microcontroller mit verschiedenen Komponenten und Sensoren. Sie wird über die Entwicklungsumgebung Arduino IDE programmiert. Lade die neueste Version als zip-Datei von der [Arduino Homepage](#)<sup>321</sup> herunter:



Arduino ist ein Open-Source Projekt und wird durch Spenden finanziert. Daher wirst du vor dem Download nach einer Spende gefragt; das kannst du überspringen, indem du auf [JUST DOWNLOAD](#) klickst.



Lege auf deiner Festplatte einen neuen Ordner an und entpacke darin die zip-Datei. Durch das Starten der Datei `arduino.exe` kann die IDE gestartet werden.

#### Arduino Software für Mac(OSX) herunterladen

Für einen reibungslosen Ablauf bitte Arduino 1.8.5 oder höher nutzen.

Die senseBox ist ein Microcontroller mit verschiedenen Komponenten und Sensoren. Sie wird über die Entwicklungsumgebung Arduino IDE programmiert. Lade die neueste Version von der [Arduino Homepage](#) herunter:



Arduino ist ein Open-Source Projekt und wird durch Spenden finanziert. Daher wirst du vor dem Download nach einer Spende gefragt; das kannst du überspringen, indem du auf `JUST DOWNLOAD` klickst.



In deinem Downloads-Ordner sollte eine `Arduino.app` Datei erscheinen. Verschiebe diese Datei in deinen "Programme"-Ordner. Durch starten der Datei `Arduino.app` kann die IDE gestartet werden.

## Arduino Software für Linux herunterladen

Für einen reibungslosen Ablauf bitte Arduino 1.8.5 oder höher nutzen.

Die senseBox ist ein Microcontroller mit verschiedenen Komponenten und Sensoren. Sie wird über die Entwicklungsumgebung Arduino IDE programmiert. Lade die neueste Version von der [Arduino Homepage](#) herunter:



Arduino ist ein Open-Source Projekt und wird durch Spenden finanziert. Daher wirst du vor dem Download nach einer Spende gefragt; das kannst du überspringen, indem du auf `JUST DOWNLOAD` klickst.



## Installation der IDE unter Linux

Linux-Nutzer können die Linuxvariante herunterladen und entpacken. Das enthaltene `install.sh` -Skript legt automatisch eine Desktopverknüpfung an. Am schnellsten geht dies über den Terminal. Öffne dazu den Terminal in dem du die Tasten `Ctrl + Alt + T` drückst und gib dort die folgenden Befehle ein:

```
# sollte die heruntergeladene Datei nicht im Downloads-Ordner abgespeichert sein, ersetze "Downloads" durch den Pfad zum entsprechenden Ordner
cd Downloads
```

```
# entpacke die Datei mit folgendem Befehl und installiere Arduino
```

```
tar -xvf arduino-1.8.5-linux64.tar.xz  
cd arduino-1.8.5  
.install.sh
```

Achte darauf, dass der Befehl auf die heruntergeladene Arduinoverision abgestimmt ist! Lädst du z.B. Arduino 1.8.6 muss auch überall wo arduino-1.8.5 steht arduino-1.8.6 stehen. Um zu überprüfen welche Version du heruntergeladen hast, schau dir den Namen der Download-Datei an.

Um den Arduino programmieren zu können, sind unter Ubuntu 14 & 16 zusätzliche Rechte notwendig. Diese können für den aktuellen Nutzer mit den folgenden Befehlen eingerichtet werden (benötigt Admin-Rechte):

Führe `udevadm monitor --udev` aus und schließe den Arduino per USB an, um die Device-ID zu bestimmen. Der angegebene Bezeichnung am Ende der Ausgabe (zB. `ttyUSB0`) ist die Device-ID. Beende `udevadm` per `ctrl+c`, und führe noch die folgenden Befehle aus, wobei die herausgefundene Device-ID eingesetzt werden muss:

```
sudo usermod -a -G dialout $(whoami)  
sudo chmod a+r /dev/<device-id>
```

Nach einem Logout und erneutem Login sollte der Arduino aus der Arduino IDE programmierbar sein!

- 
- 1. <https://www.arduino.cc/en/main/software> ↵
  - 2. <https://www.arduino.cc/en/main/software> ↵
  - 3. <https://www.arduino.cc/en/main/software> ↵

## Schritt 2: Board-Support-Package installieren

Damit die Arduino IDE deine senseBox MCU unterstützt und Du Programme auf diese übertragen kannst, musst Du vor Beginn noch zwei Board-Support-Packages installieren. Diese beinhalten die nötigen Treiber und die notwendige Software, um mit deinem Prozessor zu kommunizieren. Das Board-Support-Package der senseBox enthält außerdem bereits unsere senseBox-Libraries. Damit stehen euch alle grundlegenden Methoden, zur Programmierung der beiliegenden Sensoren, zu Verfügung.

### Libraries

Für die Programmierung der senseBox sollten zu Beginn unbedingt die senseBox-Libraries eingebunden werden. Diese Libraries haben wir für euch in das Board-Support-Package der senseBox integriert, um euch die Installation möglichst einfach zu gestalten.

#### ▼ 'Library' – Was ist das eigentlich und wofür brauche ich das?

Eine Library ist wie der Name schon sagt eine Sammlung von etwas – eine Sammlung von Methoden um genauer zu sein. Methoden sind in der Programmierung kleinere Abschnitte von Code, die auf ein Objekt angewendet werden können. Bei der senseBox zum Beispiel kann eine Methode aufgerufen werden, um die LEDs auf dem MCU ein- und auszuschalten. Es gibt eine Menge solcher Standardmethoden, die von einer Vielzahl an Programmen benutzt werden. Um diese Methoden nicht alle einzeln in den Programmcode übertragen zu müssen, können sie in Libraries abgelegt werden. Eine Library ist also eine Datei, in der viele Methoden gespeichert werden. Man kann Libraries in seinen Code einbinden. Dafür reicht es wenn sie im Arduino-Ordner für Libraries gespeichert sind und man sie dann mit einer einzigen Zeile zu Beginn des Programmcodes einbindet. Das sieht in Arduino für die Library mit dem Namen "senseBoxIO" wie folgt aus:

```
#include <senseBoxMCU.h>;
```

Ist die Library eingebunden, können alle in ihr enthaltenen Methoden im Code benutzt werden.

### Board-Support-Package einbinden

Wähle dein Betriebssystem, um die passende Anleitung zu sehen:

- Windows
- Mac(OSX)
- Linux

### Anleitung für Windows

Füge die folgende URL in deiner Arduino IDE unter Datei -> Voreinstellungen in das Feld für Zusätzliche Bordverwalter-URLs ein:

```
https://github.com/sensebox/senseBoxMCU-core/raw/master/package_sensebox_index.json
```



*Öffne die Voreinstellungen und füge die URL ein*

Öffne dann den Boardverwalter unter Werkzeuge -> Board:"..." -> Boardverwalter und installiere dort die zwei Board-Support-Packages mit den Namen Arduino SAMD Boards by Arduino und senseBox SAMD Boards by senseBox.



*Öffne den Boardverwalter und installiere die beiden Packages*

Gib "SAMD" oben in die Suchleiste ein um die Packages schneller zu finden

Da wir das senseBox SAMD Boards-Package für euch regelmäßig aktualisieren, solltet ihr immer mal wieder in den Boardverwalter gehen und nachschauen, ob das senseBox SAMD Boards-Package noch aktuell ist. Öffnet dafür, wie oben beschrieben, den Boardverwalter und sucht nach senseBox SAMD Boards. Wenn ihr auf den Eintrag in der Liste klickt, erscheint dort, im Falle einer neuen Version, ein Update-Button. Klickt diesen, um die neuste Version zu installieren.



*Klicke auf 'Update', um das Board-Support-Package zu aktualisieren*

Wichtig ist, zuerst auf den Eintrag zu klicken. Ansonsten wird der Update-Button nicht angezeigt, auch wenn es bereits eine neue Version gibt.

## Anleitung für Mac

Füge die folgende URL in deiner Arduino IDE unter *Arduino -> Einstellungen...* in das Feld für Zusätzliche Bordverwalter-URLs ein:

```
https://github.com/sensebox/senseBoxMCU-core/raw/master/package_sensebox_index.json
```



*Öffne die Voreinstellungen und füge die URL ein*

Öffne dann den Boardverwalter unter Werkzeuge -> Board..." -> Boardverwalter und installiere dort die zwei Board-Support-Packages mit den Namen Arduino SAMD Boards by Arduino und senseBox SAMD Boards by senseBox.



*Öffne den Bordverwalter*



*Installiere die beiden markierten Packages*

Gib "SAMD" oben in die Suchleiste ein um die Packages schneller zu finden

Da wir das senseBox SAMD Boards-Package für euch regelmäßig aktualisieren, solltet ihr immer mal wieder in den Boardverwalter gehen und nachschauen, ob das senseBox SAMD Boards-Package noch aktuell ist. Öffnet dafür, wie oben beschrieben, den Boardverwalter und sucht nach senseBox SAMD Boards. Wenn ihr auf den Eintrag in der Liste klickt, erscheint dort, im Falle einer neuen Version, ein Update-Button. Klickt diesen, um die neuste Version zu installieren.



*Klicke auf 'Update', um das Board-Support-Package zu aktualisieren*

Wichtig ist, zuerst auf den Eintrag zu klicken. Ansonsten wird der Update-Button nicht angezeigt, auch wenn es bereits eine neue Version gibt.

## Anleitung für Linux

Füge die folgende URL in deiner Arduino IDE unter Datei -> Voreinstellungen in das Feld für Zusätzliche Bordverwalter-URLs ein:

```
https://github.com/sensebox/senseBoxMCU-core/raw/master/package_sensebox_index.json
```



*Öffne die Voreinstellungen und füge die URL ein*

Öffne dann den Boardverwalter unter Werkzeuge -> Board:"..." -> Boardverwalter und installiere dort die zwei Board-Support-Packages mit den Namen Arduino SAMD Boards by Arduino und senseBox SAMD Boards by senseBox.



*Öffne den Bordverwalter und installiere die beiden Packages*

Gib "SAMD" oben in die Suchleiste ein um die Packages schneller zu finden

Da wir das senseBox SAMD Boards-Package für euch regelmäßig aktualisieren, solltet ihr immer mal wieder in den Boardverwalter gehen und nachschauen, ob das senseBox SAMD Boards-Package noch aktuell ist. Öffnet dafür, wie oben beschrieben, den Boardverwalter und sucht nach senseBox SAMD Boards. Wenn ihr auf den Eintrag in der Liste klickt, erscheint dort, im Falle einer neuen Version, ein Update-Button. Klickt diesen, um die neuste Version zu installieren.



*Klicke auf 'Update', um das Board-Support-Package zu aktualisieren*

Wichtig ist, zuerst auf den Eintrag zu klicken. Ansonsten wird der Update-Button nicht angezeigt, auch wenn es bereits eine neue Version gibt.

## Schritt 3: Anschluss und Verkabelung

Durch das I2C Stecksystem ist der Anschluss der Sensoren und Komponenten an die senseBox MCU sehr einfach.  
Hier siehst du noch einmal eine grobe Übersicht über die einzelnen Komponenten

### Anschluss von Bees

Der Anschluss der Bees ist ganz einfach. Durch das Stecksystem reicht es das Bee direkt auf den Microcontroller aufzusetzen. Dabei gibt es nur zwei Dinge zu beachten: 1. Die Ausrichtung auf dem Board und 2. Der richtige Portanschluss auf dem Microcontroller.

#### WiFi-Bee, Ethernet-Bee und LoRa-Bee

Diese Bees werden alle auf den Port 1 aufgesteckt. Den richtigen Port erkennst du an der Beschriftung: `XBEE1`. Die richtige Richtung beim Aufstecken erkennst du an der 7-eckigen Kennzeichnung auf dem Board und der Bee.



*Exemplarischer Anschluss der WiFi-Bee an die MCU (XBEE1)*

#### mSD-Bee

Die SD-Bee wird an Port 2 aufgesteckt, welcher standardmäßig dafür freigeschalten ist. Den richtigen Port erkennst du an der Beschriftung: `XBEE2`. Die richtige Richtung beim Aufstecken erkennst du an der 7-eckigen Kennzeichnung auf dem Board und der Bee.



*Anschluss der mSD-Bee an die MCU (XBEE1)*

### Anschluss von einfachen Sensoren

Die Sensoren, welche mit der senseBox gekauft werden können, sind mit den beiliegenden "I2C to I2C" Kabeln sehr einfach anzuschließen. Dafür musst du die Steckplätze auf dem Board verwenden die mit `I2C/Wire` gekennzeichnet sind.



*Anschluss einfacher Sensoren)*

## Anschluss vom Feinstaubsensor

Der Feinstaubsensor, der mit der senseBox gekauft werden kann, hat ein beiliegendes passendes Kabel, welches den Sensor mit dem Board verbinden kann. Hierfür musst du die Steckplätze auf dem Board verwenden die mit `UART/Serial` gekennzeichnet sind. Weitere Infos zum Anschluss des Feinstaubsensors findest du übrigens [hier<sup>1</sup>](#).



*Anschluss Feinstaubsensor*

---

<sup>1</sup>. Siehe [5.1.3.4 Feinstaub ↗](#)

## Schritt 4: Programmierung und Komponententest

In diesem Kapitel wird beschrieben wie die Programmierung der senseBox funktioniert und wie ihr die mitgelieferten Sensoren und Komponenten testen könnt.

Ein Softwareprogramm für die senseBox wird im Folgenden auch Sketch genannt.

### Programmieren mit der Arduino IDE

Mit der Arduino IDE lässt sich ein Sketch kompilieren und auf die senseBox MCU hochladen. Schließt nun die senseBox per USB Kabel an euren Computer an und befolgt die nächsten Schritte.

#### Konfiguration in der Arduino IDE

Bevor die senseBox programmieren könnt, müsst ihr ein paar Einstellungen in der Arduino IDE vornehmen. Unter dem Reiter `Werkzeuge` müsst ihr bei der Option `Boards` die senseBox MCU unten in der Liste auswählen.



*Boardauswahl*

Als nächstes wählt ihr unter `Werkzeuge -> Port` die Anschlussnummer des USB Ports aus, an dem die senseBox MCU mit dem Computer verbunden ist.



*Portauswahl*

Der Port lässt sich nur dann auswählen, wenn die senseBox mit dem USB Kabel an den Computer angeschlossen wurde.

#### Hello World Beispiel

Kopiert das Beispiel unterhalb in eure Arduino Umgebung und klickt auf das Pfeilsymbol in der Werkzeugeiste. Im unteren Teil der Arduino Oberfläche bekommt ihr Feedback zum Uploadvorgang. Wenn alles geklappt hat, erscheint dort die Meldung Hochladen abgeschlossen .

```
int ledPin = LED_BUILTIN;

void setup()
{
    pinMode(ledPin, OUTPUT);
}

void loop()
{
    digitalWrite(ledPin, HIGH); // turn the LED on (HIGH is the voltage level)
    delay(1000);             // wait for a second
    digitalWrite(ledPin, LOW); // turn the LED off by making the voltage LOW
    delay(1000);             // wait for a second
}
```

Bei dem Text der hinter dem `//` steht handelt es sich um einen Kommentar der nicht vom Kompiler mit ausgewertet wird. Das hat den Sinn, dass man sich besser im Code zurecht findet und man macht es so anderen Programmierern leichter den eigenen Code zu verstehen.

Anders als bei einem Laptop oder Smartphone läuft auf eurer senseBox kein Betriebssystem wie Windows, Linux oder MacOS. Die senseBox MCU ist ein Mikrocontroller auf dem immer nur das letzte Programm ausgeführt wird, das hochgeladen wurde.

## Sensoren und Internetverbindung testen

Bevor ihr eure senseBox mit der openSenseMap verbindet sollten alle Sensoren und das Netzwerkmodul überprüft werden, um späteren Fehlern vorzubeugen. Mit unserem Testprogramm können nach dem Aufbau der Station der Messvorgang und die Netzwerkverbindung getestet werden.

Voraussetzung dafür ist die aktuellste Version des Board-Support-Package aus [Schritt 2](#). Am Ende von Schritt 2 ist erklärt, wie ihr das Board-Support-Package auf den neusten Stand bringen könnt.

### Test-Sketch öffnen

Öffnet aus den Beispielen die Datei `mcu_component_test` ( Datei -> Beispiele -> senseBoxMCU ). Nachdem ihr diesen Sketch auf die MCU hochgeladen habt, startet ihr den seriellen Monitor indem ihr auf das Lupen-Symbol rechts oben in der Werkzeugeiste klickt.

Für den Fall dass sich der Monitor nicht öffnet solltet ihr überprüfen ob sich das Board im Programmodus befindet (einmal auf Reset drücken) und ob der richtige Port ausgewählt wurde. Danach versucht es erneut mit einem Klick auf die Lupe.

## Optionsmenü

Nachdem ihr den seriellen Monitor geöffnet habt, erscheint ein Menü auf dessen Funktionen ihr über das Eingabefeld zugreifen könnt:



*Optionsmenu*

Dazu schreibt ihr die Nummer der entsprechenden Option in das Eingabefeld und klickt auf „Senden“. Unterhalb findet ihr eine Auflistung der Optionen mit kurzen Beschreibungen.

### 1. Find connected sensors

Hier könnt ihr überprüfen, ob alle angeschlossenen Sensoren richtig initialisiert und erkannt wurden. Für jeden angeschlossenen Sensor sollte es eine Rückmeldung und eine Testmessung geben. Im Beispiel unterhalb wurde ein HDC1080 Temperatur- und Luftfeuchtesensor an einen `I2C/Wire` Port angeschlossen.



Falls bei der Ausgabe einer der angeschlossenen Sensoren fehlt, solltet ihr die Kabelverbindung überprüfen und den Test wiederholen.

### 2. Test connection to openSenseMap

Mit dieser Option wird die Internetverbindung getestet. Bei einem erfolgreichen Verbindungsaufbau sollte eine Antwort mit HTTP-Status 200 vom Server ausgegeben werden:



Falls ihr ein WiFi-Modul nutzt, wird außerdem überprüft ob die aktuellste Version der Firmware auf dem Modul installiert ist. Falls die Version veraltet ist, solltet ihr sie aktualisieren.

### 3. Get security key

Jedes senseBox Board besitzt einen eigenen, einzigartigen Sicherheitsschlüssel den ihr mit dieser Option auslesen könnt. Er wird genutzt um die Verbindung zwischen der openSenseMap und eurer senseBox zu verschlüsseln damit niemand eure Messungen von außen manipulieren kann.

Ihr benötigt diesen Schlüssel im nächsten Schritt bei der Registrierung eurer senseBox auf der openSenseMap.



# Arduino IDE und Sketches

Bevor du loslegen kannst dein erstes eigenes Programm zu schreiben, musst du dir das Arduino Programm - die sogenannte Entwicklungsumgebung (engl. IDE) - anschauen.

## Grundlagen

Wenn du die IDE öffnest, siehst du direkt einen großen weißen Bereich, in welchem du dein Programm schreiben wirst. Im schwarzen Bereich darunter werden dir Status- und Fehlermeldungen angezeigt. Es lohnt sich also immer einen Blick auf die Meldungen zu werfen.

Zuletzt solltest du dir noch die kleinen Schaltflächen über dem weißen Bereich anschauen.



auf Schreibfehler überprüfen - Programm an senseBox MCU übertragen

Das Häkchen und der Pfeil sind die beiden wichtigsten Symbole für dich: Mit dem Häkchen kannst du dein Programm auf Schreibfehler überprüfen lassen, und mit den Pfeil überträgst du dein Programm an die senseBox MCU.



neues Programm erstellen - gespeichertes Programm öffnen - Programm speichern

Die anderen drei Symbole - angefangen mit dem kleinen Blatt ganz links - stehen dafür ein neues Programm zu erstellen, ein gespeichertes zu öffnen und dein geschriebenes Programm zu speichern.

## Der Arduino-Sketch

Ein Arduino Programm (auch "Sketch" genannt) hat einen sehr einfachen Aufbau, der aus zwei Hauptbestandteilen besteht. Diese zwei benötigten Funktionen enthalten Blöcke von Anweisungen, welche den Programmablauf beschreiben:

```
void setup(){
    // Anweisung
}
void loop(){
    // Anweisung
}
```

Die `setup`-Funktion wird nur einmal beim Start des Programmes ausgeführt. In der `loop`-Funktion werden hingegen alle Anweisungen in einer endlosen Schleife wiederholt. Beide Funktionen sind zwingend notwendig, um das Programm erfolgreich kompilieren und ausführen zu können. "Kompilieren" bezeichnet die Übersetzung des Programms in einen Maschinencode, welcher vom Arduino-Prozessor verstanden werden kann; dies übernimmt die Arduino-IDE für uns.

Mit einem doppelten Schrägstrich (`//`) lassen sich Kommentare zum Programmcode hinzufügen. Es ist immer wichtig seinen Programmcode zu kommentieren, damit auch andere nachvollziehen können, was an einer bestimmten Stelle passiert.

## Kommentare im Quelltext

Das Kommentieren von Quellcode ist leider ein Thema das stiefmütterlich behandelt wird. Den Nutzen von guten Kommentaren erkennt man häufig erst dann, wenn man versucht sich durch fremden Quelltext durchzuwuseln oder wenn man ein eigenes Programm nach langer Zeit wieder 'ausgräbt'. Kommentare werden vom Kompiler nicht ausgewertet und beeinflussen den Ablauf des Programms damit nicht. Text und Programmteile die auskommentiert sind, erkennt man daran, dass sie grau gefärbt sind.

### Einzeilig

Einzeilige Kommentare finden sich oft im Quelltext. Sie dienen dazu bestimmte Befehle oder Konstrukte zu erklären. Ein einzeiliger Kommentar wird durch zwei `//` gekennzeichnet.

```
// Ich bin ein Kommentar
int led = 1; // Die variable led bekommt den Wert 1
```

### Mehrzeilige Kommentare

Mehrzeilige Kommentare stehen oft zu Beginn eines Programms oder vor einer Methode. Sie beginnen mit `/*` und enden mit `*/`. Man kann sie außerdem dazu verwenden, Teile eines Programms auszukommentieren. Etwa wenn man einen Fehler hat und überprüfen möchte, in welchem Programmteil er liegt.

```
/*
 *
 * Ich bin ein mehrzeiliger Kommentar
 * Ich kann zum Beispiel beschreiben, welchen Zweck ein ganzes Programm
 * oder eine einzelne Methode hat.
 *
 * Übrigens:
 * <- diese Sterne werden zwar automatisch erzeugt, sind aber nicht unbedingt notwendig.
 *
 */
```

### Wie viele Kommentare braucht ein Quellcode?

Das ist eine Frage, auf die es keine klare Antwort gibt. Es gibt Programmierer, die erwarten, dass jede Zeile Quelltext kommentiert wird. Dies ist bei unseren einfachen Programmen nicht notwendig. Grundsätzlich sollten mindestens folgende Programmteile kommentiert sein:

- Ein Kommentar zu Beginn, welcher den Zweck des Programms beschreibt.
- Jede Methode muss kommentiert sein und hier insbesondere die Eingabeparameter und eventuelle Rückgaben.
- Mathematisch oder logisch anspruchsvolle Befehle oder besondere 'Kniffe' die sich der Programmierer überlegt hat.

Wenn du dich weiter mit dem Thema beschäftigen möchtest, dann schau doch mal bei [Wikipedia](#)

## if/else-Bedingung

Mit if und else ist es möglich in einem Programm Entscheidungen zu fällen und die senseBox MCU, je nachdem wie die Entscheidung ausfällt, unterschiedlichen Code ausführen zu lassen.

### Verwendung von if

Um die Bedeutung von if verständlich zu machen, schauen wir uns den Programmcode zu folgendem Beispiel an:

Wenn du eine LED in Abhängigkeit von einem Schalter leuchten lassen möchtest, würde der Code wie folgt aussehen:

```
if (digitalRead(BUTTON_PIN) == HIGH) {  
    digitalWrite(LED_PIN, HIGH);  
}
```

Die erste Codezeile beginnt mit einem if. Innerhalb der darauffolgenden Klammern wird die zu prüfende Bedingung angegeben, also in diesem Fall ob der Taster gedrückt ist. Ist diese Bedingung wahr (gibt true zurück), wird der in den geschweiften Klammern eingetragene Code ausgeführt.

Wie dir bestimmt aufgefallen ist, wird in der Bedingung ein Vergleichsoperator verwendet, nämlich ein doppeltes Gleichzeichen ( == ). Ein häufiger Fehler besteht darin, dass nur ein einzelnes Gleichzeichen verwendet wird. Für den Arduino steht ein einzelnes Gleichzeichen jedoch nicht "prüfe, ob gleich" sondern für "setze linken gleich rechten Wert".

### Verwendung von else

Mit else kannst du deiner if -Anweisung noch eine zusätzliche Aktion hinzufügen, welche alternativ ausgeführt wird, falls die Bedingung nicht wahr ist. Wenn du also den obenstehenden Code um ein else ergänzt, würde der komplette Sketch so aussehen:

```
#define LED_PIN 1  
#define BUTTON_PIN 3  
  
void setup() {  
    pinMode(LED_PIN, OUTPUT);  
    pinMode(BUTTON_PIN, INPUT);  
}  
void loop() {  
    if (digitalRead(BUTTON_PIN)==HIGH){  
        digitalWrite(LED_PIN, HIGH);  
    } else {  
        digitalWrite(LED_PIN, LOW);  
    }  
}
```



# Schleifen

Wenn du eine LED 50 Mal blinken lassen willst, ist das ganz schön viel Schreibarbeit.

Da Informatiker schreibfaul sind, haben sie sich eine einfache Lösung einfallen lassen: Schleifen .

Eine Schleife führt eine Anweisungen mehrmals aus, bis eine bestimmte Bedingung erfüllt ist.

## Aufbau von Schleifen

Schleifen bestehen aus zwei Teilen, einem Schleifen-Kopf und einem Schleifen-Körper. Anweisungen die wiederholt werden sollen, werden in geschweiften Klammern in den Schleifen-Körper geschrieben. Jetzt müssen wir aber noch fest legen, wie oft diese Anweisungen wiederholt werden sollen. Dies geschieht im Schleifen-Kopf.

Es gibt verschiedene Arten von Schleifen, die je nach Bedarf verwendet werden können. Hier sollen die zwei wichtigsten Schleifentypen vorgestellt werden.

## Die for-Schleife

`for` -Schleifen werden dann verwendet, wenn man genau weiß, wie oft Anweisungen wiederholt werden sollen. In unserem Beispiel wissen wir, dass die LED 50 Mal blinken soll. Der Kopf einer `for` -Schleife besteht aus drei Teilen, welche durch ein Semikolon ( ; ) getrennt sind:

1. Es wird eine Zählvariable erzeugt, welche angibt, wie oft die schleife bereits ausgeführt wurde
2. Eine Bedingung gibt an, bis wann gezählt werden soll. Wie so eine Bedingung aussieht hast du schon einmal bei [if-Anweisungen](#)<sup>1</sup> gesehen!
3. Eine Definition, wie gezählt werden soll. Üblicherweise wird die Zählvariable um `1` erhöht.

```
for (int zaehler = 1; zaehler < 50; zaehler = zaehler + 1) {
    // lasse die LED blinken
}
```

Im diesem Beispiel heißt unsere Zählvariable `zaehler` . Die Bedingung lautet: "Solange `zaehler` kleiner als 50 ist". Nach jedem Durchlauf der Schleife wird `zaehler` um eins erhöht. Deshalb wird der Schleifenkörper 50 Mal ausgeführt.

- Für den Befehl `zaehler = zaehler + 1` wirst du häufig die Kurzschreibweise `zaehler++` finden. Diese macht das Gleiche.
- Natürlich kannst du der Zählvariablen jeden Namen geben. Häufig wird der Name `i` für "index" genutzt.

## Aufgabe 1

Schreibe nun eine Anweisung in den Schleifenkörper, die dir den Wert der Zählvariablen über den seriellen Monitor ausgibt.

Tipp: In [Der Serielle Monitor<sup>2</sup>](#) wird erklärt, wie das geht!

- **a)** Untersuche was passiert, wenn du `zaehler = zaehler + 1` durch `zaehler = zaehler*2` oder `zaehler--` ersetzt.
- **b)** Untersuche was passiert, wenn du `int zaehler = 1` durch `int zeahler = 25` ersetzt.

## Die while-Schleife

In vielen Fällen weißt du zu Beginn noch nicht, wie oft eine Anweisung wiederholt werden soll. Dann kannst du die `while`-Schleife verwenden. Die `while`-Schleife hat einen weniger strikten Aufbau: Der Schleifenkopf besteht aus dem Bezeichner `while` gefolgt von runden Klammern. In diese Klammern wird eine Bedienung geschrieben, die vor jedem Schleifendurchlauf überprüft wird. Solange diese Bedingung `true` ergibt, wird der Schleifenkörper ausgeführt

```
while (bedingung) {
    // lasse die LED blinken
}
```

Du kannst zum Beispiel einen Knopf an den Arduino anschließen und die Schleife nur dann durchlaufen, wenn der Knopf gedrückt wurde.

Achtung: Ein häufig gemachter Fehler ist, dass eine Bedingung immer wahr ist (Zum Beispiel, wenn ihr als Bedingung schreibt `1 > 0`). In diesem Fall wird eure Schleife immer wieder durchlaufen. Man spricht von einer Endlosschleife. In diesem Fall reagiert euer Arduino nicht mehr und es ist relativ schwer herauszufinden, woran das liegt.

## Aufgabe 2

- **a)** Programmiere eine `while`-Schleife, die den Text "Die Aussage stimmt!" über den seriellen Monitor ausgibt, wenn eine Variable `a` größer als 0 ist.
- **b)** Programmiere eine `while`-Schleife, die eine LED blinken lässt, wenn ein Knopf gedrückt wurde.
- **c)** Jede `for`-Schleife lässt sich auch durch eine `while`-Schleife beschreiben. Schreibe die folgende `for`-Schleife in eine `while`-Schleife um:

```
for (int i = 10; i > 0; i--) {
    Serial.print("Countdown: ");
    Serial.println(i);
}
```

<sup>1</sup>. Siehe [3.1.2 if/else - Bedingung](#) ↵

<sup>2</sup>. Siehe [3.1.5 Der Serielle Monitor](#) ↵

# Variablen und Datentypen

Um Daten in Programmen festzuhalten, verwendet man Variablen. Eine Variable ist ein Speichercontainer, der über seinen Namen angesprochen werden kann, und in dem Daten abgelegt werden können. Auf Variablen lässt sich sowohl lesend als auch schreibend zugreifen, der Wert ist also variabel.

## Datentypen

Eine Variable hat immer einen zugeordneten Datentyp, folgende Typen sind für die Arduino programmierung wichtig:

Datentypen	Bedeutung	Beschreibung
boolean	wahr o. falsch	Kann nur zwei Werte annehmen, 1 oder 0.
char	Character	Alphanumerische Zeichen (Buchstaben, Zahlen, Sonderzeichen)
byte	ganze Zahlen	ganze Zahlen von 0 bis 255
int	ganze Zahlen	ganze Zahlen von -32.758 bis 32.767
long	ganze Zahlen	ganze Zahlen von - 2 Milliarden bis 2 Milliarden
float	Fließkommazahlen	Bruchzahlen
String	Zeichenkette	Text bestehend aus ASCII Zeichen
array	Variablenfeld	Eine Reihe von Werten des selben VariablenTyps können gesammelt gespeichert werden

Beim Programmieren gibt es einige Konventionen, das heißt einige Regeln, auf die man sich geeinigt hat, um die Lesbarkeit von Programmcode zu verbessern. Eine davon ist, dass Name von Variablen immer klein geschrieben werden.

## Verwendung der Datentypen

### boolean

Ein Boolean kann nur zwei Werte annehmen, wahr oder falsch (`true` or `false`).

```
boolean testWert = false;
```

Die Zuweisung `= false` steht in diesem Fall für den Startwert der Variable.

### char

Um beispielsweise einen Buchstaben zu speichern, benötigt man den Datentyp `char`. Der Wert wird in einfachen Anführungszeichen ( ' ) übergeben.

```
char testWert = 'a';
```

## byte

Ein Byte speichert eine 8-bit große, vorzeichenlose Zahl von 0 bis 255.

```
byte testWert = 18;
byte testWert2 = B10010;
```

Das `B` kennzeichnet, dass die folgende Zahlenfolge im Binärkode geschrieben ist. `B10010` entspricht 18 im Dezimalsystem, beide Variablen enthalten also den selben Wert mit unterschiedlicher Schreibweise.

## int

Der Datentyp `int` speichert ganze Zahlen in einem Wertebereich von -32.768 bis 32.767.

```
int testWert = 99;
```

## long

Der Datentyp `long` wird dann benötigt, wenn der Wertebereich eines Integer nicht mehr ausreicht. Es können ganze Zahlen von -2 Milliarden bis 2 Milliarden gespeichert werden.

```
long testWert = 9999999;
```

## float

Um gebrochene Zahlen zu speichern benötigt man den Datentyp `float`.

```
float testWert = 2.4476;
```

## String

Ein String wird folgendermaßen definiert:

```
String testWert = "Hallo Welt";
```

Im Gegensatz zu den Datentypen, die ihr zuvor kennen gelernt habt, wird der Bezeichner `String` groß geschrieben. Darauf müsst ihr achten, sonst erkennt das Programm den Datentyp nicht. In den meisten Programmiersprachen gibt es primitive Datentypen und höhere Datentypen. Du erkennst sie daran, ob ihre Bezeichner klein (primitiver Datentyp) oder groß (höherer Datentyp) geschrieben werden. Für unsere Anwendungen in der senseBox:edu ist es nicht notwendig zwischen primitiven und höheren Datentypen zu unterscheiden; wenn du später komplexere Anwendungen programmierst, wirst du mehr darüber lernen. Möchtest du jetzt schon mehr darüber erfahren, dann schaue doch [hier<sup>1</sup>](#) nach.

## array

Ein Array ist kein eigentlicher Datentyp, sondern viel mehr eine Sammlung mehrerer Variablen desselben Typs.

```
int testArray[5] = {5, 10, 15, 20, 15};
```

Im Beispiel wird ein Array vom Typ `int` angelegt, da ganze Zahlen gespeichert werden sollen. Die 5 in eckigen Klammern hinter dem Namen der Variable legt die Anzahl an Speicherplätzen fest. Arrays auf dem Arduino haben eine feste Größe, und können nicht nachträglich vergrößert werden.

Die Speicherplätze eines Arrays werden bei 0 beginnend durchnummeriert. In einem Programm lässt sich auf die verschiedenen Speicherplätze des Arrays zugreifen, indem der Index des Speicherplatzes in eckigen Klammern hinter den Variablennamen gestellt wird:

```
Serial.print(testarray[0]); // gibt 5 aus
Serial.print(testarray[4]); // gibt 15 aus
Serial.print(testarray[5]); // erzeugt einen Fehler!
```

## Lebensdauer von Variablen

Eine Variable ist immer in dem Block (innerhalb der geschweiften Klammern) für das Programm sichtbar, in welchem die Variable deklariert wurde. Man unterscheidet zwischen globalen und lokalen Variablen. Lokale Variablen sind all diejenigen, welche innerhalb geschweifter Klammern (meist innerhalb einer Funktion) deklariert wurden. Globale Variablen werden üblicherweise vor der `setup`-Funktion definiert und sind für das gesamte Programm sichtbar.

Da globale Variablen immer sichtbar sind, verbrauchen sie auch für die gesamte Programmlaufzeit Speicherplatz. Willst Speicherplatz sparen, definiere Variablen nur dort wo du sie benötigst. Wenn du mehr über die Lebensdauer von Variablen erfahren willst, schaue bei [Wikipedia](#) nach.

---

<sup>1</sup>. [https://de.wikipedia.org/wiki/Datentyp#Zusammengesetzte\\_Datentypen](https://de.wikipedia.org/wiki/Datentyp#Zusammengesetzte_Datentypen) "Wikipedia" ↩

## Der Serielle Monitor

**Der serielle Monitor ist ein Werkzeug um Daten über die USB-Verbindung der senseBox MCU direkt in der IDE anzeigen zu lassen und Daten von der Computertastatur an die senseBox MCU zu übertragen.**

Mit diesem seriellen Monitor kann man sich am PC Daten anzeigen lassen, die der Mikrocontroller an den PC sendet (Zahlen oder Texte). Das ist sehr sinnvoll, da man nicht immer ein LCD Display am Mikrocontroller angeschlossen hat, auf dem man bestimmte Werte ablesen könnte.

### Den seriellen Monitor starten

Um den seriellen Monitor zu starten, musst du zuerst die IDE öffnen und dann in der Symbolleiste auf das Symbol mit der kleinen Lupe klicken.



*Lupen-Symbol*

Das nun geöffnete Fenster hat oben eine Eingabezeile mit "Senden"-Schaltfläche und darunter ein Ausgabefenster. Im Ausgabefenster werden fortlaufend die neusten Ausgaben angezeigt. Wenn das Häkchen bei Autoscroll gesetzt ist, werden nur die aktuellsten Ausgaben angezeigt. Das heißt, wenn das Ausgabefenster voll ist, werden ältere Daten nach oben aus dem sichtbaren Bereich des Bildschirms geschoben um Platz für die aktuellen Ausgaben zu schaffen. Deaktiviert man die Autoscroll Funktion, muss manuell über den Scrollbalken am rechten Rand gescrollt werden.



*serieller Monitor*

### Werte auf dem seriellen Monitor ausgeben

Um sich Daten im seriellen Monitor anzeigen lassen zu können, muss dieser zuerst initialisiert werden. Dies passiert über die Funktion `Serial.begin(9600)` in der `setup()` Funktion. Der Wert `9600` definiert die Baud-Rate, also die Geschwindigkeit mit der Daten zwischen Computer und Arduino übertragen werden. Der eingetragene Wert muss immer der im seriellen Monitor unten rechts ausgewählten Geschwindigkeit entsprechen.

Um Daten an den seriellen Monitor zu senden, verwendet man die Funktionen `Serial.print()` und `Serial.println()`. Die erste Variante der Funktion gibt einfach die Daten aus, während die zweite Variante einen Zeilenumbruch am Ende einfügt.

Als ersten Versuch sollst du jetzt Text im Ausgabefenster anzeigen lassen. Um Text anzeigen zu lassen, muss dieser in Anführungszeichen in den Klammern der Funktion stehen:

```
Serial.println("senseBox rocks!");
Serial.print("senseBox ");
Serial.println("rocks!");
```

Das Beispiel sollte in je einer Zeile den Text "senseBox rocks!" ausgeben. Beachte die Verwendung von `print` und `println`!

Neben Text kann man sich im seriellen Monitor auch die Inhalte von Variablen anzeigen lassen. Dazu muss statt dem gewünschten Text der Name der jeweiligen Variable eingetragen werden:

```
String beispielvariable = "hallo welt!";
Serial.println(beispielvariable);
```

# Digitale Signale

Digitale Signale können lediglich die Werte 1 oder 0 bzw. High oder Low annehmen. Sie verwenden also nur abzählbare Elemente wie zum Beispiel Finger. Daher auch der Begriff digital, der auf das lateinische digitus, der Finger zurück geht. Um zu verstehen wie digitale Signale funktionieren und wo wir mit ihnen Arbeiten müssen wir den Aufbau eines Sketches in Arduino verstehen.

## Digitale Aktoren ansteuern

Die digitalen Schnittstellen der senseBox-MCU verfügen über 4 Anschlüsse: Einen Ground(GND), eine Stromversorgung(5V) und zwei digitale Pins mit denen du die Akteure ansteuern kannst !

Um nun einen digitalen Aktor - beispielsweise eine LED - anzusteuern, benötigt man zwei Befehle: Der Erste steht im `setup()`, der Zweite im `loop()`. In der `setup`-Funktion wird mit dem Befehl `pinMode(1, OUTPUT);` festgelegt, dass an Pin Nummer 1 etwas angeschlossen ist, was als Ausgang (oder OUTPUT) benutzt werden soll. Die 1 kann hier durch jede andere Pin-Nummer ersetzt werden, je nachdem an welchen Arduino-Pin man den Aktor angeschlossen hat. Die zweite Funktion im `loop()` lautet `digitalWrite (1, HIGH);`. Damit wird der an Pin 1 angeschlossene Aktor mit Strom versorgt, also angeschaltet. Das Gegenstück zu diesen Befehl wäre `digitalWrite(1, LOW);` um die Stromversorgung wieder zu beenden. Auch hier ist die 1 wieder durch jede andere Pinnummer ersetzbar. Der Sketch sollte also wie folgt aussehen:

```
void setup(){
    pinMode(1,OUTPUT); //Deklarier den Pin an dem die LED
                        // angeschlossen ist, als Ausgang
}
void loop(){
    digitalWrite(1,HIGH); // Schalte die LED an
}
```

## Digitale Sensoren auslesen

Dieselben Pins die wir zum Ansteuern von digitalen Aktoren genutzt haben, lassen sich auch zur Registrierung von Eingangssignalen verwenden. Digitale Eingänge können dabei genau wie digitale Ausgänge zwei Zustände annehmen; `HIGH` oder `LOW`. Damit eingehende Signale verarbeitet werden können, müssen diese in [Variablen<sup>1</sup>](#) gespeichert werden.

Um digitale Signale zu speichern, eignet sich besonders eine boolesche Variable (auch `boolean` genannt), welche nur zwei Werte annehmen kann. Um nun einen digitalen Sensor auszulesen, werden ähnlich wie beim Ansteuern digitaler Sensoren zwei Befehle benötigt. Im `loop()` wird durch den Befehl `pinMode(1, INPUT);` Pin 1 des Arduino als Eingang festgelegt. Im `setup()` kann durch den Befehl `Testvariable = digitalRead(1);` ein an Pin 1 angeschlossener Sensor ausgelesen und der Wert in der zuvor angelegten Testvariable gespeichert werden. Genau wie beim Ansteuern von digitalen Aktoren steht die 1 für den verwendeten Pin und kann durch jeden anderen digitalen Pin ersetzt werden. Der Sketch sollte also wie folgt aussehen:

```
boolean TestVariable = 0;           // deklariere eine neue boolean Variable

void setup() {
    pinMode(1,INPUT);
}

void loop() {
    TestVariable = digitalRead (1); // schreibe den gelesenen Wert in die Variable
}
```

Den Inhalt der angelegten Variable kannst du dir im [seriellen Monitor](#)<sup>2</sup> anzeigen lassen.

---

1. Siehe [3.1.4 Variablen](#) ↵

2. Siehe [3.1.5 Der Serielle Monitor](#) ↵

## Der serielle Datenbus

Der Arduino kann über einen Datenbus mit anderen Geräten kommunizieren. Ein Datenbus beschreibt ein System, über das zwei oder mehr Geräte Daten auf eine geordnete Art und Weise austauschen können. Bei unserem Arduino wäre das zweite Gerät fast immer ein Sensor, bzw ein Aktor.

## Der I<sup>2</sup>C-Bus

Der I<sup>2</sup>C-Bus ist ein einfach zu verwendender Datenbus um Daten zu übermitteln. Hierbei werden die Daten zwischen dem Arduino und dem anderen Gerät durch zwei Kabel übertragen, die als `SDA` und `SCL` bezeichnet werden. Die als `SDA` (serial data) bezeichnete Leitung ist die Datenleitung, über welche die eigentlichen Daten übermittelt werden. Die `SCL` (serial clock) Leitung wird auch Taktleitung genannt und gibt die Taktfrequenz vor. Am Arduino findest du die beiden Anschlüsse als `A4` (`SDA`) und `A5` (`SCL`).

Wenn mehrere I<sup>2</sup>C Geräte an den Arduino angeschlossen werden sollen, wird dies über eine Reihenschaltung umgesetzt. Das SDA Kabel am ersten Sensor würde also auf der selben Reihe des Breadboards zum nächsten Sensor verlängert:



*senseBox MCU mit Breadboard*

Benutzt man den I<sup>2</sup>C-Bus auf dem Arduino, gilt der Arduino immer als Master-Gerät und alle anderen Geräte am Bus als Slave. Jeder Slave hat seine eigene Adresse in Form einer Hexadezimalzahl, mit welcher er eindeutig angesprochen werden kann. Für gewöhnlich bringt jedes Gerät einen Bereich von Busadressen mit, welche man verwenden kann. Die jeweiligen Adressen können im Datenblatt des Herstellers nachgeschaut werden.

## Analoge Signale

Im Gegensatz zu digitalen Signalen können analoge Signale sehr viele Werte zwischen hohem und niedrigen Pegel annehmen. Die genaue Anzahl der Werte – die Auflösung des digitalen Eingangs – liegt bei der senseBoxMCU bei 1024 Werten (10 bit). Beim senseBoxMCU entspricht der maximale Pegel 5 V und der niedrige 0 V. Diese individuellen Spannungswerte können mit den sechs analogen Pins (A0 – A5) der senseBoxMCU gemessen werden.

## Analoge Akteure ansteuern

Der Befehl `analogWrite()` gibt eine Spannung auf einen angegeben Pin aus. Er kann benutzt werden um beispielsweise einen Motor in verschiedenen Geschwindigkeiten laufen zu lassen.

Es ist auch möglich an auf digitalen Pins mehrere Spannungswerte auszugeben. Hierzu generiert die senseBoxMCU eine stetige Rechteckwelle mit der gewünschten Einschaltzeit und simuliert so ein analoges Signal (Pulsweitenmodulation oder PWM). An der senseBoxMCU kann jeder PIN diese Funktion übernehmen. Die Syntax für den Befehl lautet ähnlich wie beim digitalen Gegenstück `analogWrite(pin,)`. Der Wert kann zwischen 0 (immer aus) und 255 (immer an) liegen.

Ein Beispielhaftes Programm könnte so aussehen:

```
void setup(){
}
void loop(){
    analogWrite(A1, 60) // Ansteuern der LED
}
```

## Analoge Sensoren auslesen

Der Befehl `analogRead()` liest den Wert von einem analogen Pin. Die so gemessenen Werte zwischen 0 V und 5 V werden vom eingebauten 10-bit analog zu digital Konverter (ADC) in integer Werte zwischen 0 und 1023 umgewandelt, d.h. das Signal verfügt über eine Auflösung von 0.0049 Volt pro Wert.

Das auslesen eines Eingangs dauert etwa 0,0001 Sekunden, es können also etwa 10.000 Messungen pro Sekunde aufgenommen werden. Es bietet sich an die gemessenen Daten im [seriellen Monitor<sup>1</sup>](#) anzuzeigen.

Ein Beispielhaftes Programm könnte so aussehen:

```
int val = 0;
string analogPin = 'A1';

void setup(){
    Serial.begin(9600);
}

void loop(){
    val = analogRead(analogPin); // Auslesen des Sensors
    Serial.println(val) // Messwert im Monitor ausgeben
}
```

---

|| 1. Siehe [3.1.5 Der Serielle Monitor ↵](#)

## Bees

Ein Bee bezeichnet eine aufsteckbare Komponente, mit welcher die senseBoxMCU Daten übertragen oder speichern kann. Hier hast du die Wahl zwischen der WiFi-Bee oder der mSD-Bee.

### WiFi-Bee

Das WiFi-Bee ist das Verbindungsstück, um die senseBox mit dem Internet zu verbinden. Deine Messwerte werden per WLAN(WiFi) in das bestehende Netzwerk übertragen. Das WiFi-Bee basiert auf dem ATWINC15000 Mikrochip von Atmel, welcher einen sehr geringen Energieverbrauch und eine hohe Reichweite hat.

#### Konfigurierung der WiFi-Bee & Hochladen auf der openSenseMap

Stelle sicher, dass du das aktuellste Board-Support-Package installiert hast, da du die korrekten Software-Bibliotheken benötigst. Wie das geht wurde dir in [Schritt 2](#) erklärt!

#### ▼ Deklarierung der Objekte

Als erstes muss eine Instanz der Bee und der openSenseMap erstellt werden.

```
Bee* b = new Bee(); // Instanz der Bee
OpenSenseMap osem("senseBox ID",b); // Instanz der openSenseMap
float temp = 24.3; // Testwert den wir später auf der openSenseMap hochladen
```

Achte hier darauf, dass du den Parameter "senseBox ID" mit deiner Box ID ersetzen musst!

Haben wir dies getan, kann die Bee im Programmcode fortlaufend mit dem Kürzel `b` angesprochen werden. In der `setup()`-Funktion stellen wir nun eine Verbindung zu unserem gewünschten WiFi-Netzwerk her und laden testweise einen ersten Wert auf die openSenseMap hoch.

#### ▼ setup()

```
void setup(){
    b->connectToWifi("SSID", "PW"); // Verbindung zum WiFi herstellen
    delay(1000);
    osem.uploadMeasurement(temp,"sensor ID") // Testwert wird hochgeladen
                                                //"sensor ID" muss noch ersetzt werden
};
```

Achte hier darauf, dass du die Parameter "SSID" mit dem Netzwerknamen deines WiFi-Netzwerks, "PW" mit dem dazugehörigen Passwort und "sensor ID" mit der Sensor-ID des entsprechenden Sensors ersetzen musst!

Nun hat deine WiFi-Bee eine Verbindung zum Internet hergestellt und sollte einen ersten Wert auf der openSenseMap hochladen.

Glückwunsch du hast soeben deine ersten Daten auf der Karte hochgeladen, nun bist du bereit für deine erste Umweltmessstation<sup>1</sup>.

---

<sup>1</sup>. Siehe ../../projekte/Umweltstation/README.md ↗

## Breadboard

In der senseBox:edu findest du die senseBox MCU zusammen mit einem "Breadboard" auf einer Halterung. Was dieses "Breadboard" kann und wie es aufgebaut ist, findest du hier.

### 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 (auf der folgenden Abbildung horizontalen) Reihen für die Plus- und Minusanschlüsse sowie zweimal 30 (auf der folgenden Abbildung vertikalen) Reihen mit je fünf Federkontakten, die mit a bis e bzw. f bis j beschriftet sind. Die Plus- und Minusanschlüsse sowie die fünf horizontalen Federkontakte einer Reihe sind, wie unten dargestellt, leitend miteinander verbunden.



Breadboard

## Registrierung der sensebox:edu als home auf der openSenseMap

Registriere deine senseBox jetzt auf der openSenseMap, um Umweltmessdaten zu sammeln und die einer Community aus Forschern und Tüftlern zur Verfügung zu stellen und zu der Vermessung der Umwelt beizutragen. Werde so Teil einer der größten citizenScience Bewegungen weltweit!

### Achtung!

Die openSenseMap ist ein Portal auf dem weltweit Menschen Messdaten ihrer senseBoxen oder anderen Messstationen hochladen. Diese Messdaten werden von Privatpersonen, aber auch von Forschern, zu wissenschaftlichen Zwecken benutzt. Du solltest die senseBox:edu daher nur auf der oSeM registrieren, falls du längerfristig Daten hochladen möchtest und diese echte Umweltmessungen darstellen und nicht zu experimentalen Zwecken verfälscht werden. Möchtest du deine senseBox:edu also dennoch registrieren, musst du sie dabei einfach als senseBox:home-v2 angeben.

Du weisst noch gar nicht was die openSenseMap ist? Dann schau sie dir jetzt an, gehe auf [www.opensensemap.org](https://www.opensensemap.org)<sup>1</sup> und entdecke ein riesigen Pool aus open-data mit fast einer Milliarde Messungen weltweit!

Du hast dir die openSenseMap längst angesehen? - Na worauf wartest du noch? Gehe jetzt auf die [openSenseMap](https://opensensemap.org)<sup>2</sup> und registriere deine senseBox, um Messungen hochladen, betrachten und verarbeiten zu können.

---

<sup>1</sup>. <https://www.opensensemap.org> ↵

<sup>2</sup>. <https://opensensemap.org> ↵

# DIY-Umweltstation

In diesem Projekt erfahrt ihr, wie man eine senseBox Umweltstation aufbaut. Am Ende wird die Messung diverser Umweltphänomene wie Temperatur, Luftfeuchte, Helligkeit und Luftdruck, sowie die Veröffentlichung der Daten auf der [openSenseMap](#) möglich sein!

---

## Einleitung

Dieses Projekt ist das umfangreichste, weshalb es in mehrere Unterkapitel aufgeteilt wurde. In jedem neuen Kapitel wird ein zusätzlicher Baustein eingeführt, bis eine vollständige - den Funktionen der senseBox:home ähnliche - Wetterstation gebaut wurde!

Innerhalb dieses Projekts findet ihr die folgenden 5 Kapitel:

- [Temperatur und Luftfeuchtigkeit<sup>1</sup>](#)
- [Experimente mit Licht<sup>2</sup>](#)
- [UV-Sensor<sup>3</sup>](#)
- [Luftdruck<sup>4</sup>](#)
- [Datenupload zur OSeM<sup>5</sup>](#)

Ihr könnt die Stationen als einzelne Projekte bearbeiten, oder euren Programm-Code mit jeder Station erweitern, um alle fünf Phänomene gleichzeitig messen zu können.

Viel Erfolg beim Bau deiner DIY-Umweltstation

---

<sup>1</sup>. Siehe [4.1.1 Temperatur und Luftfeuchtigkeit](#) ↵

<sup>2</sup>. Siehe [4.1.2 Experimente mit Licht](#) ↵

<sup>3</sup>. Siehe [4.1.3 UV-Sensor](#) ↵

<sup>4</sup>. Siehe [4.1.4 Luftdruck](#) ↵

<sup>5</sup>. Siehe [4.1.5 Datenupload zur OSeM](#) ↵

## DIY - Temperatur und Luftfeuchtigkeit

Damit wir täglich den Wetterbericht im Internet, im Fernsehen, in der Zeitung oder in Apps sehen können, werden nicht nur Satellitendaten ausgewertet. Auch Daten von Wetterstationen am Boden spielen eine wichtige Rolle bei der Vorhersage. Aber wie funktioniert die Messung und Darstellung von Temperatur- und Luftfeuchtigkeitswerten?

### Vorraussetzungen

- Die Verwendung von Software-Bibliotheken<sup>1</sup>
- Der serielle Datenbus I<sup>2</sup>C<sup>2</sup>
- Der serielle Monitor<sup>3</sup>

### Ziele der Station

In dieser Station beschäftigen wir uns mit dem Temperatur- und Luftfeuchtigkeitssensor der senseBox, dem HDC1080.

### Materialien

- kombinierter Temperatur und Luftfeuchtigkeitssensor `HDC1080`

### Grundlagen

#### ▼ `HDC1080 Sensor`

Der `HDC1080`, aus der Serie HDX10XX von Texas Instruments, ist ein kombinierter Temperatur- und Luftfeuchtigkeitssensor. Der Sensor kann die Luftfeuchtigkeit von 0% bis 100%, sowie die Temperatur von -40°C bis 125°C bei einer Genauigkeit von ±2% bzw. von ±0,2°C messen.

#### ▼ `I2C Bus`

Die Kommunikation des Sensors mit dem Mikrocontroller läuft über den `seriellen Datenbus I2C`<sup>4</sup>. Anders als bei einfachen digitalen oder analogen Eingängen, können an den Datenbus mehrere I<sup>2</sup>C-Geräte (wie z.B. Sensoren oder Displays) parallel geschaltet werden. Jedes Gerät hat dabei eine eindeutige Kennung, damit der Datenbus jedes Einzelne davon zuordnen und separat ansprechen kann.

### Aufbau

Steckt den Schaltkreis wie ihr ihn unten in der Grafik seht.



Temperatur- und Luftfeuchtigkeitssensor angeschlossen über I2C-Port

## Programmierung

Stelle sicher, dass du das aktuellste Board-Support-Package installiert hast, da du die korrekten Software-Bibliotheken benötigst. Wie das geht wurde dir in [Schritt 2](#) erklärt!

Als erstes muss eine Instanz des Sensors angelegt werden.

```
#include "SenseBoxMCU.h"  
HDC1080 hdc;
```

### ▼ setup() Funktion

In der `setup()`-Funktion soll der Sensor nun gestartet werden:

```
void setup(){  
    hdc.begin();  
}
```

### ▼ loop() Funktion

Nachdem du den Sensor, wie oben beschrieben, initialisiert hast, kannst du zwei Befehle in der `loop()`-Funktion nutzen, um einen Temperatur- bzw. Feuchtigkeitswert ausgeben zu lassen:

```
void loop(){  
    hdc.getHumidity();  
    hdc.getTemperature();  
}
```

Beim Speichern der Messwerte sollten die Variablen den gleichen Datentypen haben wie die Rückgabewerte der Messfunktionen. In unserem Fall sind das beides float Werte.!

## Aufgaben

### ▼ Aufgabe 1

Baue die oben beschriebene Schaltung nach und versuche den HDC1008 auszulesen und Dir die gemessenen Daten im seriellen Monitor anzeigen zu lassen.

Schau dir hierfür die Beispiele aus den [Ersten Schritten](#) an!

---

1. Siehe [2.2 Schritt 2: Board-Support-Packages installieren](#) ↵

2. Siehe [3.3 Der serielle Datenbus\(I<sup>2</sup>C\)](#) ↵

3. Siehe [3.1.5 Der Serielle Monitor](#) ↵

4. Siehe [3.3 Der serielle Datenbus\(I<sup>2</sup>C\)](#) ↵

## DIY - Experimente mit Licht

Wenn du fern siehst, das Radio anschaltest, mit deinem Smartphone eine Nachricht schreibst oder Essen in der Mikrowelle warm machst, nutzt du dabei elektromagnetische Energie. Heutzutage sind alle Menschen ständig auf diese Energie angewiesen. Ohne sie würde das Leben in modernen Städten völlig anders sein, als du es kennst.

### Vorraussetzungen

- Die Verwendung von Software-Bibliotheken<sup>1</sup>
- Der serielle Datenbus I<sup>2</sup>C<sup>2</sup>
- Der serielle Monitor<sup>3</sup>

### Ziele der Station

In dieser Station verwendest du einen Lichtsensor, um die Beleuchtungsstärke des sichtbaren Lichts in Lux zu erfassen.

### Materialien

- Lichtsensor TSL 45315

### Grundlagen

#### ▼ Lichtintensität

Elektromagnetische Energie bewegt sich in Wellen durch den Raum. Ihr Spektrum reicht von sehr langen Radiowellen bis hin zur sehr kurzweligen Gammastrahlung. Das menschliche Auge kann dabei nur einen sehr kleinen Teil dieses Spektrums wahrnehmen: das sichtbare Licht. Unsere Sonne ist dabei die Quelle der Energie über das gesamte Spektrum hinweg. Die Atmosphäre der Erde schützt uns davor, einem zu hohen Maß an Strahlung ausgesetzt zu werden, die für uns lebensgefährlich werden könnte.

Für uns ist die Intensität des sichtbaren Lichts besonders interessant. Um die sog. Beleuchtungsstärke des einfallenden Lichts im sichtbaren Teil des Spektrums zu messen, wird die Einheit Lux verwendet. Sie gibt das Verhältnis der Helligkeit in Lumen pro Quadratmeter an. Bei einem hellen Sonnentag beträgt sie über 100.000 Lux, in einer Vollmondnacht hingegen nur etwa 1 Lux.

#### ▼ TSL45315 Sensor

Für diese Messung benutzen wir im Folgenden den Sensor TSL45315 von AMS-TAOS. Im Datenblatt des Sensors sieht man, dass seine Empfindlichkeit auf den sichtbaren Teil des Lichtspektrums angeglichen ist, der ungefähr zwischen 400 und 700 nm liegt. Laut dem Datenblatt hat dieser Sensor eine Reichweite von 2 bis 200.000 Lux, bei einer Auflösung von 3

Lux. Des Weiteren muss der Sensor mit 3,3V betrieben werden.

Der Sensor wird über das I<sup>2</sup>C Protokoll angesprochen. Wir sprechen ihn direkt mit den folgenden aus dem Datenblatt entnommenen Befehlen an:



## Aufbau



*Belichtungs- und UV-Sensor angeschlossen über I2C-Port*

## Programmierung

Stelle sicher, dass du das aktuellste Board-Support-Package installiert hast, da du die korrekten Software-Bibliotheken benötigst. Wie das geht wurde dir in [Schritt 2](#) erklärt!

Als erstes muss eine Instanz des Sensors angelegt werden.

```
#include "SenseBoxMCU.h"  
TSL45315 lux_sensor;
```

### ▼ setup() Funktion

In der `setup()`-Funktion soll der Sensor nun gestartet werden:

```
void setup(){  
    lux_sensor.begin();  
}
```

### ▼ loop() Funktion

In der `loop()`-Funktion können wir mit dem Befehl 'getIlluminance()' die aktuelle gemessene Lichtintensität abrufen:

```
void loop(){  
    lux_sensor.getIlluminance();  
}
```

## Aufgaben

▼ Aufgabe 1

Füge den Code aus dieser Lektion zusammen und ergänze eine Funktion um die Daten im Seriellen Monitor ausgeben zu lassen.

▼ Aufgabe 2

Versucht abhängig von der Beleuchtung eine LED an und auszuschalten. Hierfür kann das Kapitel [if/else – Bedingung<sup>4</sup>](#) hilfreich sein.

---

1. Siehe [2.2 Schritt 2: Board-Support-Packages installieren](#) ↵

2. Siehe [3.3 Der serielle Datenbus\(I<sup>2</sup>C\)](#) ↵

3. Siehe [3.1.5 Der Serielle Monitor](#) ↵

4. Siehe [../../grundlagen/if\\_else\\_bedingung.md](#) ↵

# DIY - UV-Licht Sensor

Wenn die Temperaturen im Sommer steigen und wir uns länger draußen aufhalten, versuchen wir uns zunehmend vor der UV-Einstrahlung der Sonne zu schützen, zum Beispiel mit Sonnencreme. Doch gibt es diese UV-Strahlung nur im Sommer? Wie sieht es aus wenn die Sonne verdeckt ist und wie stark schwanken die Werte? Finde heraus wie viel UV-Strahlung die Erde erreicht und messe die UV-Intensität mit deiner senseBox!

## Vorraussetzungen

- Die Verwendung von Software-Bibliotheken<sup>1</sup>
- Der serielle Datenbus I<sup>2</sup>C<sup>2</sup>
- Der serielle Monitor<sup>3</sup>

## Ziele der Station

In dieser Station verwenden wir einen UV-Lichtsensor, um die Intensität des UV-Lichts in Mikrowatt je Quadratzentimeter ( $\mu\text{W} / \text{cm}^2$ ) zu erfassen. Anschließend wollen wir den Wert in den UV-Index umrechnen.

## Materialien

- UV-Licht Sensor VEML6070

## Grundlagen

### ▼ UV-Strahlung

Ultraviolettrahlung (UV-Licht) ist für den Menschen unsichtbare elektromagnetische Strahlung mit einer Wellenlänge, die kürzer ist als die sichtbaren Lichtes, aber länger als die der Röntgenstrahlung. UV-Licht umfasst die Wellenlängen von 100 nm bis 380 nm. Wegen der Absorption in der Erdatmosphäre - insbesondere in der Ozonschicht - dringt nur wenig UV-B-Strahlung (100 - 300 nm) bis zur Erdoberfläche vor. UV-A-Strahlung (300 - 380 nm), welche weniger gefährlich für die menschliche Haut ist, wird weniger durch die Atmosphäre absorbiert.

UV-Lichtintensität wird in Mikrowatt je Quadratzentimeter ( $\mu\text{W} / \text{cm}^2$ ) gemessen. Unser Sensor misst im Bereich von ca. 300 - 400 nm, nimmt also nur UV-A Strahlung auf (für genauere Angaben beachte das [Datenblatt<sup>1</sup>](#)).

## Aufbau

*Belichtungs- und UV-Sensor angeschlossen über I2C-Port*

Schließe den Sensor an die senseBoxMCU an, wie es in der Grafik dargestellt ist.

## Programmierung

Stelle sicher, dass du das aktuellste Board-Support-Package installiert hast, da du die korrekten Software-Bibliotheken benötigst. Wie das geht wurde dir in [Schritt 2](#) erklärt!

Als erstes muss eine Instanz des Sensors angelegt werden.

```
#include "SenseBoxMCU.h"
VEML6070 vml;
```

### ▼ setup() Funktion

In der `setup()`-Funktion soll der Sensor nun gestartet werden:

```
void setup(){
    vml.begin();
}
```

### ▼ loop() Funktion

In der `loop()`-Funktion können wir mit dem Befehl `'getIlluminance()'` die aktuelle gemessene Lichtintensität abrufen:

```
void loop(){
    vml.getIlluminance();
}
```

Möchtest du dir den UV-Index anzeigen lassen so musst du dir vorher eine Funktion deklarieren die das für dich übernimmt. Wie das geht erfährst du im nächsten Schritt!

### ▼ Umrechnung in den UV-Index

Da im Alltag häufig mit dem [UV-Index<sup>2</sup>](#) gearbeitet wird, wollen wir nun eine Methode schreiben, welche uns den Messwert in einen UV-Index umrechnet:

```
/*
 * getUVI()
 * erwartet den Messwert des UV-Sensors als Eingabeparameter
 * und gibt den entsprechenden Wert auf dem UV-Index zurück
 */
float getUVI(int uv) {
```

```
float refVal = 0.4; // Referenzwert: 0,01 W/m^2 ist äquivalent zu 0.4 als UV-Index
float uvi = refVal * (uv * 5.625) / 1000;
return uvi;
}
```

## Aufgaben

### ▼ Aufgabe 1

Versuche dir im seriellen Monitor nun mithilfe der `getUVI()`-Funktion den UV-Index ausgeben zu lassen.

1. Siehe [2.2 Schritt 2: Board-Support-Packages installieren](#) ↵
  2. Siehe [3.3 Der serielle Datenbus\(I<sup>2</sup>C\)](#) ↵
  3. Siehe [3.1.5 Der Serielle Monitor](#) ↵
1. [https://github.com/sensebox/resources/raw/master/datasheets/datasheet\\_veml6070-UV-A-Light-Sensor.pdf](https://github.com/sensebox/resources/raw/master/datasheets/datasheet_veml6070-UV-A-Light-Sensor.pdf)  
↪
2. <https://de.wikipedia.org/wiki/UV-Index> ↵

# Luftdruck

Die Messung des Luftdrucks erlaubt neben Wettervorhersagen auch indirekt die Bestimmung der Höhe des Sensors.

## Vorraussetzungen

- Die Verwendung von Software-Bibliotheken<sup>1</sup>
- Der serielle Datenbus I<sup>2</sup>C<sup>2</sup>
- Der serielle Monitor<sup>3</sup>

## Materialien

- Luftdrucksensor BMP280

## Grundlagen

### ▼ BMP280 Sensor

Der BMP280 Sensor misst sowohl Luftdruck (hPa) als auch Temperatur (°C). Dieser Sensor wird über das I<sup>2</sup>C Protokoll<sup>4</sup> angesteuert, und benötigt eine Betriebsspannung von 3.3 bis 5 Volt.

I<sup>2</sup>C-Geräte werden an die senseBoxMCU über den I<sup>2</sup>C/Wire Port angeschlossen, und so digital ausgelesen (siehe auch [Der serielle Datenbus<sup>5</sup>](#)). Die I<sup>2</sup>C-Adresse des BMP280 kann über den `sdo` Pin umgeschaltet werden: Liegt `sdo` auf Masse (`GND`) ist die Adresse `0x76`, sonst `0x77`. Diese Kommunikation übernimmt die senseBox-Bibliothek für uns.

### ▼ Höhenbestimmung über den Luftdruck

Da der Luftdruck von der Höhe über dem Meeresspiegel abhängt, kann über den `BMP280` auch die Aufbau-Höhe der senseBox bestimmt werden. Dazu wird ein Referenzdruck `P0` benötigt, dessen Höhe bekannt ist. Üblicherweise wird dazu der aktuelle Luftdruck auf Meeresspiegelniveau verwendet. Da der Luftdruck in Abhängigkeit vom aktuellen Wetter stark schwanken kann, ist diese "Höhenmessung" aber nicht sehr akkurat, und muss immer wieder neu kalibriert werden.

## Aufbau

Um den Sensor zum Laufen zu bringen schließe ihn einfach an den I<sup>2</sup>C/Wire Port an !



Temperatur- und Luftdrucksensor angeschlossen über I2C-Port

## Programmierung - Auslesen des Sensors

Der Sensor kann über die Bibliothek `SenseBoxMCU.h` angesteuert werden. Nachdem diese eingebunden wurde, muss eine Instanz `bmp` davon erstellt werden. Auf diesem Objekt werden alle Funktionen der Bibliothek aufgerufen:

```
#include <SenseBoxMCU.h>
BMP280 bmp_sensor;
```

### ▼ setup() Funktion

In der `setup()`-Funktion muss der Sensor initialisiert werden. Verwende dazu die folgenden Zeilen:

```
void setup(){
    bmp_sensor.begin();
}
```

### ▼ loop() Funktion

Nun muss der Sensor in der `loop()`-Funktion ausgelesen werden. In den Variablen `temp` und `pressure` stehen dann jeweils die aktuellen Messwerte.

```
void loop(){
    double temp, pressure;
    pressure = bmp_sensor.getPressure();
    temp = bmp_sensor.getTemperature();
}
```

## Aufgaben

### ▼ Aufgabe 1

Verbinde den `BMP280` Sensor mit dem Arduino, und erstelle einen Arduino-Sketch, welcher regelmäßig Luftdruck und Temperatur auf dem seriellen Monitor ausgibt!

### ▼ Aufgabe 2

Du hast erfahren, dass sich aus dem gemessenen Luftdruck die Aufbauhöhe der senseBox bestimmen lässt. Verwende die Funktion `bmp.altitude(...)` um die Höhe zu berechnen, und gib diese ebenfalls auf dem Seriellen Monitor aus.

Sieh dir das der BMP280-Bibliothek beiliegende Beispiel an. Der Referenzdruck P0 muss an die derzeitige Wetterlage angepasst werden: [Hier]([https://www.meteoblue.com/en/weather/webmap/index/?variable=mslp\\_pressure&level=surface&lines=none](https://www.meteoblue.com/en/weather/webmap/index/?variable=mslp_pressure&level=surface&lines=none))[^1] findest du den aktuellen Luftdruck.\*

- 1. Siehe [2.2 Schritt 2: Board-Support-Packages installieren](#) ↵
- 2. Siehe [3.3 Der serielle Datenbus\(I<sup>2</sup>C\)](#) ↵
- 3. Siehe [3.1.5 Der Serielle Monitor](#) ↵
- 4. Siehe [3.3 Der serielle Datenbus\(I<sup>2</sup>C\)](#) ↵
- 5. Siehe [3.3 Der serielle Datenbus\(I<sup>2</sup>C\)](#) ↵
- 1. [https://www.meteoblue.com/en/weather/webmap/index/?variable=mslp\\_pressure&level=surface&lines=none](https://www.meteoblue.com/en/weather/webmap/index/?variable=mslp_pressure&level=surface&lines=none)  
↵

## Datenupload

Wenn wir unsere Wetterstation aufgebaut haben, wäre es doch schön die gewonnenen Daten immer von jedem Ort aus abrufen zu können. Dazu gibt es die [openSenseMap](https://openSenseMap.org/)[^1] (OSeM), welche diverse Sensordaten online sammelt und auf einer Karte darstellt. Über den Ethernet- oder WiFi-Bee oder können wir unsere senseBox ans Internet anbinden und die Daten zur OSeM hochladen.

## Voraussetzungen

- Die Verwendung von Software-Bibliotheken<sup>1</sup>: Du solltest dir zu diesem Zweck die ersten Schritte angeguckt haben.
- Bees<sup>2</sup>: Lese dieses Kapitel um zu erfahren, wie die senseBox MCU mithilfe von Bees eine Netzwerkverbindung herstellen kann, um so Daten auf die openSenseMap zu übertragen.

## Ziele der Station

In dieser Station wird beispielhaft die Integration eines Sensors in die openSenseMap gezeigt, sodass die gewonnenen Daten online verfügbar sind.

## Materialien

- WiFi-Bee
- Mindestens einen (beliebigen) Sensor

## Programmierung

Im Kapitel Bees<sup>3</sup> hast du schon gelernt wie du eine Verbindung zum Internet aufbaust, nun schauen wir uns an, wie wir unsere Messwerte kontinuierlich auf der openSenseMap hochladen können. Wie dort bereits beschrieben müssen wir erstmal die Instanzen für die openSenseMap erstellen und unser WLAN-Netzwerk + Zugangsdaten bereitstellen.

### ▼ Deklarierung der Objekte

```
#include "SenseBoxMCU.h"
Bee* b = new Bee(); // Instanz der Bee
OpenSenseMap osem("senseBox ID",b); // Instanz der openSenseMap
HDC1080 hdc; // Instanz des Temperatur - & Luftfeuchte Sensor
void setup(){
    b->connectToWifi("SSID","PW"); // Verbindung zum WiFi herstellen
    hdc.begin();
};
```

In der `loop()` -Funktion laden wir nun unsere Messwerte hoch.

▼ loop()

```
void loop(){
    osem.uploadMeasurement(hdc.getTemperature(), "Sensor ID")
    delay(5000);
}
```

## Aufgaben

▼ Aufgabe 1

Mache dich mit der openSenseMap vertraut (siehe Voraussetzungen), und registriere dort deine senseBox mit den Sensoren, welche du bisher angeschlossen hast.

▼ Aufgabe 2

In dem Arduino-Sketch, den du bei der Registrierung in Aufgabe 1 erhalten hast, fehlt noch das Auslesen von Sensoren. Erweitere den Sketch von der OSeM-Registrierung sodass deine angeschlossenen Sensoren ausgelesen werden.

Folge der Anleitung im [Grundlagenkapitel](#). Du kannst deinen bisherigen Code überwiegend wiederverwenden!

1. <https://openSenseMap.org/> ↵

1. Siehe [2.2 Schritt 2: Board-Support-Packages installieren](#) ↵

2. Siehe [3.5 Bees](#) ↵

3. Siehe [../../grundlagen/Bees.md](#) ↵

# Ampel

Es soll eine Ampel simuliert werden. Mit einem Button kann man die Ampel umschalten.

## Materialien

- senseBox MCU
- rote LED
- gelbe LED
- grüne LED
- 3x 470Ω Widerstand
- Button
- 10Ω Widerstand
- 2x senseBox JST-Adapterkabel

## Aufbau

### Hardwarekonfiguration

Um alle Komponenten anzuschließen benötigst Du zwei JST - Adapterkabel. Das erste wird an Digital A (also den digitalen Pins 1 und 2) angeschlossen, das zweite an Digital B (also den digitalen Pins 3 und 4) angeschlossen. Am Kabel in Digital A werden die rote und die gelbe LED angeschlossen, am Kabel in Digital B die grüne LED und der Button.



Verkabelung der Ampelschaltung

## Sketch

- Arduino Quellcode
- Blockly

## Arduino Quellcode

```
int rot = 1;
int gelb = 2;
int gruen = 3;

int button = 4;
```

```

void setup() {
    pinMode(rot, OUTPUT);
    pinMode(gelb, OUTPUT);
    pinMode(gruen, OUTPUT);

    // Der Button soll Signale messen, also INPUT
    pinMode(button, INPUT);

    // Ampel zuerst auf ROT setzen
    digitalWrite(rot, HIGH);
    digitalWrite(gelb, LOW);
    digitalWrite(gruen, LOW);
}

void loop() {

    // Hier wird geprüft ob der Button gedrückt wird
    if(digitalRead(button) == HIGH) {

        delay(5000);

        // ROT zu GRUEN
        digitalWrite(rot, HIGH);
        digitalWrite(gelb, HIGH);
        digitalWrite(gruen, LOW);

        delay(1000);

        digitalWrite(rot, LOW);
        digitalWrite(gelb, LOW);
        digitalWrite(gruen, HIGH);

        delay(5000);

        // GRUEN zu ROT
        digitalWrite(rot, LOW);
        digitalWrite(gelb, HIGH);
        digitalWrite(gruen, LOW);

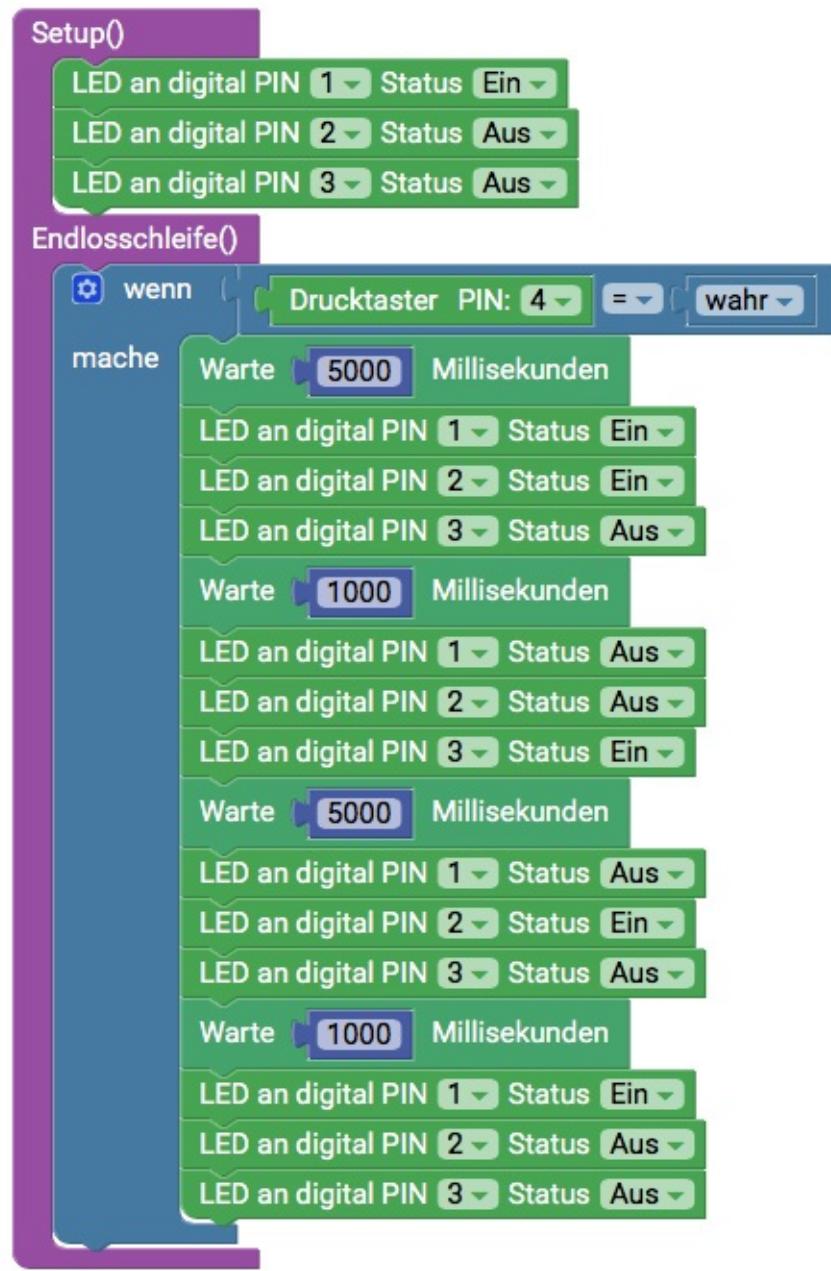
        delay(1000);

        digitalWrite(rot, HIGH);
        digitalWrite(gelb, LOW);
        digitalWrite(gruen, LOW);
    }
}

```

- Am Anfang der `loop()` Funktion wird jedesmal abgefragt ob der Button gedrückt wird.
- `digitalRead(button)` liest den aktuellen Zustand des Buttons aus. Wird er gedrückt, liefert die Funktion `HIGH` aus, ansonsten `LOW`.
- Um zu Prüfen ob der Button gedrückt wurde muss `digitalRead(button)` mit `HIGH` verglichen werden. Der Vergleich geschieht mit **zwei** Gleichzeichen `==` (Vergleichsoperator). **Ein** Gleichzeichen `=` ist eine Zuweisung, wie etwa `int rot = 13`.

## Blockly



Blockly

# Verkehrszähler

Ziel ist es, einen Verkehrs- oder Personenzähler zu entwickeln.

Dazu verwenden wir einen Ultraschall-Distanzsensor. Die so aufgenommenen Werte sollen im Seriellen Monitor ausgegeben werden.

## Materialien

- Ultraschall-Distanzsensor
- senseBox MCU
- senseBox JST-Adapterkabel

## Grundlagen

Der Ultraschall-Distanzsensor nutzt den Schall um die Entfernung von Objekten zu bestimmen. Der Sensor sendet einen Impuls aus und misst die Zeit, bis er das Echo des Impulses wieder empfängt. Aus dieser Zeit errechnet man mit Hilfe der Schallgeschwindigkeit die Entfernung des Objekts.

## Aufbau

Der Ultraschallsensor wird mit einem JST-Adapterkabel mit der senseBox MCU verbunden. Dazu wird das JST-Adapterkabel mit dem Steckplatz Digital A verbunden. Zur Stromversorgung wird der VCC-Pin des Sensors mit dem roten Kabel (5V) und der GND-Pin des Sensors mit dem schwarzen Kabel (GND) verbunden. Zur Datenübertragung wird dann das grüne Kabel (1) mit dem Echo- und das gelbe Kabel (2) mit dem Trig-Pin des Sensors verbunden.



Verkabelung des Ultraschall-Sensors

Hinweis: Ihr könnt natürlich jeden mit "Digital" beschrifteten Steckplatz verwenden, denkt aber daran den Code anzupassen.

## Programmierung

Definiert die Pins an dem ihr den Sensor angeschlossen habt wie üblich. Außerdem werden zwei Variablen angelegt in der die gemessene Zeit und die errechnete Distanz gespeichert werden.

```
int trig = 2; // Trig-Pin des Sensors ist an Pin 2 angeschlossen.  
int echo = 1; // Echo-Pin des Sensors ist an Pin 1 angeschlossen.
```

```
unsigned int time = 0;
unsigned int distance = 0;
```

Im `setup()` müsst ihr nun den Seriellen Monitor starten und die Pins an welchen der Sensor angeschlossen ist als Ein- bzw. Ausgang definieren. Der Trigger-Pin des Sensors muss als Ausgang und der Echo-Pin als Eingang definiert werden.

```
Serial.begin(9600);
pinMode(trig, OUTPUT);
pinMode(echo, INPUT);
```

Im `loop()` wird mit den Befehlen

```
digitalWrite(trig, HIGH);
delayMicroseconds(10);
digitalWrite(trig, LOW);
```

ein 10 Mikrosekunden langer Ultraschallimpuls ausgesendet. Der darauffolgende Befehl `time = pulseIn(echo, HIGH);` speichert die Zeit bis zum Empfang des Echoes in der Variable `time`. Zum Schluss muss noch die Distanz aus der Zeit errechnet werden, sowie die Werte auf dem Seriellen Monitor angezeigt werden.

```
distance = time / 58;
Serial.println(distance);
```

Hinweis Wir gehen davon aus, dass sich der Schall mit 348 Metern pro Sekunde ausbreitet. Diese Zahl ist nicht fix sondern hängt von der Umgebungstemperatur ab<sup>1</sup>.

### ▼ Aufgabe 1

Versucht mit Hilfe bekannter Befehle und dem oben angegebenen Sketch zum Ultraschallsensor einen Personen- bzw. Verkehrszähler zu entwickeln.

Beachtet dabei folgende Hinweise:

- Versucht nur einen bestimmten Entfernungsbereich auszuwerten, damit es nicht zu Störungen durch Bewegungen im Hintergrund kommt. Effektiv misst der Sensor ca. 3 Meter.
- Um Mehrfachzählungen eines stehenden Fahrzeuges zu vermeiden solltet ihr eine Bedingung programmieren, der den Zählvorgang stoppt bis die Spur wieder frei ist, der Sensor also eine vorher festgelegte Maximaldistanz für die Spur misst. Dazu bietet sich ein `while`-Schleife an. Zuerst muss überprüft werden, ob sich etwas im Messbereich befindet. Solange der Sensor nicht misst, dass die Fahrbahn wieder frei ist, soll er erneut messen. Erst wenn die Fahrbahn wieder frei ist erhöhe deine Zählvariable um eins.
- Damit die Messwerte beim einfahren in den Messbereich nicht zu sehr schwanken, kann es helfen, zwischen den einzelnen Messungen eine Verzögerung von 200ms zu programmieren.

---

<sup>1</sup>. [https://de.wikipedia.org/wiki/Schallgeschwindigkeit#Temperaturabh.%C3%A4ngigkeit\\_in\\_Luft](https://de.wikipedia.org/wiki/Schallgeschwindigkeit#Temperaturabh.%C3%A4ngigkeit_in_Luft) ↵

## Mobile Station

Eine mobile senseBox kann Daten auf die openSenseMap hochladen von überall wo du Internet hast!  
Wie das funktioniert wird dir in diesem Kapitel an einem Beispiel erläutert.

## Materialien

- senseBoxMCU
- GPS-Modul
- senseBox JST-Adapterkabel

## Grundlagen

Eine mobile senseBox hat vielerlei Anwendungsfälle. Möchtest du zum Beispiel die Konzentration von Feinstaub entlang deiner täglichen Route zur Arbeit oder Schule messen. Das GPS-Modul empfängt die Position (Längengrad/Breitengrad/Höhe) der senseBox. Dieser Sensor ist kompatibel mit den gängigen GNS Systemen (GPS, QZSS, GLONASS, BeiDou, Galileo) und basiert auf dem u-blox CAM-M8Q Multi GNSS Modul.



Eine mobile senseBox auf der openSenseMap

## Aufbau

Damit das GPS-Modul die ganze Messperiode über ein Signal hat und somit den Standort anfragen kann, musst du sicherstellen, dass das Modul durch nichts verdeckt wird. Idealerweise bringst du es für die Messperiode außerhalb des Gehäuses an.

Für die Stromversorgung während der Messperiode kannst du zum Beispiel eine Powerbank benutzen. Manche Powerbanks schalten sich ab, wird zu wenig Strom gezogen (z.B. dann wenn ein Handy voll geladen ist), dadurch kann es vorkommen, dass die Powerbank die senseBoxMCU nicht dauerhaft mit Strom versorgt. Gehe vor der Messung sicher, dass dies nicht der Fall sein wird.

Es kann sein, dass der GPS Sensor anfangs eine Weile braucht, damit korrekte Signale empfangen werden. Vorallem bei erstmaliger Benutzung kann dieser Prozess bis zu 5 Minuten dauern. Nachdem ein Netzwerk erkannt wurde, werden die Informationen dazu intern im Sensor gespeichert damit es bei der nächsten Verwendung schneller gehen kann !

## Programmierung

Stelle sicher, dass du das aktuellste Board-Support-Package installiert hast, da du die korrekten Software-Bibliotheken benötigst. Wie das geht wurde dir in [Schritt 2](#) erklärt!

Wie das GPS-Modul programmiert wird, wird hier Beispielhaft anhand einer Temperaturmessung veranschaulicht. Jeder Messwert einer mobilen Station wird zusammen mit den dazugehörigen Werten für Breiten- und Längengrad hochgeladen.

Als erstes muss eine Instanz der Sensoren angelegt werden. Zusätzlich definieren wir noch 2 zwei Variablen für Breiten- und Längengrad

```
#include "SenseBoxMCU.h"
HDC1080 hdc;

GPS gps;
float lat; // Geografische Breite
float lon; // Geografische Länge
```

In der `setup`-Funktion starten wir nun die beiden Sensoren.

### ▼ `setup()` Funktion

```
void setup(){
    hdc.begin();
    gps.begin();
}
```

Die `loop`-Funktion fragt nun den Standort der Station ab und lädt diesem gemeinsam mit dem Wert für Temperatur auf die openSenseMap hoch.

### ▼ `loop()` Funktion

```
void loop(){
    lat = gps.getLatitude();
    lon = gps.getLongitude();
    temp = hdc.getTemperature();

    osem.uploadMobileMeasurement(temp, "SensorID", lat, lon)
}
```

# Lauschangriff

In dieser Station wollen wir lernen, wie wir das Mikrofon mit dem Arduino nutzen können.

## Materialien

- Mic-Breakout

## Grundlagen

Das Mikrofon (Mic-Breakout) benötigt eine Betriebsspannung von 2.7V-5.5V und ist in der Lage Geräusche zwischen 58dB und 110dB wahrzunehmen.

## Aufbau

Das Mikrofon wird mit einem JST-Adapterkabel mit der senseBox MCU verbunden. Dazu wird das JST-Adapterkabel mit dem Steckplatz Digital A verbunden. Es verfügt über drei Pins(GND, VCC und OUT). Das schwarze Kabel wird mit GND verbunden, das rote mit VCC und das grüne mit OUT. Hierbei stellen wir mit dem schwarzen Kabel einen Minuspol, mit dem roten Kabel die Stromversorgung und mit dem grünen Kabel eine Datenübertragung zum Port 1 her. Steckt den Schaltkries wie ihr ihn unten in der Grafik seht.



## Programmierung

Eine Variable in der die Werte des Mikrofons gespeichert werden sowie der digitale Port an der das Mikrofon angeschlossen ist, müssen definiert werden.

```
int micValue = 0;
int mic = 1; // Abhängig von dem Port an dem ihr das Mikrofon angeschlossen habt
```

Nun muss die serielle Ausgabe initialisiert werden und dem Pin `mic` der Modus `INPUT` zugeordnet werden. Dies geschieht in der `setup()` -Funktion.

### ▼ `setup()` Funktion

```
void setup(){
    Serial.begin(9600);
    pinMode(mic, INPUT);
}
```

In der `loop()` -Funktion kann nun der gemessene Wert über den seriellen Monitor ausgegeben werden.

#### ▼ loop() Funktion

```
void loop(){
    micValue = analogRead(mic);
    Serial.println(micValue);
}
```

In einem leisen Raum wird der ausgegebene Wert um den Wert 510 schwanken. Bei lauten Geräuschen können auch negative Werte zurückgegeben werden. Um die Lesbarkeit der erhaltenen Werte zu verbessern können wir einige zusätzliche Berechnungen in einer extra Methode ausführen.

#### ▼ Funktion zur Lesbarkeit

```
long getMicVal(){
    int period = 3; // mittelt drei Werte um 'Ausreißer' abzufangen
    int correction_value = 510;
    for(int i = 0; i < period; i++){
        // berechnet den Absolutbetrag des Wertes um negative Ausschläge abzufangen
        micValue = micValue + abs(analogRead(mic)-correction_value);
        delay(5);
    }
    micVal = constrain(abs(micValue/period),1,500);
    return(micValue);
}
```

Diese Funktion kann nun anstelle von `analogRead(mic)` in der `loop()` -Funktion verwendet werden.

#### ▼ Aktualisierte loop() Funktion

```
void loop(){
    micValue = getMicVal(mic);
    Serial.println(micValue);
}
```

Jetzt könnt ihr ausprobieren welche Geräusche welche Ausschläge verursachen:

- Wie stark ist der Ausschlag bei Gesprächen ?
- Was passiert wenn du den etwas lautes vor das Mikrofon hältst ?
- Und was, wenn du hinein pustest ?

## Die ersten Töne - Nutzung eines Summers

Bis jetzt ist unsere senseBox noch recht schweigsam, aber das wollen wir in dieser Station ändern. In einem ersten Schritt wollen wir den Summer zum Piepen bringen. In einem zweiten Schritt wollen wir die Lautstärke des Summers verändern und in einem dritten Schritt soll der Summer eine kurze Melodie abspielen. Während die ersten beiden Ziele recht schnell erreicht werden, ist der dritte Teil ein kleines bisschen kniffeliger.

## Materialien

- Summer
- Potentiometer

## Grundlagen

### ▼ Summer

Ein Summer oder Piezo ist ein Bauteil, welches elektrische Signale in Töne umwandelt. Die Lautstärke beträgt bis zu 80dB. Der Summer hat zwei Pins, mit denen er auf das Steckboard gesteckt werden kann. Die Betriebsspannung des Summers liegt zwischen 1V und 12V, wobei er bis zu 19mA verbraucht. Wie bei der LED kann der Strom nur in eine Richtung fließen. Der kürzer Pin muss mit Ground (GND) verbunden werden und der längere mit einer Spannungsquelle.

### ▼ Potentiometer

Das Potentiometer ist ein elektrisches Bauelement dessen Widerstandswert sich stufenlos einstellen lässt. Über einen Widerstandskörper wird der sogenannte Schleifer bewegt. Die Position des Schleifers bestimmten den Widerstand. Gewöhnlich hat ein Potentiometer drei Anschlüsse: Zwei für den Widerstand und einen dritten für den Abgriff. Unser Potentiometer hat einen maximalen Widerstand von 10k Ohm.

Achtung: Kleine Potentiometer sind nur für einen relativ geringen Stromfluss konstruiert. Für die elektrischen Bauteile in der senseBox reicht das Potentiometer aus, doch wenn du Bauteile mit einer größeren Stromaufnahme anschließen möchtest (z.B. einen Servomotor), benötigst du ein größeres Potentiometer.

## Aufbau

### Schritt 1

Der Summer benötigt einen Minus und einen Pluspol. Stecke den Schaltkreis so wie auf der Abbildung zu sehen.

Der längere Pin am Summer ist der Pluspol(rotes Kabel verbinden) und der kurze Pin der Minuspol(schwarzes Kabel verbinden)



*Verkabelung des Summers*

Wenn ihr den Schaltkreis wie in der Grafik steckt und den Arduino mit dem Netzteil verbindet, sollte der Summer einen lauten Dauerton erzeugen. Damit haben wir unsere erste Aufgabe bereits erledigt.

## Schritt 2

Nun wollen wir ein weiteres Bauteil in unseren Schaltkreis integrieren, mit dessen Hilfe sich die Lautstärke des Summers verändern lässt. Wie in älteren Radios wollen wir dazu ein Potentiometer verwenden.

Verbinde dazu den 5V Ausgang des Arduinos mit dem Potentiometer, und dieses mit dem längeren Pin des Summers. Nun musst du den kurzen Pin des Summers noch mit GND verbinden und schon kannst du über das Potentiometer die Lautstärke verändern.



*Verkabelung des Potentiometers*

Tipp: Falls ihr ein paar Informationen zur Funktionsweise eines Potentiometers lesen möchtet, schaut euch den Eintrag im Glossar an.

Idee: Falls ihr nicht jedes Mal das Kabel ziehen wollt um den Summer auszuschalten, könnt ihr noch einen Drucktaster einbauen.

## Schritt 3

Ein einzelner durchgehender Ton ist nicht besonders spannend; unser Summer kann mehr. Um dem Summer verschiedene Töne zu entlocken, müssen wir Pulsweiten ausgeben. Für nähere Information zu Pulsweitenmodulation (PWM) kannst du hier mehr erfahren. Für dieses Projekt ist das aber nicht unbedingt nötig.

Ein Summer setzt jede Pulsweite in einen spezifischen Ton um. Wir wollen unser Programm so schreiben, dass eine Note (Tonleiter: c, d, e, f, g, a, h, c) für eine Pulsweite steht. Dazu benutzen wird das Konstrukt #define wie folgt:

### ▼ Definieren der Noten

```
#define h      4064    // 246 Hz
```

```
#define c    3830 // 261 Hz
#define d    3400 // 294 Hz
#define e    3038 // 329 Hz
#define f    2864 // 349 Hz
#define g    2550 // 392 Hz
#define a    2272 // 440 Hz
#define h    2028 // 493 Hz
#define C    1912 // 523 Hz
#define E    1518 // 659 Hz
#define F    1432 // 698 Hz
#define R    0     // Definiere eine Note als Ersatz für eine Pause
```

Nun benötigen wir einige Variablen um das spätere Abspielverhalten des Arduinos zu steuern. Die Werte könnt ihr für den Anfang so übernehmen. Wenn ihr später ein fertiges Programm habt, dann könnt ihr verschiedene Werte einsetzen und prüfen wie sich das auf die Melodie auswirkt:

#### ▼ Definieren der globalen Variablen und setup()-Funktion

```
// Geschwindigkeit
long tempo = 26000;
// Länge der Pausen zwischen den Noten
int pause = 1000;
// Variable für die Schleife um die Rest-Länge zu erhöhen
int rest_count = 50;
```

Zusätzlich benötigen wir noch einige globale Variablen welche von den Abspielfunktionen intern genutzt werden, und definieren im setup unseren Ausgangs-Pin:

```
int sound = 0;
int beat = 0;
long duration = 0 ;
int speakerOut = 1; // Abhängig von dem Port an dem ihr den Summer angeschlossen habt

void setup(){
    pinMode(speakerOut,OUTPUT);
}
```

Jetzt könne wir unsere Melodie in ein Array schreiben. Ein weiteres Array beats definiert, wie lange die entsprechende Note in melody gespielt werden soll:

#### ▼ Definieren der Melodie

```
int melody[] = { g, e, R, R, R, e, f, g, E, E, C }; //example melody
int beats[] = { 8, 8, 8, 8, 8, 8, 8, 16, 16, 32 };
```

Natürlich dürft ihr hier später eure eigene Melodie einfügen. Dazu könnt ihr im Internet einmal die Noten eures Lieblingssongs suchen (siehe unten).

Wir schreiben uns eine Hilfsmethode, welche genau einen Ton unserer Melodie abspielt. Dazu überprüft sie in der ersten if-Anweisung ob es sich um eine Note oder eine Pause handelt. Falls es sich um eine Note handelt, wird die Note in einer Schleife für duration Millisekunden gespielt:

#### ▼ Definieren der Hilfsmethode Nr.1

```
void playTone() {
```

```
long elapsed_time = 0;
if (tone > 0) { // if this isn't a Rest beat
    while (elapsed_time < duration) {

        digitalWrite(speakerOut, HIGH);
        delayMicroseconds(tone / 2);

        // DOWN
        digitalWrite(speakerOut, LOW);
        delayMicroseconds(tone / 2);

        // Keep track of how long we pulsed
        elapsed_time += (tone);
    }
} else { // Rest beat;
    for (int j = 0; j < rest_count; j++) {
        delayMicroseconds(duration/2);
    }
}
}
```

Nachdem wir jetzt einen Ton abspielen können, soll eine weitere Hilfsmethode die gesamte Melodie abspielen. Dazu geht eine `for`-Schleife unser Array `melody` durch und ruft für jeden Eintrag die Hilfsfunktion `playTone()` auf, die wir weiter oben definiert haben. Zusätzlich wird nach jeder Note eine kurze Pause eingefügt.

#### ▼ Definieren der Hilfsmethode Nr.2

```
int MAX_COUNT = sizeof(melody) / 2; // number of tones

void playMelody(){
    for (int i=0; i<MAX_COUNT; i++) {
        tone = melody[i];
        beat = beats[i];

        duration = beat * tempo; // Set up timing

        playTone();

        delayMicroseconds(pause);
    }
}
```

Es fehlt uns nur noch die Hauptschleife, welche den Ablauf des Programms steuert:

#### ▼ loop()-Funktion

```
void loop(){
    playMelody();
}
```

Idee: Falls ihr für eure Melodie höhere oder tiefere Töne benötigt, so könnt ihr diese wie im Beispiel oben definieren. Wie viel Hertz ein Ton hat könnt ihr hier nachschauen. Diesen Wert müsst ihr dann mit einem antiproportionalen Dreisatz umrechnen.

Achtung: Zwei Variablen im Programm dürfen nicht den gleichen Namen haben!

# Mehrfarbige LED

In dieser Station wollen wir lernen, wie man eine mehrfarbige LED verwendet.

## Materialien

- RGB-LED
- 1x  $470\Omega$  Widerstand
- JST-Adapterkabel

## Aufbau

Um die mehrfarbige LED mit der senseBoxMCU zu verbinden stecke die Pins wie in der Abbildung unten.



Verkabelung der mehrfarbigen LED

## Programmierung

Nun kannst du mit dem Schreiben des Programms beginnen, indem du die NeoPixel Bibliothek einbindest. Anschließend solltest du eine Variable erstellen, in der du speicherst, an welchem Pin deine LED angeschlossen wird. Dann erstellst du ein Objekt `rgb_led` über das du die LED später steuern kannst.

### ▼ Definieren der globalen Variablen

```
#include <Adafruit_NeoPixel.h>
int pin = 9;

Adafruit_NeoPixel rgb_led = Adafruit_NeoPixel(1, pin, NEO_GRB + NEO_KHZ800);
```

Nun muss die LED in der `setup()` -Methode gestartet werden.

### ▼ setup()-Funktion

```
void setup() {
    rgb_led.begin();
}
```

Die Farbe der LED lässt sich über den Befehl `rgb_led.Color(x, y, z)` bestimmen. Jede der Variablen `x`, `y` und `z` kann man durch eine Zahl zwischen 0 und 255 ersetzen, wobei `x` für den Rotanteil, `y` für den Grünanteil und `z` für den Blauanteil steht. Anschließend muss die definierte Farbe an die LED übergeben werden. Dies geschieht über den Befehl `rgb_led.setPixelColor(0, rgb_led.Color(x,y,z))`. Sichtbar wird die gewählte Farbe mit dem Befehl `rgb_led.show()`. Im folgenden siehst du ein Beispiel, das die drei Grundfarben Rot, Grün und Blau der Reihe nach leuchten lässt.

#### ▼ `loop()`-Funktion

```
void loop() {
    // rot
    rgb_led.setPixelColor(0,rgb_led.Color(255, 0, 0));
    rgb_led.show();
    delay(200);
    // gruen
    rgb_led.setPixelColor(0,rgb_led.Color(0, 255, 0));
    rgb_led.show();
    delay(200);
    // blue
    rgb_led.setPixelColor(0,rgb_led.Color(0, 0, 255));
    rgb_led.show();
    delay(200);
}
```

# Kaminfeuer

Es soll ein Kaminfeuer simuliert werden. Dazu wird eine rote LED zum flackern gebracht.

## Materialien

- LED
- 1x  $470\Omega$  Widerstand
- JST-Adapterkabel

## Aufbau

Es wird nur die LED angeschlossen. Diese wird am langen Beinchen mit einem  $470\Omega$  Widerstand mit dem digitalen Port 1 verbunden. Das kurze Beinchen wird mit GND verbunden.



Verkabelung der einfachen LED

## Programmierung

### ▼ Definieren der globalen Variable und setup()-Funktion

```
// die LED ist an den digitalen Port 1 angeschlossen
int led = 1;

void setup() {
    // der digitale Port 1 wird als OUTPUT definiert
    // d.h.: es werden Signale herausgeschickt
    pinMode(led, OUTPUT);
}
```

Wir speichern den digitalen Port 13 in einer Variable, damit wir uns nur noch den aussagekräftigen Variablennamen merken müssen und nicht die Portnummer. Das ist vor allem bei mehreren angeschlossenen LEDs hilfreich

### ▼ loop()-Funktion

```
void loop() {
    // generiert Zufallszahl zwischen 0 und 1000 und speichert ihn in randomDelayAn
```

```
int randomDelayAn = random(1000);
digitalWrite(led, HIGH);

// Zufallswert wird eingesetzt, LED bleibt entsprechend lange an (in Millisekunden)
delay(randomDelayAn);

// generiert Zufallswert zwischen 0 und 500 und speichert ihn in randomDelayAus
int randomDelayAus = random(500);
digitalWrite(led, LOW);

// Zufallswert wird eingesetzt, LED bleibt entsprechend lange aus (in Millisekunden)
delay(randomDelayAus);
}
```

Die Funktion `random(max)` generiert Zufallszahlen von 0 bis max. Falls man ebenfalls ein Minimum angeben will kann man die Funktion `random(min, max)` benutzen.

## Umweltstation betrieben von einer Solarzelle

Wenn du deine senseBox ohne Steckdose oder Powerbank betreiben möchtest kannst du hierfür auch eine Solarzelle verwenden. Zusätzlich zeigen wir dir wie die LoRa Datenübertragung funktioniert. Somit kannst du auch an entlegenen Orten deine senseBox betreiben ohne auf Strom oder Internet angewiesen zu sein! Am Ende wird dir zudem gezeigt wie du einen Schalter zum Ein- und Ausschalten mit der senseBox verbinden kannst.

## Materialien

- Sensoren
  - Luftdruck(BMP280)
  - Temperatur und Luftfeuchte (HDC1080)
  - UV-Intensität und Beleuchtungsstärke(TSL & VML)
  - Feinstaub (SDS)
- Solarzelle<sup>1</sup>
- 3.7V LiPo Akku
- Adafruit Powerboost 1000C<sup>32</sup>
- LoRa Bee
- Power Switch (An/Aus)

## Aufbau

Achtung: Aufgrund der erhöhten Komplexität der Installation empfehlen wir dieses Projekt ausschließlich fortgeschrittenen Nutzern von Open-Hardware.

Um die senseBox per USB mit Strom zu versorgen und zur gleichen Zeit einen Akku zu benutzen, der von der Solarzelle aufgeladen wird benötigen wir den Adafruit Powerboost 1000C. Um den Stromfluss vom Akku über den Powerboost zur MCU zu regulieren benötigen wir zudem einen Power Switch<sup>4</sup>.

Der muss Switch mit dem Powerboost verbunden werden. Hierfür benötigst du Werkzeug zum Löten. Wie du den Switch mit dem Powerboost verbindest wird dir [hier<sup>5</sup>](#) erklärt. Nun können wir den USB-Eingang am Powerboost verwenden um den Stromfluss zur MCU, mit dem Power Switch, zu regulieren.

In diesem Beispiel wurde ein Loch für den Schalter in das Gehäuse gebohrt. So ist er von außen immer zugänglich und die senseBox kann nach Belieben zum Stromsparen an oder ausgeschaltet werden. Sieh dir die Bildergallerie unten an um zu sehen wie wir das gemacht haben.

- [Aufbau der gesamten Station](#)
- [Power Switch mit Powerboost](#)
- [Schalter im Gehäuse](#)



## Programmierung

Bei der Programmierung muss in diesem Beispiel nichts besonderes beachtet werden. Solange die MCU mit Strom versorgt kann sie die Sketches ausführen die dir in vorherigen Anleitungen schon beigebracht wurden.

---

1. [https://www.amazon.de/Watt-Solarpanel-Volt-Powerbanks-Solarleuchten/dp/B014HWT520/ref=sr\\_1\\_1?ie=UTF8&qid=1532255953&sr=8-1&keywords=villageboom ↵](https://www.amazon.de/Watt-Solarpanel-Volt-Powerbanks-Solarleuchten/dp/B014HWT520/ref=sr_1_1?ie=UTF8&qid=1532255953&sr=8-1&keywords=villageboom)
2. [https://learn.adafruit.com/adafruit-powerboost-1000c-load-share-usb-charge-boost/overview ↵](https://learn.adafruit.com/adafruit-powerboost-1000c-load-share-usb-charge-boost/overview)
3. [https://learn.adafruit.com/adafruit-powerboost-1000c-load-share-usb-charge-boost/overview ↵](https://learn.adafruit.com/adafruit-powerboost-1000c-load-share-usb-charge-boost/overview)
4. [https://www.adafruit.com/product/805 ↵](https://www.adafruit.com/product/805)
5. [https://learn.adafruit.com/adafruit-powerboost-1000c-load-share-usb-charge-boost/assembly#on-slash-off-switch-3-5 ↵](https://learn.adafruit.com/adafruit-powerboost-1000c-load-share-usb-charge-boost/assembly#on-slash-off-switch-3-5)

## Thermometer Hütte für die senseBox

Beim Suchen nach dem idealen Aufstellort der kam die Idee eine altbewährte Wetterhütte zu verwenden, und die Temperatur in 2 Meter Höhe zu messen. Die Thermometer Hütte aus Holz nach Stevenson ist ein geeigneter Strahlungsschutz für den Temperatur/Luftfeuchtigkeit Sensor der senseBox. Lamellierte Gehäusewände und -türe sorgen für eine gute Belüftung und Schutz gegen direkte und reflektierende Sonnenstrahlung. Die Wetterhütte aus Holz wurde für einen optimalen Wärmestrahlungsschutz von innen und aussen weiss lackiert.

Dieses Projekt wurde von einem unserer Community-Member realisiert und dokumentiert! Der Author ist OllyS und ist wie viele andere im [senseBox Forum](#) zu finden. Im Forum findest du zudem Anregungen, neue Ideen oder Diskussionen rund um die senseBox!

### Materialien

- senseBox MCU (Outdoor Case, Netzteil und Kabel, Temperatur & relative Luftfeuchte Sensor,Wifi-Bee)
- Sensoren (Luftdruck & Temperatur, Beleuchtungsstärke & UV-Strahlung, Feinstaub (PM10 & PM2.5)
- Wetterhaus (Bausatz roh oder gestrichen) Bezugsquelle Stiftung zur Palme<sup>1</sup>



### *Bausatz Wetterhaus Palme*

- benötigtes Kleinmaterial
  - Gerätebuchse IP68 4Pol Distrelec:300-47-767 (3)
  - Kabelstecker IP68 4 Pol Distrelec:300-47-727 (3)
  - Datenkabel ungeschirmt 3x 2x0.25mm<sup>2</sup> Distrelec:155-73-712
  - I2C-I2C Kabel Watterott: (3)
  - Nova SDS011 senseBox Anschlusskabel - 350mm Watterott (1)
  - USB-Buchse Panel-Kontakt micro-USB B Distrelec:301-03-704 (1)
  - Druckausgleichselement GR M12x1.5 Distrelec:150-15-938 (1)
  - Kunststoffgehäuse 65x115x40mm hellgrau P Distrelec:300-64-459 (1)
  - Distanzbolzen 20mm 5mm Distrelec: 110-76-305 (4)
  - Senkkopfschrauben M2.5 x 8mm (4)
  - Muttern M2.5 (4)
  - Schrumpfschlauch 2.5 mm

## Aufbau

Die senseBox MCU Home bietet von sich aus schon eine gute Wetterfestigkeit und kann ohne Modifikationen "Outdoor" verwendet werden. Damit der Luftdurchfluss des PM2.5/10 Sensors gewährleistet ist wird eine zusätzliche Bohrung in das Gehäuse angebracht. Um zu verhindern, dass Insekten das Gehäuse entern wird der Lufteinlass und die Bohrung mit einem Stück Insektengitter geschützt.

- [Bohren mit Stufenbohrer](#)
- [Insektenschutz](#)



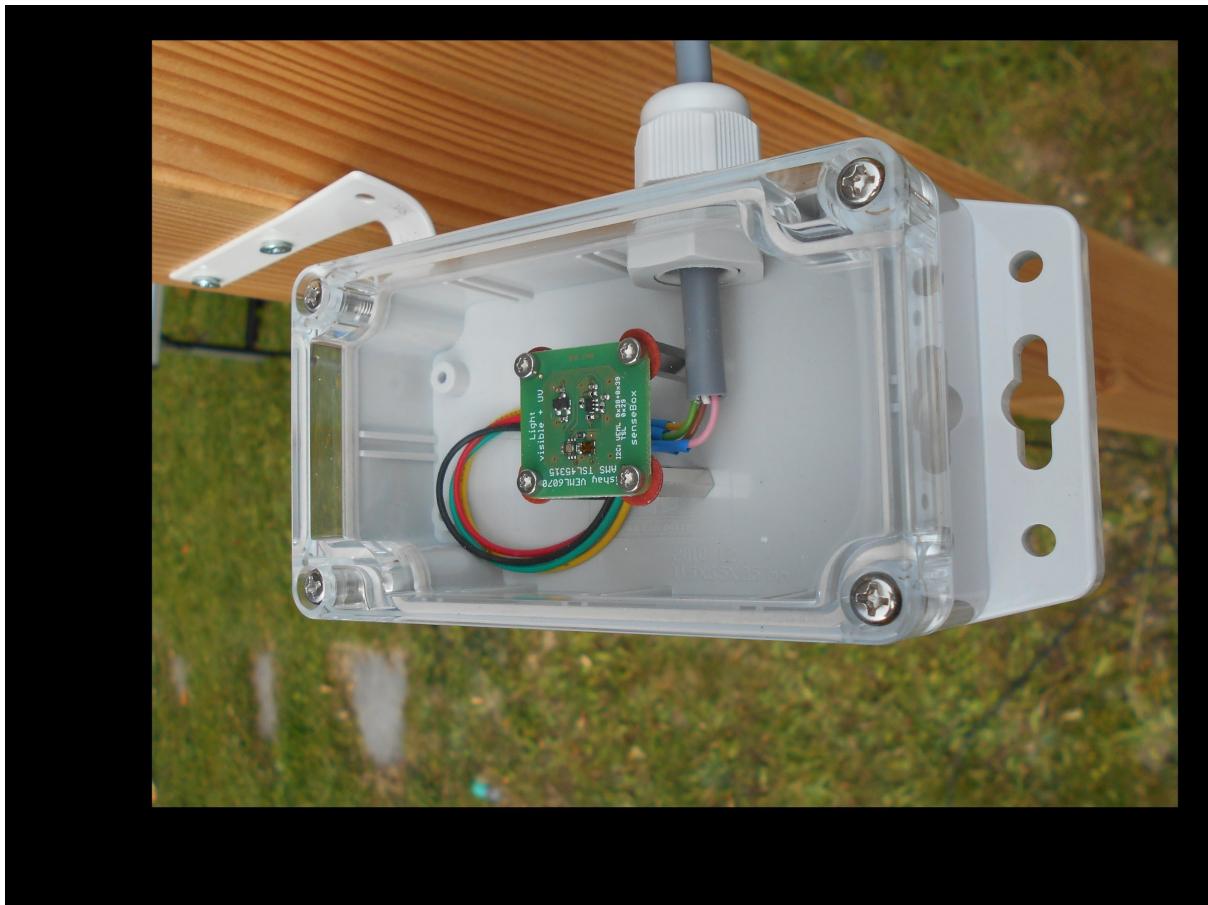
*Bohren*



*InsektenSchutz*

Wegen der Servicefreundlichkeit (Austausch einzelner Komponenten im Betrieb) werden die Boxen der einzelnen Sensoren jeweils mit längeren Kabeln und wasserfesten Steckern versehen. Im MCU Gehäuse wurde ein wasserdichtes Druckausgleichselement eingebaut und der Helligkeits/UV Sensor wurde in einem eigenen Gehäuse untergebracht.

- Helligkeit/UV-Sensor
- Pol Stecker löten
- Modifizierung SDS011
- USB Ausgang am Gehäuse
- Gehäuse
- Druckausgleichselement
- MCU-Platine



Sensor-UV



4 Pol Stecker löten

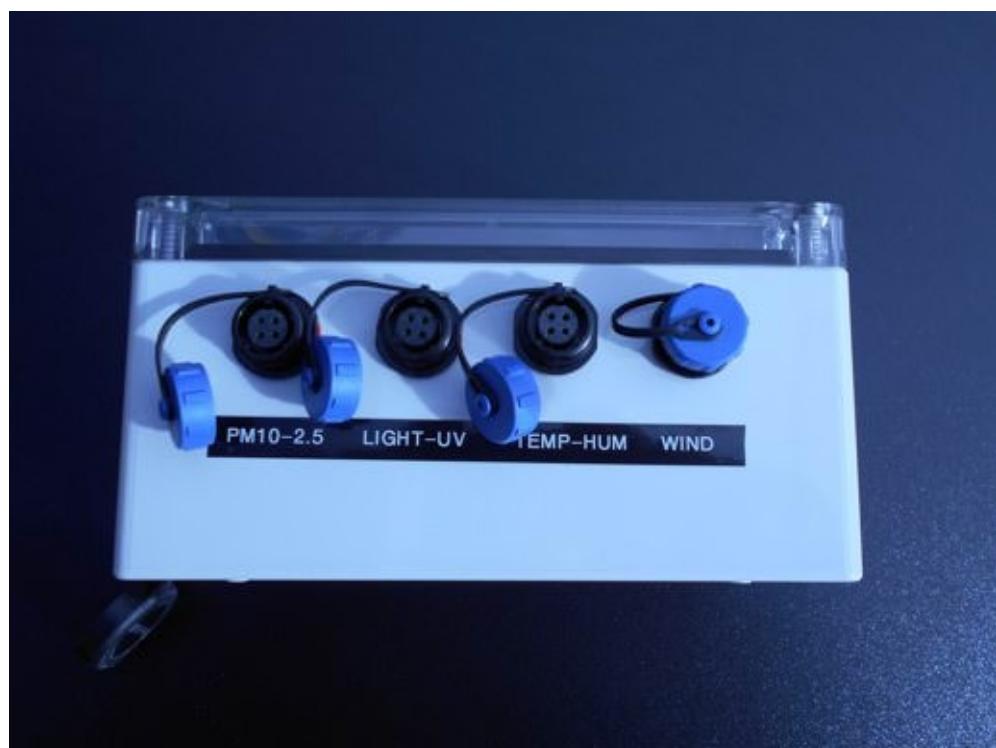


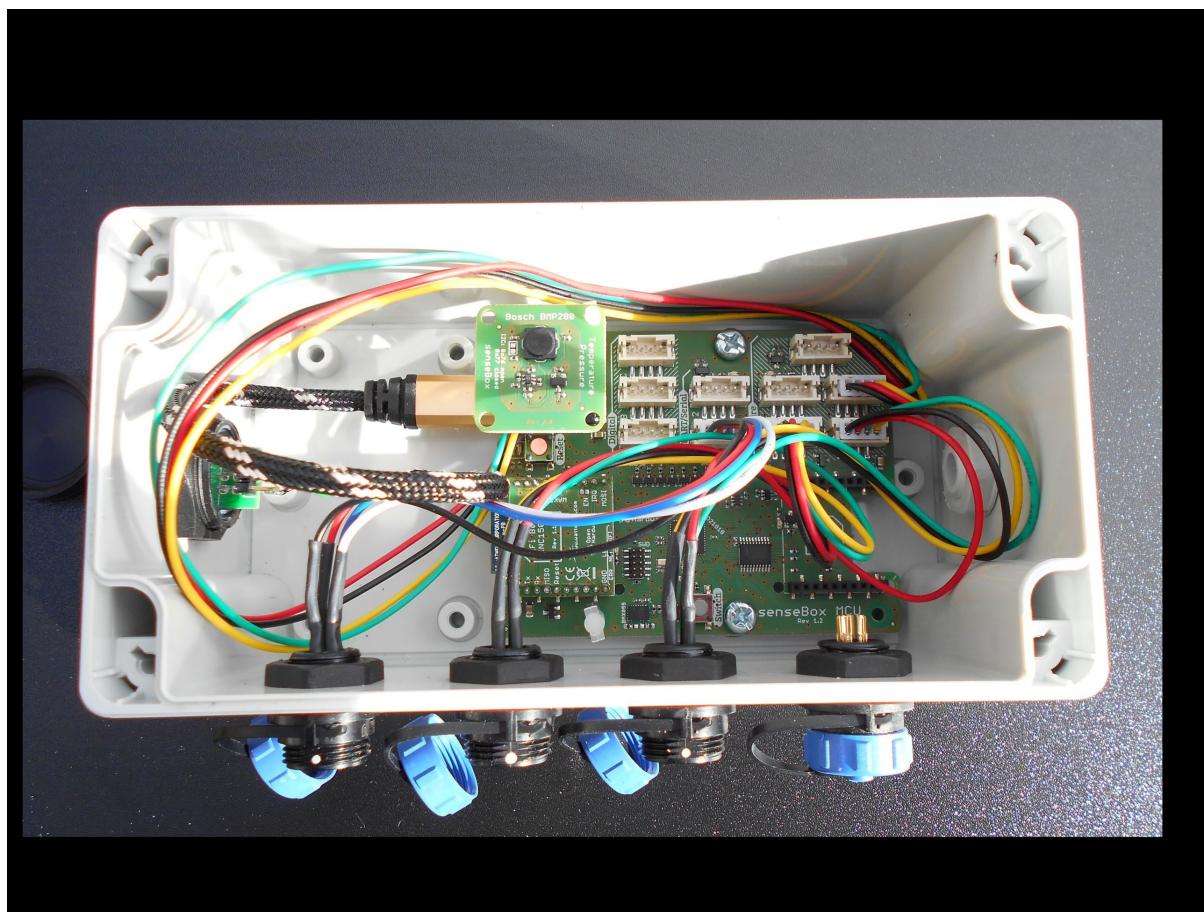
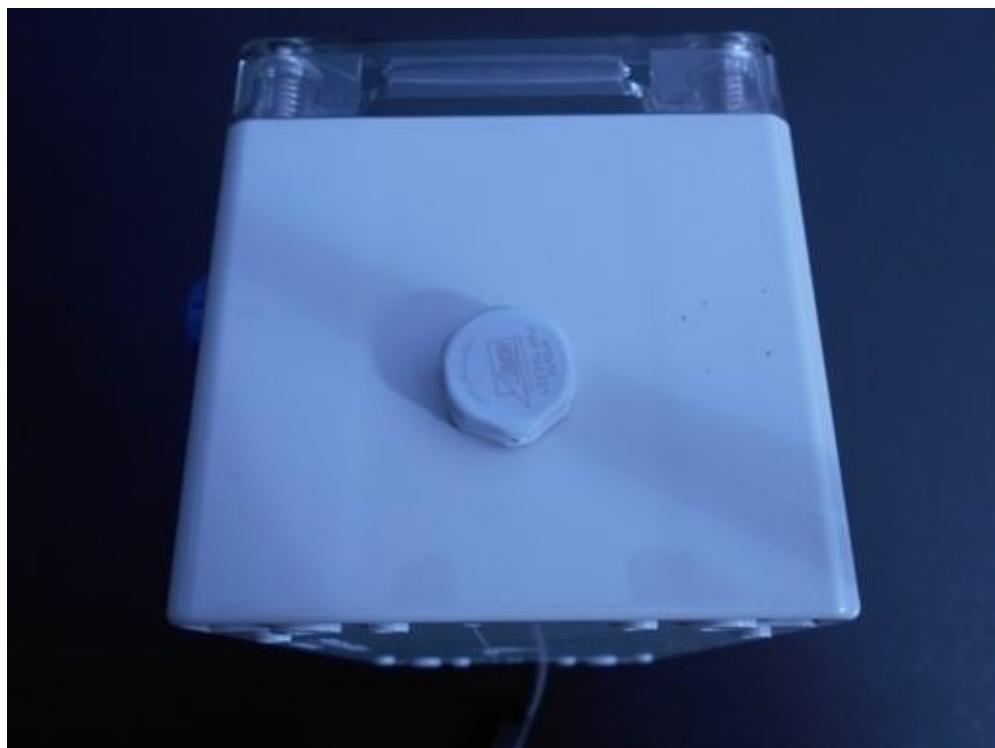
**SB2-PM**

Um Fremdlichteinstrahlung beim SDS011 zu vermeiden wurde der durchsichtige Ansaugschlauch mit Isolierband umwickelt.



Für Stromversorgung und Wartung (Aufspielen Sketches) wurde eine wasserfeste micro-USB Buchse in das MCU-Gehäuse eingebaut.





## Belegung der Buchsen und Stecker

### M16 Panel Mount Micro USB 2.0 - Type B

Hier löten wir ein Micro USB Kabel an, praktisch eine Verlängerung zur MCU.

- Pin 1 = 5 Volt (rot)
- Pin 2 = Data - (weiss)
- Pin 3 = Data + (grün)
- Pin 4 = ID (nicht verwendet)
- Pin 5 = GND (schwarz)
- Pin 6 = Shield (Abschirmung)

### 4 Pol Gerätebuchse IP68 I2C

Hier wird ein fertig konfiguriertes I2C Kabel in der Mitte durchgeschnitten und an die Buchse angelötet, die einzelnen Kontakte werden mit Schrumpfschlauch geschützt. Das andere Ende wird in die MCU Platine an einen I2C Anschluss eingesteckt.

- Pin 1 = GND (Ground) schwarz
- Pin 2 = VCC (+5 Volt) rot
- Pin 3 = SDA (Data) grün
- Pin 4 = CLK (Clock) gelb

### 4 Pol Gerätebuchse IP68 UART/Seriell (PM2.5/PM10 Sensor)

Hier wird das SDS011 senseBox Anschlusskabel in der Mitte getrennt, und das Teil mit dem kleinen Stecker an die Buchse angelötet. Die andere Teil Hälfte des Kabels wird im Feinstaubsensor-Gehäuse an das Datenkabel gelötet (Bild 6 & 7).

Die Pinbelegung der Buchse des Feinstaubensors:

- Pin 1 = GND (Ground) schwarz
- Pin 2 = Tx (Transmit from Host) weiss
- Pin 3 = Rx (Receive from Host) blau
- Pin 4 = VCC (Power 5V) rot

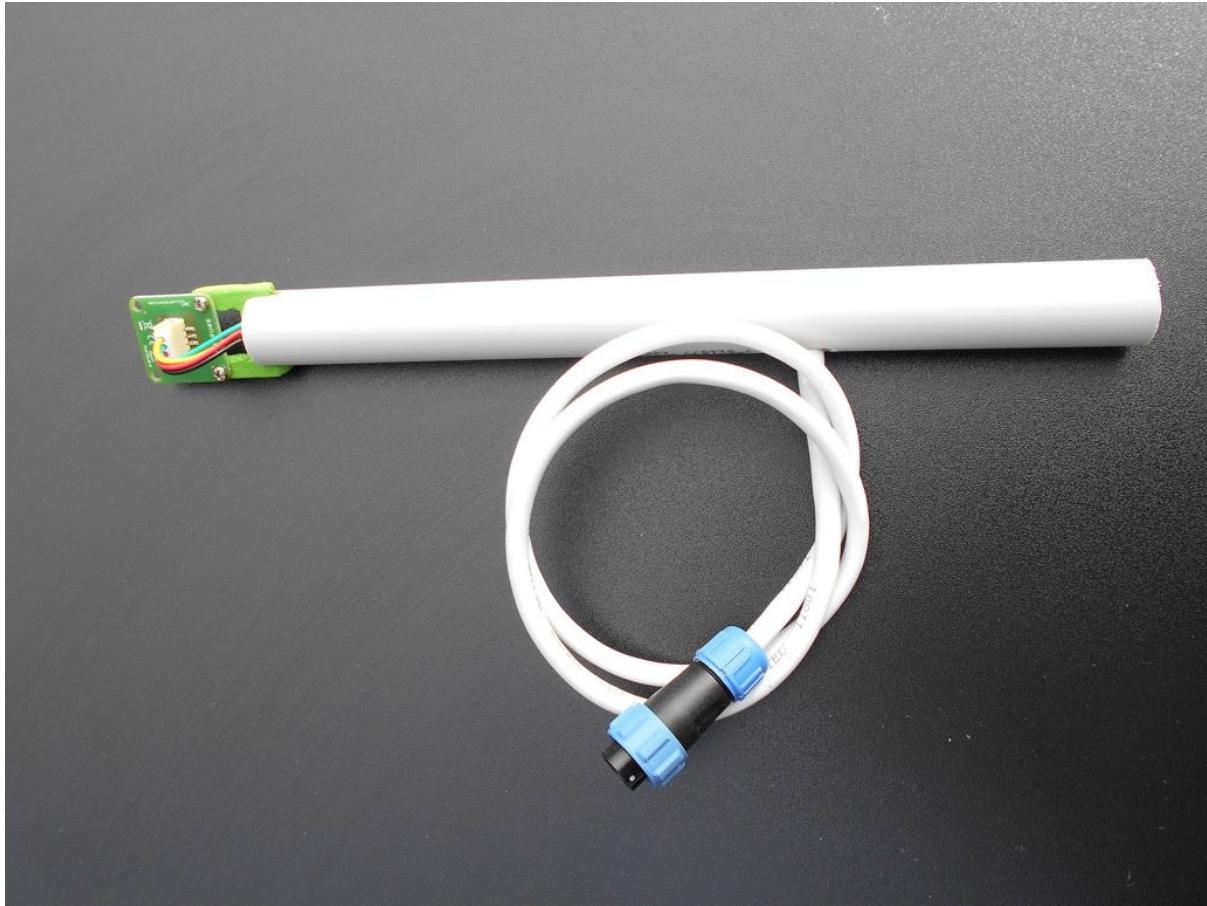
## Zusammenbau

Der Hüttenbausatz von der [Stiftung zur Palme](#)<sup>2</sup> Schreinerei ist eine gute Tischlerarbeit und alles passte problemlos zusammen. Um die Hütte zu grundieren und mit Lackfarbe zu streichen wurden ca. 4 Stunden benötigt. Man kann sie gegen Aufpreis auch fertig gestrichen bestellen.

Die MCU und der Luftqualitätssensor wurden der Einfachheit halber auch in die Hütte gestellt.



Befestigt wurde die Hütte auf einem 11x11 cm Holzpfosten, der Temperatursensor befindet sich auf ca. 2 Meter Höhe über dem Rasen. Befestigt wurde er mit einem Stück Kunststoffrohr und einem Stückchen Balsaholz.



Die einflügelige, verschliessbare Tür ist nach Norden gerichtet, der Helligkeits- und UV-Sensor nach Süden.



Alle Daten werden von der senseBox via WLAN an einen Router im Haus gesendet und von dort aus zur openSenseMap.  
Info: <https://opensensemap.org/info><sup>3</sup>

### Geplante Erweiterungen

- Autonomer Betrieb mit Solarpanel, Akku und LoRa-Bee.
- Messeinheit für Windrichtung, Windgeschwindigkeit und Niederschlagsmenge.

## Literatur

1. Kopp M. (2012): Vergleich verschiedener Wetterhütten zur Messung der 2m-Lufttemperatur in Abhängigkeit von Wind und Strahlung. [Link<sup>4</sup>](#)
2. Deutscher Wetterdienst: Vorschriften und Betriebsunterlagen Nr. 3 (VuB 3) TECHNIKHANDBUCH (THB) für Wettermeldestellen des synoptisch-klimatologischen Mess- und Beobachtungsnetzes<sup>5</sup>

## Weblinks:

1. Verein GLOBE Schweiz: Wetterhaus [https://www.globe-swiss.ch/de/Zyklus3/Angebote\\_Wetter\\_und\\_Atmosphare/Atmosphere/Wetterhaus/](https://www.globe-swiss.ch/de/Zyklus3/Angebote_Wetter_und_Atmosphare/Atmosphere/Wetterhaus/)<sup>6</sup>
1. Das senseBox:Home Buch: <https://sensebox.github.io/books-v2/home/de/index.html><sup>7</sup>
2. openSenseMap: <https://sensebox.github.io/books-v2/osem/><sup>8</sup>
3. senseBox Shop: <https://sensebox.kaufen/product/sensebox-home><sup>9</sup>

## Autor:

OllyS - zu finden im [senseBox Forum](#)<sup>10</sup>

---

1. [https://www.globe-swiss.ch/files/Downloads/902/Download/Wetterhausflyer\\_2017.pdf](https://www.globe-swiss.ch/files/Downloads/902/Download/Wetterhausflyer_2017.pdf) ↵
2. <https://www.palme.ch/pages/schreinerei.htm> "Stiftung zur Palme" ↵
3. <https://opensensemap.org/info> "https://opensensemap.org/info" ↵
4. <https://wikis.fu-berlin.de/pages/viewpage.action?pageId=841581560&preview=%2F841581560%2F841581563%2FH%C3%BCttenvergleichENDVERSION.pdf> ↵
5. [https://www.dwd.de/DE/leistungen/pbfb\\_verlag\\_vub/pdf\\_einzelbaende/vub\\_3\\_thb\\_gesamt\\_pdf.pdf?blob=publicationFile&v=5](https://www.dwd.de/DE/leistungen/pbfb_verlag_vub/pdf_einzelbaende/vub_3_thb_gesamt_pdf.pdf?blob=publicationFile&v=5) ↵
6. [https://www.globe-swiss.ch/de/Zyklus3/Angebote\\_Wetter\\_und\\_Atmosphare/Atmosphere/Wetterhaus/](https://www.globe-swiss.ch/de/Zyklus3/Angebote_Wetter_und_Atmosphare/Atmosphere/Wetterhaus/) ↵
7. <https://sensebox.github.io/books-v2/home/de/index.html> ↵
8. <https://sensebox.github.io/books-v2/osem/> ↵
9. <https://sensebox.kaufen/product/sensebox-home> ↵
10. <https://forum.sensebox.de/> ↵

# Übersicht aller Komponenten

Hier findest du eine Liste aller Sensoren, Bees und anderen Zubehörteile der senseBox. Wir haben für jedes Teil eine eigene Seite erstellt, um es jeweils kurz zu erklären und dir alle nötigen Informationen bereit zu stellen. So kannst du mit deinen Komponenten und der senseBox voll durchzustarten!

Wenn du wissen willst, wofür du ein Teil benutzen kannst klicke einfach auf den Namen.

1. [senseBox MCU](#)
2. [Bees](#)
  - [Wifi-Bee<sup>1</sup>](#)
  - [LAN-Bee<sup>2</sup>](#)
  - [SD-Bee<sup>3</sup>](#)
  - [LoRa-Bee<sup>4</sup>](#)
3. [Sensoren](#)
  - [Temperatur & Luftfeuchte \(HDC1080\)<sup>5</sup>](#)
  - [Luftdruck & Temperatur<sup>6</sup>](#)
  - [Belichtung und UV<sup>7</sup>](#)
  - [Feinstaub<sup>8</sup>](#)
4. [Zubehör](#)
  - [Strahlenschutz<sup>9</sup>](#)
  - [Gehäuse<sup>10</sup>](#)
  - [Netzteil und USB-Kabel<sup>11</sup>](#)
  - [LED-Display<sup>12</sup>](#)
  - [HUB<sup>13</sup>](#)
  - [Micro-SD Karte<sup>14</sup>](#)
  - [GPS<sup>15</sup>](#)

Natürlich gibt es noch viele andere Sensoren, die du mit ein bisschen Tüftelei und Eigeninitiative an deine senseBox anschliessen kannst. Zurzeit können wir dir leider nur Anleitungen für die aufgelisteten Komponenten geben. Falls du Fragen zu anderen Sensoren hast kann dir unser [Forum<sup>1</sup>](#) weiterhelfen. Dort bekommst du nicht nur Hilfe von uns sondern auch von Tüftlern aus der ganzen Welt.

---

<sup>1</sup>. Siehe [5.1.2.1 Wifi-Bee ↵](#)

<sup>2</sup>. Siehe [5.1.2.2 LAN-Bee ↵](#)

<sup>3</sup>. Siehe [5.1.2.3 mSD-Bee ↵](#)

<sup>4</sup>. Siehe [5.1.2.4 LoRa-Bee ↵](#)

<sup>5</sup>. Siehe [5.1.3.1 Temperatur & Luftfeuchte \(HDC1080\) ↵](#)

- | 6. Siehe [5.1.3.2 Luftdruck & Temperatur](#) ↵
- | 7. Siehe [5.1.3.3 Belichtung und UV](#) ↵
- | 8. Siehe [5.1.3.4 Feinstaub](#) ↵
- | 9. Siehe [5.1.4.1 Strahlenschutz](#) ↵
- | 10. Siehe [5.1.4.2 Gehäuse](#) ↵
- | 11. Siehe [5.1.4.3 Netzteil und USB-Kabel](#) ↵
- | 12. Siehe [5.1.4.4 LED-Display](#) ↵
- | 13. Siehe [5.1.4.5 Expander](#) ↵
- | 14. Siehe [5.1.4.6 Micro-SD Karte](#) ↵
- | 15. Siehe [5.1.4.7 GPS](#) ↵
- | 1. <https://forum.sensebox.de/> ↵

## senseBox MCU

Der senseBox-Microcontroller ist speziell für die Bedürfnisse der senseBox entwickelt. Deshalb hat der Microcontroller insbesondere drei Eigenschaften: er ist schnell, energiesparend und hat einen großen Programmspeicher.

Auch ohne Vorkenntnisse kann der Microcontroller ganz einfach mit der Übertragung eines vorgefertigten Sketches in 10 Minuten "programmiert" werden. Für fortgeschrittene Programmierer kann wie gewohnt die Arduino IDE eingesetzt werden um neue und individuelle Projekte durchzuführen.



Die senseBox MCU

## Technische Spezifikationen

### Prozessor

Der Prozessor basiert auf dem ARM Cortex-M0+ Prozessor aus der SAM D21 Familie von Microchip.

### Schnittstellen

Sensoren und Aktoren werden über die bewährten Schnittstellen wie I2C, UART und digitale I/Os mit einem robusten JST-Steckersystem angesprochen (5V tolerant).

### Datenübertragung

Über die beiden XBee kompatiblen Sockel werden UART oder SPI Module angeboten. Wahlweise kann die Datenübertragung dadurch per WLAN, LAN, oder LoRa in Echtzeit durchgeführt werden, oder auf einer Mikro-SD Karte abgespeichert werden.

### Features

1. Crypto Authentication für OTA (Over the Air)
2. Firmware-Upgrades durch den ATECC608A von Microchip
3. Integrierter BMX055 Sensor von Bosch, womit sich die Beschleunigung, die Neigung und die Orientierung zum Erdmagnetfeld bestimmen lassen
4. USB CDC+MSC Bootloader (Arduino compatible)
5. Schnittstellen: I2C = 5 (erweiterbar mit I2C Hub) | 2 UART | 6 analoge und digitale IOs



## Bees

Es gibt verschiedene Bees mit der die senseBox Daten speichern, oder diese ins Internet übermitteln kann. Im Moment kann man zwischen vier verschiedenen Bees wählen.

---

## Übersicht

Klicke eines der Bees an und schau dir Informationen, Tipps und technische Details an:

- Wifi-Bee<sup>1</sup>
  - LAN-Bee<sup>2</sup>
  - SD-Bee<sup>3</sup>
  - LoRa-Bee<sup>4</sup>
- 

<sup>1</sup>. Siehe [5.1.2.1 Wifi-Bee](#) ↵

<sup>2</sup>. Siehe [5.1.2.2 LAN-Bee](#) ↵

<sup>3</sup>. Siehe [5.1.2.3 mSD-Bee](#) ↵

<sup>4</sup>. Siehe [5.1.2.4 LoRa-Bee](#) ↵

## Wifi-Bee

Verbindungsstück um die senseBox mit dem Internet zu verbinden. Die Daten der senseBox werden per WLAN (Wifi) in das bestehende Netzwerk übertragen. Das Wifi-Bee basiert auf dem ATWINC1500 Mikrochip von Atmel welcher einen sehr geringen Energieverbrauch und eine hohe Reichweite hat.

Bei einigen unserer WiFi Bees vom Typ WINC1500 kann es vorkommen, dass eine veraltete Firmware (Version 19.4.4) installiert ist. Das kann zu Übertragungsproblemen führen. Sollten diese Probleme bei dir auftreten, schaue dir [diese Hilfeseite](#) an, um die Firmware zu aktualisieren.



*Wifi-Bee*

## Technische Informationen

- "Plug-in-and-Go" senseBox kompatibel
- Single-band 2.4GHz b/g/n
- Operating voltage: 3.0V to 4.2V
- Serial host interface: SPI
- Security protocols supported: WPA/WPA2 Personal, TLS, SSL
- Network services: DHCP, DNS, TCP/IP (IPv4), UDP, HTTP, HTTPS
- Bezeichnung: WINC1500
- Maße: 24mm x 25mm x 9mm
- Gewicht: 3,5 g

## Lan-Bee

Verbindungsstück um die senseBox mit dem Internet zu verbinden. Die Daten der senseBox werden per LAN-Kabel an direkt an einen Router übertragen. Das LAN-Bee basiert auf dem W5500 Mikrochip von Wiznet welcher eine hohe Ethernet Datenübertragungsrate ermöglicht.



*LAN Bee*

## Technische Informationen

- "plug-in-and-go" senseBox kompatibel
- 3.3V Betriebsspannung mit 5V I/O signal Toleranz
- Bezeichnung: W5500
- Maße: 46mm x 25mm x 12mm
- Gewicht: 9,2 g

## Hinweise und Tipps

Das LAN-Bee wird ohne LAN-Kabel geliefert. Es eignen sich aber insbesondere flache LAN-Kabel, welche man einfach durch Fenster und Türen nach außen legen kann. Insbesondere in Kombination mit Power over Ethernet (PoE) kann die senseBox dann einfach betrieben werden.

## mSD-Bee

Mit dem mSD-Bee können die Daten der senseBox auf einer SD-Karte gespeichert werden. So kann man messen, auch wenn keine Internetanschluss in der Nähe der senseBox ist.



*microSD-Bee*

## Technische Informationen

- "Plug-in-and-Go" senseBox kompatibel
- Port für miniSD-Karte
- Bezeichnung: mSD-Bee
- Maße: 24mm x 21mm x 9mm
- Gewicht: 2,4 g

## Hinweise

Achtung: Das SD-Bee wird ohne SD-Karte geliefert, falls ihr im senseBox-Shop bestellt.

## Beispiel

Im folgenden Beispiel zeigen wir euch wie ihr eure Daten auf einer SD-Karte speichern könnt.

### ▼ Deklarierung der Objekte und setup()

```
// Einladen der Bibliotheken
#include <SPI.h>
#include <SD.h>
#include "SenseBoxMCU.h"

HDC1080 hdc;

void setup(){

    // Starten der SD-Bee
    SD.begin(28);
    // Öffnen der Datei auf der SD-Karte
    dataFileSenseBox = SD.open("SenseBox.txt", FILE_WRITE);
    dataFileSenseBox.close();
```

```
    hdc.begin();
};
```

In der `loop()` -Funktion speichern wir nun unsere Messwerte in der eben erstellten Datei.

▼ **loop()**

```
void loop(){
    dataFileSenseBox.println(hdc.getTemperature());
};
```

## LoRa-Bee

Verwendet die LoRa-Schnittstelle um Daten ins Internet zu übertragen. Neu ist das LoRa WAN-XBee-Modul, mit dem eine stromsparende und kostenlose Möglichkeit der Datenübertragung ins Internet über den LoRa-Funk-Standard ermöglicht wird. Dafür werden bestehende LoRa-Netzwerke, wie zum Beispiel TheThingsNetwork genutzt um Daten zu übertragen. Die hierzu benötigte Infrastruktur wird bei TheThingsNetwork von der Community bereit gestellt, und ist in immer mehr Regionen verfügbar.



*Lora Bee*

## Technische Informationen

- HopeRF RFM95W/RFM96W LoRa Transceiver
- LoRa-Bee 868 / 915 MHz nutzt RFM95W (SX1276 kompatibel)
- LoRa-Bee 433 / 470 MHz nutzt RFM96W (SX1276 kompatibel)
- SPI interface
- Bezeichnung: RFN9xW
- Maße: 46mm x 25mm x 12mm
- Gewicht: 1,1 g

## Hinweise

Bitte prüfe bevor du dir eine senseBox mit LoRa Bee holst, ob dein Gebiet bereits von LoRa erschlossen ist:

<https://www.thethingsnetwork.org/community#list-communities-map>

Achtung: Aufgrund der erhöhten Komplexität der Installation empfehlen wir das LoRa-Modul ausschließlich fortgeschrittenen Nutzern von Open-Hardware

## Upload über LoRaWAN

Es ist möglich Sensordaten per LoRaWAN™ durch das [TheThingsNetwork<sup>21</sup>](#) (TTN) auf die openSenseMap zu laden. LoRa ist ein zunehmend Verbreitung findender Funkstandard, welcher ähnlich wie WiFi digitale Datenübertragung in einem IP-Netzwerk erlaubt, jedoch deutlich andere Features bietet:

- Datendurchsatz: 300 - 3000 Bit/s
- Reichweite: bis zu 15km

TTN ist eins von mehreren Projekten, welches die zur Funk-Hardware zugehörige Infrastruktur für das IP-Netzwerk implementiert, wodurch registrierte Geräte mit dem Internet verbunden werden können.

Nutzer können Gateways sowie Nodes zu dem Netzwerk hinzufügen.

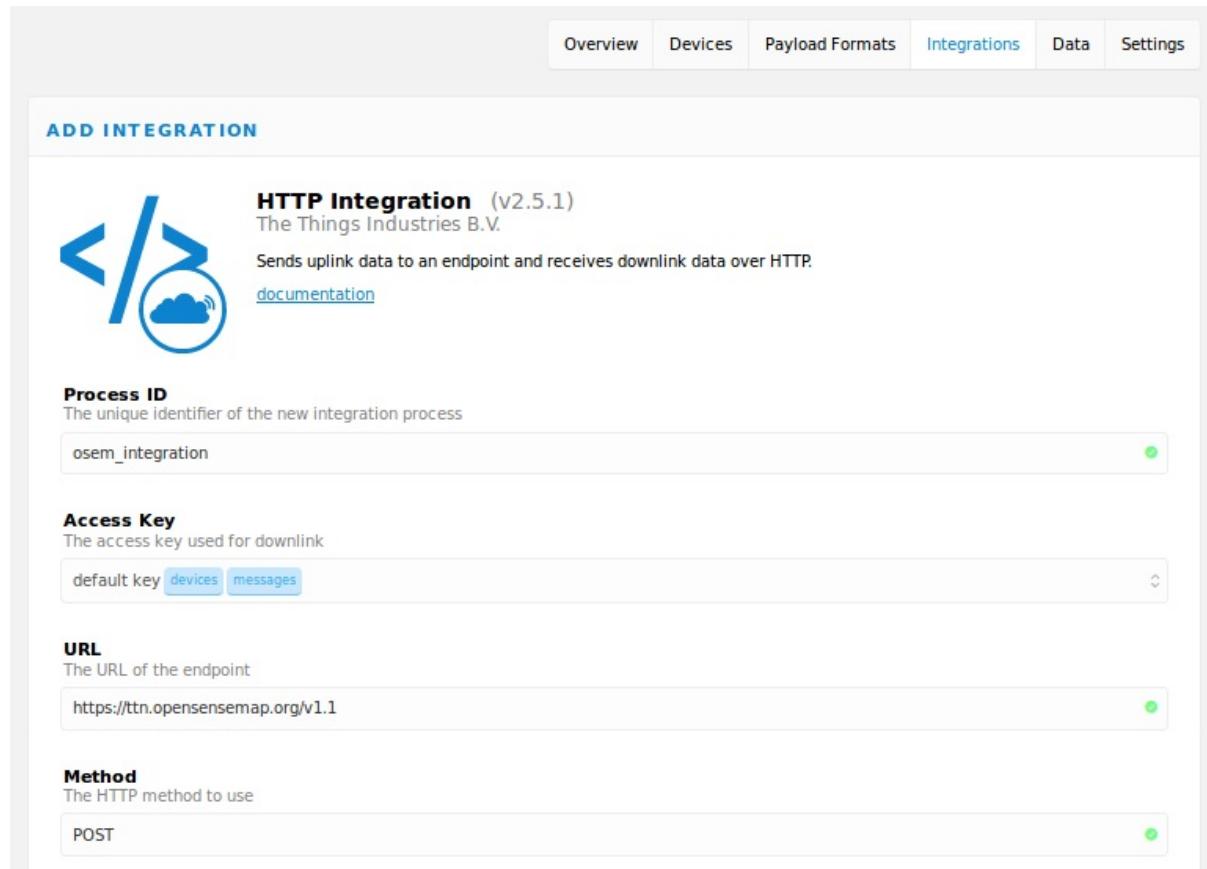
## TTN openSenseMap Integration

Die openSenseMap bietet eine direkte Integration in das TTN Netzwerk, was die Konfiguration stark vereinfacht. Hierfür musst du einen Account [TheThingsNetwork](#) erstellen.

### Registrierung in TTN Console

Um ein Gerät in das TTN einzubinden, muss für dieses zunächst unter [thethingsnetwork.org](#)<sup>3</sup> eine Application und ein Device registriert werden. Hierbei erhält man eine `app_id` und eine `dev_id`.

Für die registrierte Application muss die HTTP Integration unter [https://console.thethingsnetwork.org/applications/DEINE\\_APPID/integrations/create/http-ttn](https://console.thethingsnetwork.org/applications/DEINE_APPID/integrations/create/http-ttn) aktiviert werden. Diese muss konfiguriert werden, dass sie die Nachrichten von Devices per `POST` an <https://ttn.opensensememap.org/v1.1> weiterleitet. Das Authorization-Feld kann leer bleiben!



The screenshot shows the 'Add Integration' page for the 'HTTP Integration' provider. The top navigation bar includes 'Overview', 'Devices', 'Payload Formats', 'Integrations' (which is active and highlighted in blue), 'Data', and 'Settings'. The main section displays the 'HTTP Integration' provider details: version v2.5.1 by The Things Industries B.V., a description stating it 'Sends uplink data to an endpoint and receives downlink data over HTTP.', and a link to the 'documentation'. Below this, configuration fields are shown:

- Process ID:** A unique identifier for the integration process, currently set to 'osem\_integration'.
- Access Key:** The access key used for downlink, currently set to 'default key' with options for 'devices' and 'messages'.
- URL:** The URL of the endpoint, currently set to 'https://ttn.opensensememap.org/v1.1'.
- Method:** The HTTP method to use, currently set to 'POST'.

*HTTP Integration zur openSenseMap*

Für die Datenübertragung zur openSenseMap müssen die `app_id` und `dev_id` bei der Registrierung auf der openSenseMap in der TTN-Konfiguration angegeben werden. Darüber hinaus muss ein passendes Decoding-Profil konfiguriert werden, welches bestimmt wie die - wegen der geringen Bandbreite als rohe Bytes übertragenen - Daten als Messungen interpretiert werden sollen.

## Erweitert

TheThingsNetwork - TTN

Die openSenseMap bietet eine Integration mit TheThingsNetwork an. Für eine Erklärung der Parameter siehe [hier](#)

TheThingsNetwork

Dekodierungs-Profil

senseBox:home

TTN Application-ID

my-osem-app

TTN Device-ID

my-osem-device

Dekodierungsoptionen

[]

Port

### Registrierung auf der openSenseMap

Optional kann im Feld `port` noch der Port angegeben werden, auf welchem der Sender seine Daten an das TTN schickt. So lassen sich die selbe `app_id` und `dev_id` für mehrere Sensorstationen verwenden.

## Arduino Sketch

So könnte ein Arduino Sketch aussehen, mit dem du Daten über das TTN-Netzwerk an die openSenseMap senden kannst. Mit diesem Sketch werden die Phänomene:Lufttemperatur, Luftfeuchte, PM10, PM2.5, UV-Intensität, Beleuchtungsstärke und Luftdruck gemessen.

Wichtig: Du musst deine eben erstellte Application-EUI, Device-EUI und den App-Key in den Sketch einfügen. Dies machst du in den ersten Zeilen des Programmcode wo 'INSERT YOUR ID HERE' steht.

Achte darauf, dass auf der TTN-Homepage du für die Device-EUI und die Application-EUI das lsb-Format und für den App-Key das msb-Format ausgewählt hast!



### *Ausgewählte ID's und Keys*

#### ▼ Deklarieren der globalen Variablen und Deffinierung der Sensoren

```
/*
senseBox:home - Citizen Sensingplatform

Version: lorav2.0.0

Date: 2018-09-11

Homepage: https://www.sensebox.de https://www.opensensemap.org

Author: Reedu GmbH & Co. KG

Note: Sketch for senseBox:home LoRa MCU Edition

Model: homeV2lora

Email: support@sensebox.de

Code is in the public domain.

https://github.com/sensebox/node-sketch-templater

*/
#include <LoraMessage.h>

#include <lmic.h>

#include <hal/hal.h>

#include <SPI.h>

#include <senseBoxIO.h>

#include <Adafruit_Sensor.h>

#include <Adafruit_HDC1000.h>

#include <Adafruit_BMP280.h>

#include <Makerblog_TSL45315.h>

#include <VEML6070.h>

#include <SDS011-select-serial.h>

// Uncomment the next line to get debugging messages printed on the Serial port

// Do not leave this enabled for long time use
```

```
#define ENABLE_DEBUG

#ifndef ENABLE_DEBUG

#define DEBUG(str) Serial.println(str)

#else

#define DEBUG(str)

#endif

// Connected sensors

// Temperatur

#define HDC1080_CONNECTED

// rel. Luftfeuchte

#define HDC1080_CONNECTED

// Luftdruck

#define BMP280_CONNECTED

// Beleuchtungsstärke

#define TSL45315_CONNECTED

// UV-Intensität

#define VEML6070_CONNECTED

// PM10

#define SDS011_CONNECTED

// Number of serial port the SDS011 is connected to. Either Serial1 or Serial2

#ifndef SDS011_CONNECTED

#define SDS_UART_PORT (Serial1)

#endif

//Load sensors / instances

#ifndef HDC1080_CONNECTED

Adafruit_HDC1000 HDC = Adafruit_HDC1000();

float temperature = 0;

float humidity = 0;

#endif

#ifndef BMP280_CONNECTED

Adafruit_BMP280 BMP;

double pressure;

#endif

#ifndef TSL45315_CONNECTED

uint32_t lux;
```

```

    Makerblog_TSL45315 TSL = Makerblog_TSL45315(TSL45315_TIME_M4);

#endif

#ifndef VEML6070_CONNECTED

VEML6070 VEML;

uint16_t uv;

#endif

#ifndef SDS011_CONNECTED

SDS011 SDS(SDS_UART_PORT);

float pm10 = 0;

float pm25 = 0;

#endif

```

Nun müssen wir im Sketch die eben erstellten Device und Application EUI's sowie den App Key eingeben

#### ▼ Einfügen der Device und Application EUI und App Key

```

// This EUI must be in little-endian format, so least-significant-byte

// first. When copying an EUI from ttntctl output, this means to reverse

// the bytes. For TTN issued EUIs the last bytes should be 0xD5, 0xB3,

// 0x70.

static const u1_t PROGMEM APPEUI[8] = {DIE APPLICATION EUI HIER(lsb-Format)};

void os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);}

// This should also be in little endian format, see above.

static const u1_t PROGMEM DEVEUI[8] = {DIE DEVICE EUI HIER(lsb-Format)};

void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);}

// This key should be in big endian format (or, since it is not really a

// number but a block of memory, endianness does not really apply). In

// practice, a key taken from ttntctl can be copied as-is.

// The key shown here is the semtech default key.

static const u1_t PROGMEM APPKEY[16] = {DER APP KEY HIER(msb-Format)};

void os_getDevKey (u1_t* buf) { memcpy_P(buf, APPKEY, 16);}

```

Nachfolgend geben wir ein Interval an in welchen die Daten an das TTN-Netzwerk geschickt werden sollen. In diesem Sketch beträgt das Intervall alle 60 Sekunden. Des weiteren werden die Pins gemappt und eine `onEvent()` -Funktion zum Debuggen im Seriellen Monitor wird erstellt.

#### ▼ Intervall, Mapping und Debug

```

static osjob_t sendjob;

// Schedule TX every this Demajetny seconds (might become longer due to duty
// cycle limitations).

const unsigned TX_INTERVAL = 60;

// Pin mapping

const lmic_pinmap lmic_pins = {

    .nss = PIN_XB1_CS,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = LMIC_UNUSED_PIN,
    .dio = {PIN_XB1_INT, PIN_XB1_INT, LMIC_UNUSED_PIN},
};

void onEvent (ev_t ev) {

    senseBoxIO.statusGreen();

    DEBUG(os_getTime());

    switch(ev) {

        case EV_SCAN_TIMEOUT:
            DEBUG(F("EV_SCAN_TIMEOUT"));
            break;

        case EV_BEACON_FOUND:
            DEBUG(F("EV_BEACON_FOUND"));
            break;

        case EV_BEACON_MISSED:
            DEBUG(F("EV_BEACON_MISSED"));
            break;

        case EV_BEACON_TRACKED:
            DEBUG(F("EV_BEACON_TRACKED"));
            break;

        case EV_JOINING:
            DEBUG(F("EV_JOINING"));
            break;

        case EV_JOINED:
            DEBUG(F("EV_JOINED"));

            // Disable link check validation (automatically enabled

```

```

// during join, but not supported by TTN at this time).

LMIC_setLinkCheckMode(0);

break;

case EV_RFU1:

DEBUG(F("EV_RFU1"));

break;

case EV_JOIN_FAILED:

DEBUG(F("EV_JOIN_FAILED"));

break;

case EV_REJOIN_FAILED:

DEBUG(F("EV_REJOIN_FAILED"));

break;

case EV_TXCOMPLETE:

DEBUG(F("EV_TXCOMPLETE (includes waiting for RX windows)"));

if (LMIC.txrxFlags & TXRX_ACK)

DEBUG(F("Received ack"));

if (LMIC.dataLen) {

DEBUG(F("Received "));

DEBUG(LMIC.dataLen);

DEBUG(F(" bytes of payload"));

}

// Schedule next transmission

os_setTimedCallback(&sendjob, os_getTime() + sec2osticks(TX_INTERVAL), do_send);

break;

case EV_LOST_TSYNC:

DEBUG(F("EV_LOST_TSYNC"));

break;

case EV_RESET:

DEBUG(F("EV_RESET"));

break;

case EV_RXCOMPLETE:

// data received in ping slot

DEBUG(F("EV_RXCOMPLETE"));

break;

case EV_LINK_DEAD:

```

```

    DEBUG(F("EV_LINK_DEAD"));

    break;

    case EV_LINK_ALIVE:

    DEBUG(F("EV_LINK_ALIVE"));

    break;

    default:

    DEBUG(F("Unknown event"));

    break;

}

}

```

Als nächstes definieren wir das LoRa Paket was an das TTN-Netzwerk übermittelt wird. Mit `LoraMessage message` deklarieren wir das Paket `message` welchen sukzessiv die Messwerte unserer Sensoren hinzugefügt wird.

#### ▼ Erstellen der LoRa-Message

```

void do_send(osjob_t* j){

    // Check if there is not a current TX/RX job running

    if (LMIC.opmode & OP_TXRXPEND) {

        DEBUG(F("OP_TXRXPEND, not sending"));

    } else {

        LoraMessage message;

        //----Temperature----//

        //----Humidity----//

        #ifdef HDC1080_CONNECTED

            DEBUG(F("Temperature: "));

            temperature = HDC.readTemperature();

            DEBUG(temperature);

            message.addUint16((temperature + 18) * 771);

            delay(2000);

            DEBUG(F("Humidity: "));

            humidity = HDC.readHumidity();

            DEBUG(humidity);

            message.addHumidity(humidity);

        }
    }
}

```

```

    delay(2000);

#endif

//----Pressure----//

#ifndef BMP280_CONNECTED

float altitude;

pressure = BMP.readPressure()/100;

altitude = BMP.readAltitude(1013.25); //1013.25 = sea level pressure

DEBUG(F("Pressure: "));

DEBUG(pressure);

message.addUInt16((pressure - 300) * 81.9187);

delay(2000);

#endif

//----Lux----//

#ifndef TSL45315_CONNECTED

DEBUG(F("Illuminance: "));

lux = TSL.readLux();

DEBUG(lux);

message.addUInt8(lux % 255);

message.addUInt16(lux / 255);

delay(2000);

#endif

//----UV intensity----//

#ifndef VEML6070_CONNECTED

DEBUG(F("UV: "));

uv = VEML.getUV();

DEBUG(uv);

message.addUInt8(uv % 255);

message.addUInt16(uv / 255);

delay(2000);

#endif

//----PM----//

#ifndef SDS011_CONNECTED

uint8_t attempt = 0;

while (attempt < 5) {

```

```

        bool error = SDS.read(&pm25, &pm10);

        if (!error) {

            DEBUG(F("PM10: "));
            DEBUG(pm10);
            message.addUInt16(pm10 * 10);

            DEBUG(F("PM2.5: "));
            DEBUG(pm25);
            message.addUInt16(pm25 * 10);

            break;
        }

        attempt++;
    }

#endif

// Prepare upstream data transmission at the next possible time.

LMIC_setTxData2(1, message.getBytes(), message.getLength(), 0);

DEBUG(F("Packet queued"));

}

// Next TX is scheduled after TX_COMPLETE event.

}

```

Anschließend folgt die `setup()`-Funktion die du es aus den bisherigen Arduino-Sketches bereits kennen solltest. In dieser starten wir unsere Sensoren und den seriellen Monitor.

#### ▼ `setup()`-Funktion

```

void setup() {

#ifdef ENABLE_DEBUG
    Serial.begin(9600);
#endif

    delay(3000);

    // RFM9X (LoRa-Bee) in XBEE1 Socket
    senseBoxIO.powerXB1(false); // power off to reset RFM9X
    delay(250);
    senseBoxIO.powerXB1(true); // power on
}

```

```

// Sensor initialization

DEBUG(F("Initializing sensors..."));

#ifndef VEML6070_CONNECTED
    VEML.begin();
    delay(500);
#endif

#ifndef HDC1080_CONNECTED
    HDC.begin();
#endif

#ifndef BMP280_CONNECTED
    BMP.begin(0x76);
#endif

#ifndef TSL45315_CONNECTED
    TSL.begin();
#endif

#ifndef SDS011_CONNECTED
    SDS_UART_PORT.begin(9600);
#endif

DEBUG(F("Sensor initializing done!"));
DEBUG(F("Starting loop in 3 seconds."));

delay(3000);

// LMIC init

os_init();

// Reset the MAC state. Session and pending data transfers will be discarded.

LMIC_reset();

// Start job (sending automatically starts OTAA too)

do_send(&sendjob);

}

```

Zu guter Letzt fehlt jetzt noch die `loop()`-Funktion. In dieser wird deklariert, dass der in den globalen Variablen definierte `os_loop()` ausgeführt wird.

#### ▼ `loop()`-Funktion

```
void loop() {
```

```
    os_runloop_once();
}
```

Zum einfachen Copy&Paste hier der gesamte Sketch. Denk daran deine eigenen ID's und den App Key einzugeben.

#### ▼ Gesamter Sketch

```
/*
senseBox:home - Citizen Sensingplatform

Version: lorav2.0.0

Date: 2018-09-11

Homepage: https://www.sensebox.de https://www.opensensemap.org

Author: Reedu GmbH & Co. KG

Note: Sketch for senseBox:home LoRa MCU Edition

Model: homeV2lora

Email: support@sensebox.de

Code is in the public domain.

https://github.com/sensebox/node-sketch-templater

*/
#include <LoraMessage.h>
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>
#include <senseBoxIO.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_HDC1000.h>
#include <Adafruit_BMP280.h>
#include <Makerblog_TSL45315.h>
#include <VEML6070.h>
#include <SDS011-select-serial.h>

// Uncomment the next line to get debugging messages printed on the Serial port
// Do not leave this enabled for long time use

#define ENABLE_DEBUG
#ifdef ENABLE_DEBUG
#define DEBUG(str) Serial.println(str)
#else
```

```
#define DEBUG(str)

#endif

// Connected sensors

// Temperatur

#define HDC1080_CONNECTED

// rel. Luftfeuchte

#define HDC1080_CONNECTED

// Luftdruck

#define BMP280_CONNECTED

// Beleuchtungsstärke

#define TSL45315_CONNECTED

// UV-Intensität

#define VEML6070_CONNECTED

// PM10

#define SDS011_CONNECTED

// Number of serial port the SDS011 is connected to. Either Serial1 or Serial2

#ifdef SDS011_CONNECTED

#define SDS_UART_PORT (Serial1)

#endif

//Load sensors / instances

#ifdef HDC1080_CONNECTED

Adafruit_HDC1000 HDC = Adafruit_HDC1000();

float temperature = 0;

float humidity = 0;

#endif

#ifdef BMP280_CONNECTED

Adafruit_BMP280 BMP;

double pressure;

#endif

#ifdef TSL45315_CONNECTED

uint32_t lux;

Makerblog_TSL45315 TSL = Makerblog_TSL45315(TSL45315_TIME_M4);

#endif

#ifdef VEML6070_CONNECTED

VEML6070 VEML;
```

```

    uint16_t uv;

#endif

#ifndef SDS011_CONNECTED

SDS011 SDS(SDS_UART_PORT);

float pm10 = 0;

float pm25 = 0;

#endif

//measurement variables
const int GAUGE_PIN = 6; // Pin connected to reed switch

unsigned int rainCounter = 0;

// This EUI must be in little-endian format, so least-significant-byte

// first. When copying an EUI from ttncctl output, this means to reverse

// the bytes. For TTN issued EUIs the last bytes should be 0xD5, 0xB3,

// 0x70.

static const u1_t PROGMEM APPEUI[8]={ 0x33, 0x2F, 0x01, 0xD0, 0x7E, 0xD5, 0xB3, 0x70 };

void os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);}

// This should also be in little endian format, see above.

static const u1_t PROGMEM DEVEUI[8]={ 0x48, 0x33, 0xD0, 0x7A, 0xE7, 0xA7, 0x5D, 0x00 };

void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);}

// This key should be in big endian format (or, since it is not really a

// number but a block of memory, endianness does not really apply). In

// practice, a key taken from ttncctl can be copied as-is.

// The key shown here is the semtech default key.

static const u1_t PROGMEM APPKEY[16] ={ 0xAC, 0x9E, 0x89, 0x27, 0x70, 0x06, 0xAF, 0x89, 0xC3, 0xB3, 0xE0, 0xA6,
0x3F, 0x74, 0x1C, 0x23 };

void os_getDevKey (u1_t* buf) { memcpy_P(buf, APPKEY, 16);}

static osjob_t sendjob;

// Schedule TX every this Demajetny seconds (might become longer due to duty

// cycle limitations).

const unsigned TX_INTERVAL = 60;

// Pin mapping

const lmic_pinmap lmic_pins = {

.nss = PIN_XB1_CS,
.rxtx = LMIC_UNUSED_PIN,

```

```

.rst = LMIC_UNUSED_PIN,
.dio = {PIN_XB1_INT, PIN_XB1_INT, LMIC_UNUSED_PIN},
};

void onEvent (ev_t ev) {
    senseBoxIO.statusGreen();
    DEBUG(os_getTime());
    switch(ev) {
        case EV_SCAN_TIMEOUT:
            DEBUG(F("EV_SCAN_TIMEOUT"));
            break;
        case EV_BEACON_FOUND:
            DEBUG(F("EV_BEACON_FOUND"));
            break;
        case EV_BEACON_MISSED:
            DEBUG(F("EV_BEACON_MISSED"));
            break;
        case EV_BEACON_TRACKED:
            DEBUG(F("EV_BEACON_TRACKED"));
            break;
        case EV_JOINING:
            DEBUG(F("EV_JOINING"));
            break;
        case EV_JOINED:
            DEBUG(F("EV_JOINED"));
            // Disable link check validation (automatically enabled
            // during join, but not supported by TTN at this time).
            LMIC_setLinkCheckMode(0);
            break;
        case EV_RFU1:
            DEBUG(F("EV_RFU1"));
            break;
        case EV_JOIN_FAILED:
            DEBUG(F("EV_JOIN_FAILED"));
            break;
        case EV_REJOIN_FAILED:
            break;
    }
}

```

```

    DEBUG(F("EV_REJOIN_FAILED"));

    break;

    case EV_TXCOMPLETE:

        DEBUG(F("EV_TXCOMPLETE (includes waiting for RX windows)"));

        if (LMIC.txrxFlags & TXRX_ACK)

            DEBUG(F("Received ack"));

        if (LMIC.dataLen) {

            DEBUG(F("Received "));

            DEBUG(LMIC.dataLen);

            DEBUG(F(" bytes of payload"));

        }

        // Schedule next transmission

        os_setTimedCallback(&sendjob, os_getTime() + sec2osticks(TX_INTERVAL), do_send);

        break;

    case EV_LOST_TSYNC:

        DEBUG(F("EV_LOST_TSYNC"));

        break;

    case EV_RESET:

        DEBUG(F("EV_RESET"));

        break;

    case EV_RXCOMPLETE:

        // data received in ping slot

        DEBUG(F("EV_RXCOMPLETE"));

        break;

    case EV_LINK_DEAD:

        DEBUG(F("EV_LINK_DEAD"));

        break;

    case EV_LINK_ALIVE:

        DEBUG(F("EV_LINK_ALIVE"));

        break;

    default:

        DEBUG(F("Unknown event"));

        break;

}

```

```

}

void do_send(osjob_t* j){

    // Check if there is not a current TX/RX job running

    if (LMIC.opmode & OP_TXRXPEND) {

        DEBUG(F("OP_TXRXPEND, not sending"));

    } else {

        LoraMessage message;

        //----Temperature----//

        //----Humidity----//

        #ifdef HDC1080_CONNECTED

            DEBUG(F("Temperature: "));

            temperature = HDC.readTemperature();

            DEBUG(temperature);

            message.addUint16((temperature + 18) * 771);

            delay(2000);

        DEBUG(F("Humidity: "));

        humidity = HDC.readHumidity();

        DEBUG(humidity);

        message.addHumidity(humidity);

        delay(2000);

#endif

        //----Pressure----//

        #ifdef BMP280_CONNECTED

            float altitude;

            pressure = BMP.readPressure()/100;

            altitude = BMP.readAltitude(1013.25); //1013.25 = sea level pressure

            DEBUG(F("Pressure: "));

            DEBUG(pressure);

            message.addUint16((pressure - 300) * 81.9187);

            delay(2000);

#endif

        //----Lux----//

        #ifdef TSL45315_CONNECTED

            DEBUG(F("Illuminance: "));


```

```

lux = TSL.readLux();

DEBUG(lux);

message.addUInt8(lux % 255);

message.addUInt16(lux / 255);

delay(2000);

#endif

//----UV intensity----//

#ifndef VEML6070_CONNECTED

DEBUG(F("UV: "));

uv = VEML.getUV();

DEBUG(uv);

message.addUInt8(uv % 255);

message.addUInt16(uv / 255);

delay(2000);

#endif

//----PM----//

#ifndef SDS011_CONNECTED

uint8_t attempt = 0;

while (attempt < 5) {

bool error = SDS.read(&pm25, &pm10);

if (!error) {

DEBUG(F("PM10: "));

DEBUG(pm10);

message.addUInt16(pm10 * 10);

DEBUG(F("PM2.5: "));

DEBUG(pm25);

message.addUInt16(pm25 * 10);

break;

}

attempt++;

}

#endif

// Prepare upstream data transmission at the next possible time.

LMIC_setTxData2(1, message.getBytes(), message.getLength(), 0);

```

```
    DEBUG(F("Packet queued"));

}

// Next TX is scheduled after TX_COMPLETE event.

}

void setup() {

#define ENABLE_DEBUG

Serial.begin(9600);

#endif

delay(3000);

// RFM9X (LoRa-Bee) in XBEE1 Socket

senseBoxIO.powerXB1(false); // power off to reset RFM9X

delay(250);

senseBoxIO.powerXB1(true); // power on


// Sensor initialization

DEBUG(F("Initializing sensors..."));

#ifndef VEML6070_CONNECTED

VEML.begin();

delay(500);

#endif

#ifndef HDC1080_CONNECTED

HDC.begin();

#endif

#ifndef BMP280_CONNECTED

BMP.begin(0x76);

#endif

#ifndef TSL45315_CONNECTED

TSL.begin();

#endif

#ifndef SDS011_CONNECTED

SDS_UART_PORT.begin(9600);

#endif


DEBUG(F("Sensor initializing done!"));

DEBUG(F("Starting loop in 3 seconds."));
```

```
delay(3000);

// LMIC init

os_init();

// Reset the MAC state. Session and pending data transfers will be discarded.

LMIC_reset();

// Start job (sending automatically starts OTAA too)

do_send(&sendjob);

}

void loop() {

os_runloop_once();

}
```

## Decoding Profile

Das Decoding wird nun von der openSenseMap durchgeführt. Es ist also keine manuelle Einrichtung eines Decoders nötig. Achte nur darauf, dass du bei der Registration deiner senseBox auf der openSenseMap die Datenübertragung per LoRa angibst!

1. <https://thethingsnetwork.org> ↵
2. <https://thethingsnetwork.org> ↵
3. <https://console.thethingsnetwork.org/> ↵

## Sensoren

Mit Sensoren kannst du verschiedene Umweltpheomene messen und damit Phänomene erkennen, beobachten und analysieren.

## Übersicht

In unserem [senseBox-Shop](#)<sup>1</sup> findest du eine Liste mit Sensoren, mit denen auch wir arbeiten und deshalb kleine Anleitungen geschrieben haben. Zu diesen Sensoren bieten wir dir hier Informationen, aber auch Hilfe beim Anschliessen an die senseBox.

Hier findest du Informationen zu den folgenden Sensoren, klicke einfach den Namen des Sensors und du wirst auf eine eigene Seite weitergeleitet:

- Temperatur & Luftfeuchte (HDC1080)<sup>1</sup>
- Luftdruck & Temperatur<sup>2</sup>
- Belichtung und UV<sup>3</sup>
- Feinstaub<sup>4</sup>

Natürlich kannst Du auch jeden anderen dir bekannten Sensor an die senseBox anschliessen, dann bist du als Tüftler allerdings selbst gefragt, wenn es erstmal an das Verkabeln und Programmieren geht ;) Wenn du Lust hast kannst du aber gerne deinen Sensor beschreiben und wir nehmen die Anleitung mit in dieses Book auf.

---

<sup>1</sup>. <https://sensebox.kaufen/> ↵

<sup>1</sup>. Siehe [5.1.3.1 Temperatur & Luftfeuchte \(HDC1080\)](#) ↵

<sup>2</sup>. Siehe [5.1.3.2 Luftdruck & Temperatur](#) ↵

<sup>3</sup>. Siehe [5.1.3.3 Belichtung und UV](#) ↵

<sup>4</sup>. Siehe [5.1.3.4 Feinstaub](#) ↵

## Temperatur- und Luftfeuchtesensor (HDC1080)

Der HDC1080 ist ein digitaler Feuchtigkeits- und Temperatursensor. Der Sensor hat eine hohe Genauigkeit und eine sehr geringe Stromaufnahme und passt dadurch ideal zur senseBox. Die Sensoren sind ab Werk kalibriert und können direkt eingesetzt werden.



Temperatur und Luftfeuchtesensor

### Technische Information

- Relative Luftfeuchte (RH) Betriebsbereich 0% bis 100%
- 14 Bit Measurement Resolution
- Relative Luftfeuchte Genauigkeit  $\pm 4\%$
- Temperatur Genauigkeit  $\pm 0.2^\circ\text{C}$
- 2100nA Sleep Mode Current
- Betriebsspannung 2.7 V bis 5.5 V
- I2C Interface
- "Plug-in-and-Go" senseBox kompatibel
- Average Supply Current: 710 nA @ 1sps, 11 bit RH Measurement 1.3  $\mu\text{A}$  @ 1sps, 11 bit RH and Temperature Measurement

## Luftdruck- und Temperatursensor

Dieser Sensor misst den Luftdruck und die Temperatur und basiert auf dem BMP280 Sensor von Bosch.



*Der Luftdruck- und Temperatursensor*

### Technische Informationen

- Maße: 25mm x 25mm x 9mm
- Gewicht: 2,4 g
- "Plug-in-and-Go" senseBox kompatibel
- Betriebsdruck 300 bis 1100 hPa
- Relative Präzision  $\pm 0.12$  hPa
- Absolute Präzision  $\pm 1$  hPa
- Betriebsversorgungsstrom  $2.7\mu\text{A}$  bei 1 Hz Sampling Frequenz

## Beleuchtungsstärke und UV-Strahlung Sensor

Auf diesem senseBox-Bauteil sind zwei Sensoren zusammengelegt. Die Lichtintensität wird mit dem TSL45315-Sensor von AMS-TAOS gemessen. Dieser Sensor erkennt die Lichtverhältnisse ähnlich dem menschlichen Auge und gibt die Helligkeitswerte direkt in Lux, mit großem Dynamikbereich (3 Lux bis 220k Lux), aus. Der zweite Sensor ist ein Vishay VEML6070 Ultravioletter (UV)-Lichtsensor. Dieser wandelt die Intensität des UV-Lichts der Sonne in digitale Daten um. Der Sensor hat eine hervorragende UV-Empfindlichkeit und Linearität über Filtron™-Technologie. Er hat eine gute UV-Strahlungsmessung auch bei langer Sonnen-UV-Belastung und kann exzellenter Temperaturschwankungen ausgleichen.

### *Beleuchtungsstärke und UV-Strahlung*

## Technische Details

### Belichtungssensor

- 3,3V - 5V tolerantes I2C/TWI Interface
- Eingangsspannungsbereich: 3,3V - 5V
- on-board 2,5V Spannungsregler
- on-board Pegelwandler

### UV-Sensor

- Betriebsspannung: 2,7V - 5,5V I2C Interface
- Unterstützt Quittierungsfunktion (Active Acknowledge-Funktion)
- Temperaturkompensation: -40°C bis +85°C
- Software-Abschaltregelung für Immunität bei flackernden Leuchtstofflampen

### Maße

- 25mm x 25mm x 9mm
- Gewicht: 2,5 g



## Feinstaubsensor

Mit diesem Sensor SDS011 ist es möglich die Feinstaubkonzentration in der Luft zu bestimmen. Der Sensor gibt zwei Werte aus: Die Konzentration von PM2.5 (Partikel < 2.5 um) und PM10 (Partikel < 10 um). Dieser Sensor ist mit einem kleinen Ventilator ausgestattet, um Luft anzusaugen. In seinem Inneren befindet sich ein Laser, der zusammen mit einer Photodiode die Anzahl der Partikel misst. Die Ergebnisse der Messungen werden in  $\mu\text{g}/\text{m}^3$  (Mikrogramm pro Kubikmeter) angegeben.



Feinstaubsensor für PM10 und PM2.5

### Technische Details

- Schnelle Reaktionszeit von weniger als 10 Sekunden
- "Plug-in-and-Go" senseBox kompatibel
- Hohe Auflösung bis zu  $0.3\mu\text{g}/\text{m}^3$
- Mehrfach wissenschaftliche Prüfung der Datengenauigkeit

### Zum Aufbau benötigte Bauteile



Benötigte (mitgelieferte) Bauteile

- SDS011 Feinstaub Sensor
- Verbindungskabel
- Stück Teflonschlauch  $\varnothing = 6\text{mm}$  innen und  $\varnothing = 8\text{mm}$  außen
- Gehäuse
- Kabelverschraubung 16mm

### Anschluss und Programmierung

Mit dem mitgelieferten Verbindungskabel kannst du deinen Feinstaub Sensor mit dem "UART/Serial"-Port der senseBoxMCU verbinden. Ist dies erledigt können wir nun im Programmcode den Sensor initialisieren und uns die ersten Messwerte ausgeben lassen.

Stelle sicher, dass du das aktuellste Board-Support-Package installiert hast, da du die korrekten Software-Bibliotheken benötigst. Wie das geht wurde dir in [Schritt 2](#) erklärt!

Als erstes muss eine Instanz des Sensors erstellt werden. Dazu erstellen wir noch 2 Variablen in denen wir unsere beiden Messwerte für PM10 und PM2.5 speichern

```
SDS011 my_sds(Serial1) // Serial1 gibt hier den seriellen Port an, an dem du den Sensor angeschlossen hast  
float p10, p25
```

#### ▼ **setup()** Funktion

In der `setup()` -Funktion soll der Sensor nun gestartet werden:

```
void setup(){  
    // Normalen seriellen Port initialisieren  
    Serial.begin(9600);  
    while(!Serial);  
    // seriellen Port an dem unser Sensor angeschlossen ist initialisieren  
    Serial1.begin(9600);  
    delay(5000);  
}
```

#### ▼ **loop()** Funktion

In der `loop()` -Funktion können wir mit dem Befehl 'getPm10()' und 'getPm25()' die aktuell gemessenen Feinstaubwerte abrufen:

```
void loop(){  
    // Variablen den gemessenen Feinstaubwerte zuweisen  
    p10 = my_sds.getPm10();  
    p25 = my_sds.getPm25();  
    // Werte in der Konsole drucken  
    Serial.println("P2.5: "+String(p25));  
    Serial.println("P10: "+String(p10));  
    delay(1000);  
}
```

## Zubehör

Hier findest du alle weiteren Zubehör-Teile die sich nicht den Sensoren oder Bees zuordnen lassen. Darunter sind Teile, wie GPS-Modul, Gehäuse oder Strahlenschutz, die nicht standardmäßig in der edu-Version der senseBox enthalten sind. Um trotzdem herausfinden zu können wofür diese von Nutzen sind, findest dennoch Infos zu diesen Teilen.

## Übersicht

Es gibt folgende Komponenten die mit der senseBox angeboten und für euch getestet worden sind:

- [Strahlenschutz<sup>1</sup>](#)
- [Gehäuse<sup>2</sup>](#)
- [Netzteil und USB-Kabel<sup>3</sup>](#)
- [LED-Display<sup>4</sup>](#)
- [HUB<sup>5</sup>](#)
- [Micro-SD Karte<sup>6</sup>](#)
- [GPS<sup>7</sup>](#)

---

1. Siehe [5.1.4.1 Strahlenschutz ↵](#)

2. Siehe [5.1.4.2 Gehäuse ↵](#)

3. Siehe [5.1.4.3 Netzteil und USB-Kabel ↵](#)

4. Siehe [5.1.4.4 LED-Display ↵](#)

5. Siehe [5.1.4.5 Expander ↵](#)

6. Siehe [5.1.4.6 Micro-SD Karte ↵](#)

7. Siehe [5.1.4.7 GPS ↵](#)

## Strahlenschutz

Der Strahlenschutz schützt den Außensender für Temperatur + Luftfeuchtigkeit (BMP280) vor Witterung. Er dient als Schutz vor Niederschlag und direkter Sonneneinstrahlung. Er ist einfach zu montieren und hat eine Öffnung für die Fühlerkabel.

### Hinweise für die Standortwahl des Strahlenschutzes

- Suche Dir im Freien einen schattigen Platz für den Strahlenschutz aus. (Direkte Sonneneinstrahlung verfälscht die Messwerte).
- Prüfe, ob eine Übertragung der Messwerte vom Sender am gewünschten Aufstellort zur Basisstation stattfindet (bei massiven Wänden, insbesondere mit Metallteilen kann sich die Sendereichweite erheblich reduzieren).

### Montieren der Schutzhülle

- Drehe die Schutzhülle im Uhrzeigersinn von der Bodenplatte ab.
- Für einen sicheren Halt kann die Schutzhülle auf einem geeigneten Untergrund festgeschraubt werden.
- Suche Dir eine glatte, waagrechte Stelle aus.
- Führe eine Schraube durch die Öffnung in der Mitte und schraube die Bodenplatte fest.

### Befestigung an der Wand oder an einem Pfosten

- Entferne die Wandhalterung von der Schutzhülle, in dem Du sie nach unten schiebst.
- Befestige die Wandhalterung mit den Schrauben und den Dübeln an der Wand. Achte auf die Markierung „UP“ (oben).
- Möchtest Du die Wandhalterung an einem Pfosten anbringen, kannst Du auch die Kabelbinder verwenden.
- Hinweis: Achte beim Montieren bitte darauf, dass die Schutzhülle problemlos in die Wandhalterung eingesetzt werden kann.
- Setze die Schutzhülle mit der Bodenplatte und dem befestigten Sender von oben in die Wandhalterung ein.

### Befestigung des Senders

- Damit der Sender sicher auf der Bodenplatte steht, befestige das Klettband mit dem doppelseitigen Klebeband am Ständer und an der Rückseite des Senders und fixiere den Sender. Alternativ kannst Du auch die Kabelbinder verwenden oder mit Heißkleber arbeiten.
- Führe die Kabel durch die dafür vorgesehene Öffnung in der Bodenplatte.
- Setze die Schutzhülle auf die Bodenplatte mit dem befestigten Sender und drehe sie gegen den Uhrzeigersinn fest.

### Pflege und Wartung

- Reinige die Schutzhülle mit einem weichen, leicht feuchten Tuch. Keine Scheuer- oder Lösungsmittel verwenden!

## Technische Daten

Innenraummaße: Höhe 160 mm Innendurchmesser: 60 mm Gehäusemaße: 95 x 102 (108) x 180 mm Gewicht: 163 g

## Gehäuse

**Das Gehäuse schützt den Mikrocontroller und einige Sensoren vor der Witterung.**

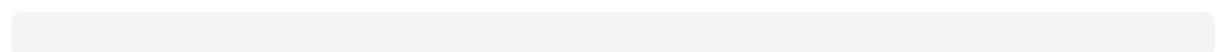


### *Das Gehäuse*

Das Gehäuse aus Polycarbonat mit durchsichtigem Deckel schützt die senseBox MCU, sowie die dazugehörigen Sensoren vor der Witterung.

Maße: 82mm x 162mm x 85mm Gewicht: 300 g

## Netzteil und USB-Kabel



## LED-Display

Dieser hochwertige OEL Display von Univision ist einer der neusten Komponenten zur senseBox. Es stehen 128 x 64 Pixel zum anzeigen von Werten, oder Statusmeldungen der senseBox zur Verfügung.



Das OLED-Display

### Technische Details

- Number of Pixels: 128 × 64
- Panel Size: 26.70 × 19.26 × 1.45 (mm)
- Active Area: 21.744 × 10.864 (mm)
- Pixel Pitch: 0.17 × 0.17 (mm)
- Pixel Size: 0.154 × 0.154 (mm)
- Wieght: 1.54 (g)

### Zubehör

- JST to JST Kabel(300mm)

### Anschluss und Programmierung

Mit dem , in deiner senseBox beiliegenden, "JST to JST" Kabel kannst du das Display an einen I<sup>2</sup>C-Port deiner senseBoxMCU anschließen. Ist dies erledigt können wir nun im Programmcode das Display initialisieren und unsere Messwerte live auf dem Display anzeigen!

Stelle sicher, dass du das aktuellste Board-Support-Package installiert hast, da du die korrekten Software-Bibliotheken benötigst. Wie das geht wurde dir in [Schritt 2](#) erklärt!

Anfangs müssen einige neue Libraries einladen, damit das Display funktioniert.

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <senseBoxIO.h>
```

```
#include "SenseBoxMCU.h"
```

Ist dies passiert, deklarieren wir nun unsere Variablen für das Display.

```
#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_RESET);
```

#### ▼ setup() Funktion

In der `setup()`-Funktion soll das Display gestartet werden.

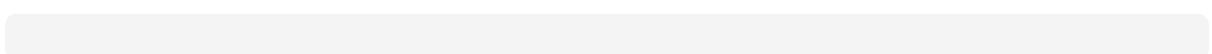
```
void setup(){
    senseBoxIO.powerI2C(true);
    delay(2000);
    display.begin(SSD1306_SWITCHCAPVCC, 0x3D);
    display.display();
    delay(100);
    display.clearDisplay();
}
```

#### ▼ loop() Funktion

In der `loop()`-Funktion wird nun die Textfarbe, Schriftgröße und Wert den das Display anzeigen soll ausgegeben.

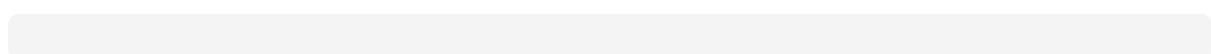
```
void loop(){
    display.setCursor(0,0);
    display.setTextSize(1); // Verändere Schriftgröße hier
    display.setTextColor(WHITE,BLACK); // Verändere Schriftfarbe hier
    /* Gebe hier an was dein Display anzeigen soll!
       Hier kannst du auch die Ergebnisse deiner Messwerte anzeigen lassen */
    display.println("senseBox rocks!");
    display.display();
}
```

## Expander



*Expander*

## Micro-SD-Karte



## GPS

Das GPS-Modul empfängt die Position (Längengrad/Breitengrad/Höhe) der senseBox. Diese kann für mobile Anwendungen, z.B. die mobile Datenübertragung auf die openSenseMap. Dieser Sensor ist kompatibel mit den gängigen GNS Systemen (GPS, QZSS, GLONASS, BeiDou, Galileo) und basiert auf dem u-blox CAM-M8Q Multi GNSS Modul.



*Der GPS-Sensor*

## Funktion

Der GPS Sensor wird an einen I2C-Port angeschlossen. Ein Beispiel wie man den GPS-Sensor verwendet findet ihr in Kapitel [Mobile Station](#)<sup>1</sup>

---

<sup>1</sup>. [https://sensebox.github.io/books-v2/edu/de/projekte/Mobile\\_Station.html](https://sensebox.github.io/books-v2/edu/de/projekte/Mobile_Station.html) ↵

# FAQ

Hier findest du häufig gestellte Fragen und Antworten. Wenn du Probleme hast schaue zuerst hier nach ob deine Frage beantwortet wird. Ansonsten schau in unser Forum oder stell dort deine Fragen.

## Fragen zur Programmierung

### ▼ Ich möchte eine externe Library einbinden. Geht das?

Ja das geht. Die senseBox ist nicht auf die mitgelieferten Sensoren beschränkt. Du kannst sie mit jeglichen Sensoren erweitern. Zur Programmierung werden oft externe Libraries von den Anbietern der Sensoren benötigt.

Wie du diese manuell in die Arduino IDE einbinden kannst erfährst du auf [dieser Hilfeseite<sup>1</sup>](#)!

## Fragen zur Datenübertragung

### ▼ Ich habe Probleme bei der Übertragung von Daten per Wifi. Was kann ich tun?

Bei einigen unserer WiFi Bees vom Typ WINC1500 kann es vorkommen, dass eine veraltete Firmware (Version 19.4.4) installiert ist. Das kann zu Übertragungsproblemen führen. Sollten diese Probleme bei dir auftreten, schaue dir [diese Hilfeseite<sup>2</sup>](#) an, um die Firmware zu aktualisieren.

## Fragen zur Verbindung von senseBox mit dem Computer

### ▼ Ich habe Probleme bei dem Verbinden meiner senseBox MCU mit meinem Windows-Rechner. Was kann ich tun?

Bei einigen unserer Windows-Rechnern kann es vorkommen, dass die USB-Bootloader Treiber nicht korrekt installiert sind. Das kann dazu führen, dass der Computer die senseBox MCU nicht als USB-Gerät erkennt und somit keine Dateien übertragen kann. Sollten diese Probleme bei dir auftreten, schaue dir [diese Hilfeseite<sup>3</sup>](#) an, um zu prüfen, ob deine Treiber funktionieren und sie ggf. zu aktualisieren.

## Fragen zum senseBox-Projekt

### ▼ Wo kann ich die neue senseBox Version 2 kaufen?

Ganz einfach, gehe auf [sensebox.kaufen<sup>1</sup>](#) und klicke dir dort die senseBox passend zu deinen Bedürfnissen zusammen.

### ▼ Wie kann ich auf dem neusten Stand bleiben was die senseBox angeht?

Melde dich ganz einfach auf [sensebox.de<sup>2</sup>](#) zu unserem Newsletter an und verpasste nie wieder, wenn es etwas neues gibt!

- | 1. Siehe [6.3 Externe Libraries hinzufügen](#) ↵
- | 2. Siehe [6.4 Firmware Update Wifi-Bee](#) ↵
- | 3. Siehe [6.5 Windows USB-Bootloader Treiber aktualisieren](#) ↵
- | 1. <https://sensebox.kaufen> ↵
- | 2. <https://sensebox.de/> ↵

## Downloads

In diesem Bereich findest du verschiedene Downloads, die dir bei der Verwendung der senseBox helfen können.

### Dokumentation als PDF

Dieses Buch ist auch als PDF zum ausdrucken verfügbar!

PDF-Download: [senseBox:edu-Dokumentation<sup>1</sup>](https://github.com/sensebox/books-v2/raw/gh-pages/senseBox:edu.pdf)

Weitere nützliche Downloads folgen in Kürze ;)

---

<sup>1</sup>. [https://github.com/sensebox/books-v2/raw/gh-pages/senseBox:edu.pdf ↗](https://github.com/sensebox/books-v2/raw/gh-pages/senseBox:edu.pdf)

## Manuelles Einbinden von Libraries

Um eigene Sensoren an die senseBox anschließen zu können, geben die Hersteller vieler Sensoren passende Libraries mit. Hier zeigen wir euch, wie ihr aus einem Github-Repository Libraries herunterladen und diese manuell einbinden könnt. Ihr könnt diese Anleitung für jegliche externe Libraries verwenden, die ihr in Arduino einbinden möchtet.

ACHTUNG: Die für die senseBox benötigten Libraries befinden sich bereits im Board-Support-Package, welches in Schritt 2 heruntergeladen wird. Ihr solltet diese nicht mehr manuell einbinden, so wie es in früheren Versionen dieses Buches angegeben wurde. Dadurch entstehen Doppelungen, die zu Fehlern führen können.

### Libraries herunterladen und hinzufügen

#### ▼ 'Library' – Was ist das eigentlich und wofür brauche ich das?

Eine Library ist wie der Name schon sagt eine Sammlung von etwas – eine Sammlung von Methoden um genauer zu sein. Methoden sind in der Programmierung kleinere Abschnitte von Code, die auf ein Objekt angewendet werden können. Bei der senseBox zum Beispiel kann eine Methode aufgerufen werden, um die LEDs auf dem MCU ein- und auszuschalten. Es gibt eine Menge solcher Standardmethoden, die von einer Vielzahl an Programmen benutzt werden. Um diese Methoden nicht alle einzeln in den Programmcode übertragen zu müssen, können sie in Libraries abgelegt werden. Eine Library ist also eine Datei, in der viele Methoden gespeichert werden. Man kann Libraries in seinen Code einbinden. Dafür reicht es wenn sie im Arduino-Ordner für Libraries gespeichert sind und man sie dann mit einer einzigen Zeile zu Beginn des Programmcodes einbindet. Das sieht in Arduino für die Library mit dem Namen "senseBoxIO" wie folgt aus:

```
#include <senseBoxMCU.h>;
```

Ist die Library eingebunden, können alle in ihr enthaltenen Methoden im Code benutzt werden.

Bei der manuellen Installation der Libraries können sehr schnell Fehler auftreten, daher sollte man hier besonders genau auf die einzelnen Schritte achten. Um dich bei der Installation möglichst gut zu unterstützen, haben wir für jedes Betriebssystem eine separate Anleitung geschrieben. Wähle das zu deinem Computer passende System und folge den angegebenen Schritten.

- [Libraries einfügen Windows](#)
- [Libraries einfügen Mac](#)
- [Libraries einfügen Linux](#)

### Libraries einfügen Windows

1. Die meisten externen Libraries findest du in Github-Repositories. Um sie herunterzuladen, musst du den grünen Button `clone or download` und daraufhin im sich öffnenden Fenster `Download ZIP` klicken.



*Beispielhafter Download der \*

2. Sollte der Download nicht von alleine starten, öffnet sich ein Fenster, in dem du das Feld `Datei speichern auswählen` musst und den Ordner an einen beliebigen Ort auf deinem Computer legst (standardmäßig ist das der Downloads-Ordner).
3. Die heruntergeladene Datei ist ein `.zip`-Archiv, also eine komprimierte Version der Library. Es gilt daher als nächstes dieses `.zip`-Archiv zu entpacken. Dafür öffnest du den Speicherort des Archivs und klickst es mit Rechtsklick an und wählst im erscheinenden Menü `Alle extrahieren...`. Wähle als Speicherort denselben Ordner wie beim Download (z.B. den Downloads-Ordner).
4. Öffne nun die Arduino IDE. Gehe auf `Datei -> Voreinstellungen`:



*Klicke 'Datei' und dann 'Voreinstellungen'*

und schaue im Feld unter `Sketchbook-Speicherort` nach an welchem Ort der Sketchbook Ordner gespeichert ist.



*Schaue im rot-markierten Feld nach, wo dein Sketchbook-Speicherort ist*

Merke dir den Pfad zu diesem Ordner, also den Ort wo dieser gespeichert ist.

Du musst den Library-Ordner, den du bereits heruntergeladen und entpackt hast, im nächsten Schritt in den Sketchbook-Speicherort verschieben. Es ist daher sehr wichtig, dass du dir den entsprechenden Speicherort aus Punkt 4 genau merbst, um so später auftretende Fehler zu vermeiden.

5. Nun navigierst du in deinem Datei-Explorer zum Sketchbook-Speicherort (siehe 4.). Beachte, dass der Zielordner im Datei-Explorer am Sketchbook-Speicherort den Namen `Arduino` trägt. Wähle den Ordner mit Doppelklick aus um seinen Inhalt zu sehen. Der Ordner enthält einen weiteren Ordner mit dem Namen "libraries".

► **Was mache ich, wenn es keinen 'libraries'-Ordner gibt?**

Kopiere oder ziehe jetzt den heruntergeladenen und entpackten Ordner in den `libraries`-Ordner.

6. Schließe jetzt das Programm Arduino vollständig und starte es erneut, um die Installation der entsprechenden Libraries abzuschließen.

Leider ist ein typischer Fehler, dass die senseBox Library nicht in den richtigen Ordner gelegt wird. Bitte überprüfe nochmal, ob du die Datei in den richtigen Ordner aus 4. gelegt hast.

## Libraries einfügen Mac

1. Die meisten externen Libraries findest du in Github-Repositories. Um sie herunterzuladen, musst du den grünen Button `Clone or download` und daraufhin im sich öffnenden Fenster `Download ZIP` klicken.



*Beispielhafter Download der \*

2. Der Download sollte von alleine starten und die Datei automatisch entpackt und in deinem "Downloads"-Ordner abgelegt werden. Öffne den Downloads-Ordner und schaue ob der heruntergeladene Ordner dort vorhanden ist. Sollte dort anstelle eines Ordners nur eine .zip-Datei liegen, doppel-klicke diese, um sie zu entpacken.
3. Öffne nun die Arduino IDE. Gehe oben auf `Arduino -> Einstellungen ...`:



*Klicke 'Arduino' und dann 'Einstellungen ...'*

und schaue im Feld unter `Sketchbook-Speicherort` nach an welchem Ort der Sketchbook Ordner gespeichert ist.



*Schaue im rot-markierten Feld nach, wo dein Sketchbook-Speicherort ist*

Merke dir den Pfad zu diesem Ordner, also den Ort wo dieser gespeichert ist.

Du musst die Libraries, die du bereits heruntergeladen hast, im nächsten Schritt in den Sketchbook-Speicherort verschieben. Es ist daher sehr wichtig, dass du dir den entsprechenden Speicherort aus Punkt 3 genau merkst, um so später auftretende Fehler zu vermeiden.

- Nun navigierst du in deinem Finder zum Sketchbook-Speicherort. Beachte, dass der Zielordner im Finder am Sketchbook-Speicherort den Namen `Arduino` trägt. Wähle den Ordner mit Doppelklick aus um seinen Inhalt zu sehen. Der Ordner enthält einen weiteren Ordner mit dem Namen "libraries".

► Was mache ich, wenn es keinen 'libraries'-Ordner gibt?

Kopiere oder ziehe jetzt den heruntergeladenen (entpackten) Ordner in den `libraries`-Ordner.

- Schließe jetzt das Programm Arduino vollständig und starte es erneut, um die Installation der entsprechenden Libraries abzuschließen.

Leider ist ein typischer Fehler, dass die `senseBox` Library nicht in den richtigen Ordner gelegt wird. Bitte überprüfe nochmal, ob du die Datei in den richtigen Ordner aus 3. gelegt hast.

## Libraries einfügen Linux

- Die meisten externen Libraries findest du in Github-Repositories. Um sie herunterzuladen, musst du den grünen Button `clone or download` und daraufhin im sich öffnenden Fenster `Download ZIP` klicken.



*Beispielhafter Download der \*

- Der Download startet von alleine und legt ein `.zip`-Archiv in deinem Downloads-Ordner ab. Öffne den Downloads-Ordner und entpacke die `.zip` Datei mit Rechts-Klick -> `Extrahe Hier ( Extract Here )`.
- Öffne nun die Arduino IDE. Gehe auf `Datei -> Voreinstellungen`:



*Klicke 'Datei' und dann 'Voreinstellungen'*

und schaue im Feld unter `Sketchbook-Speicherort` nach an welchem Ort der Sketchbook Ordner gespeichert ist.



*Schaue im rot-markierten Feld nach, wo dein Sketchbook-Speicherort ist*

Merke dir den Pfad zu diesem Ordner, also den Ort wo dieser gespeichert ist.

Du musst die Libraries, die du bereits heruntergeladen und entpackt hast, im nächsten Schritt in den Sketchbook-Speicherort verschieben. Es ist daher sehr wichtig, dass du dir den entsprechenden Speicherort aus Punkt 3 genau merkst, um so später auftretende Fehler zu vermeiden.

4. Nun navigierst du in deinem Datei-Explorer zum Sketchbook-Speicherort (siehe 3.). Beachte, dass der Zielordner im Datei-Explorer am Sketchbook-Speicherort den Namen `Arduino` trägt. Wähle den Ordner mit Doppelklick aus um seinen Inhalt zu sehen. Der Ordner enthält einen weiteren Ordner mit dem Namen "libraries".

► **Was mache ich, wenn es keinen 'libraries'-Ordner gibt?**

Kopiere oder ziehe jetzt den heruntergeladenen (entpackten) Ordner in den `libraries` -Ordner.

5. Schließe jetzt das Programm Arduino vollständig und starte es erneut, um die Installation der entsprechenden Libraries abzuschließen.

Leider ist ein typischer Fehler, dass die `senseBox` Library nicht in den richtigen Ordner gelegt wird. Bitte überprüfe nochmal, ob du die Datei in den richtigen Ordner aus 3. gelegt hast.

## Firmware Update Wifi-Bee

Leider haben einige unserer WiFi Bees vom Typ WINC1500 eine veraltete Firmware (Version 19.4.4) installiert. Leider gibt es keine andere Möglichkeit dieses Firmware zu updaten als es manuell durchzuführen. Im folgenden Kapitel wird erklärt wie man rausfindet welche Firmware man benutzt und (falls man eine veraltete Version hat) wie man diese updatet.

### Test der Version

Zuerst muss rausgefunden werden, welche Version das gelieferte WiFi-Bee hat. Gehe dazu auf `Datei -> Beispiele` und unter "Beispiele für senseBox MCU" auf `Test_WINC1500`.



*Open the WiFi-Test*

Lade nun den Sketch auf dein Board (durch klicken des Pfeil-Symbols). Beachte, dass auf deinem senseBox Board das WiFi-Bee aufgesteckt sein muss (bitte auf XBEE1 aufstecken). Klicke dann auf den seriellen Monitor (durch klicken des Lupen-Symbols) und es wird geprüft ob dein WiFi-Bee funktionsfähig ist und angezeigt welche Firmware darauf installiert ist.



*Test results with a non-current firmware*

Wenn du eine Firmware 19.5.2, oder höher hast kannst du hier abbrechen. Dein WiFi-Bee funktioniert einwandfrei.

Wenn du eine Firmware geringer als 19.5.2 hast musst du leider die Firmware updaten. Wie das funktioniert erfährst du im nächsten Schritt.

### WiFi-Bee Firmware Update

Um die Firmware upzudaten, folge dem Pfad von oben: Datei -> Beispiele und unter "Beispiele für senseBox MCU" auf WINC1500\_Updater .



#### *Open the WINC1500\_Updater*

Lade nun den Sketch auf dein Board (durch klicken des Pfeil-Symbols), auf welchem das WiFi-Bee aufgesteckt ist (bitte auf XBEE1 aufstecken).

Öffne dieses Mal nicht den seriellen Monitor (nicht auf das Lupen-Symbols klicken)

Jetzt auf Werkzeuge klicken und Wifi 101 Firmware Updater auswählen.



#### *Choose Wifi 101 Firmware Updater*

Zuerst kannst du die Verbindung testen, indem du auf den angezeigten COM Port klickst und danach auf Test connection . Es sollte folgende Information zurück kommen: "The programmer is working!"



#### *Test connection of the WiFi-Bee*

Fast geschafft, jetzt nur noch auf Update Firmware klicken und der Upload beginnt. Danach sollte eine Erfolgsmeldung kommen "The firmware has been updated!".



#### *Finally Update Firmware*

Nicht davon irritieren lassen, dass es auch eine Version der Firmware 19.5.4 gibt, diese aber nicht in Arduino zu finden ist wenn man die Firmware updaten möchte. Diese Firmware wird mit dem Arduino Release 1.8.6 integriert und ist ab dann zu finden.

Wir entschuldigen uns für den Umweg und wünschen auch weiterhin viel Spaß mit der senseBox.

Falls dir dieser Artikel nicht weitergeholfen hat, kannst du versuchen auf [www.forum.sensebox.de](http://www.forum.sensebox.de) nach einer Lösung suchen, oder gegebenenfalls selbst einen Beitrag einreichen.

# Windows USB-Bootloader Treiber aktualisieren

Unter Windows kann es in seltenen Einzelfällen zu Problemen mit den Treibern des USB-Bootloader kommen. Was du tun kannst, um zu überprüfen ob deine Treiber korrekt installiert sind und wie du sie ggf. installieren kannst erfährst du hier.

## Treiberstatus überprüfen

Um zu überprüfen, ob deine Treiber für den USB-Bootloader funktionieren, befolge die folgenden Schritte:

- Schließe deine senseBox MCU Board per USB-Kabel an deinen Windows-Computer an
- Öffne den Geräte-Manager, indem du in der Windowsleiste nach `Geräte-Manager` suchst und diesen durch einen Klick öffnest.
- Aktiviere den Bootloadermodus der senseBox MCU indem du den `Reset`-Knopf (roter mechanischer Knopf auf der senseBox MCU), zweimal schnell nacheinander drückst
- Im Gerätemanager sollte nun ein Punkt `Anschlüsse (COM & LPT)` erscheinen
- Durch anklicken des Punktes öffnet sich eine Liste mit angeschlossenen Geräten, dort sollte die senseBox MCU aufgeführt sein
- Sollte die senseBox MCU nicht aufgeführt sein, trenne das USB-Kabel vom Computer und verbinde sie erneut - lasse dabei den Gerätemanager geöffnet und schau dir an was passiert

Falls dort kein entsprechendes Gerät angezeigt wird und auch auf erneutes Verbinden kein neues Gerät angezeigt wird, sind deine USB-Bootloader Treiber nicht korrekt installiert. Lade dir die aktuellen Treiber hier mit einem Klick herunter:

[senseBox MCU Treiber herunterladen<sup>1</sup>](#)

Gehe nun im Gerätemanager auf `Treiber aktualisieren` -> `Auf dem Computer suchen` und wähle die soeben heruntergeladenen Treiber aus.

Starte den Computer neu und überprüfe den Treiberstatus erneut, wie oben angegeben. Die senseBox MCU sollte nun erkannt werden.

Falls dir dieser Artikel nicht weitergeholfen hat, kannst du versuchen in unserem [Forum](#) nach einer Lösung suchen, oder gegebenenfalls selbst dort einen Beitrag erstellen.

---

<sup>1</sup>. <https://github.com/watterott/senseBox-MCU/raw/master/arduino/driver.zip> ↵

# Aktualisierung von Board-Support-Package und Libraries

Um die Installation und Updates der senseBox-Libraries benutzerfreundlicher zu gestalten, haben wir die Installationsschritte verändert.

Auf dieser Seite zeigen wir euch, welche Schritte ihr befolgen müsst, um euer Board-Support-Package und eure senseBox-Libraries zu aktualisieren. Diese Anleitung betrifft euch nur, wenn ihr die ersten Schritte dieses Buches vor dem 23. Juni 2018 durchgeführt habt.

## Was ist neu?

Wir haben ein neues Board-Support-Package entwickelt, welches das alte Board-Support-Package mit den senseBox-Libraries vereint. Dadurch wird die fehleranfällige, manuelle Installation der senseBox-Libraries umgangen. Gleichzeitig kann die integrierte Update-Funktion für Board-Support-Packages aus der Arduino IDE benutzt werden, um die Libraries auf den neusten Stand zu bringen. So können Updates in Zukunft mit deutlich geringerem Aufwand eingespielt werden.

## Anleitung zur Aktualisierung

Die Aktualisierung besteht aus 2 Schritten:

1. Dem Löschen der senseBox-Libraries aus dem Sketchbook-Ordner, um Doppelungen der Libraries und die Benutzung alter Versionen zu vermeiden.
2. Der Installation des neuen Board-Support-Packages, um die Libraries über dieses in Arduino einzubinden.

Wähle dein Betriebssystem, um die passende Anleitung zu sehen:

- Windows
- Mac(OSX)
- Linux

### Schritt 1: Löschen der senseBox-Libraries aus dem Sketchbook-Folder

1. Öffne nun die Arduino IDE. Gehe auf Datei -> Voreinstellungen :

Klicke 'Datei' und dann 'Voreinstellungen'

und schaue im Feld unter Sketchbook-Speicherort nach an welchem Ort der Sketchbook-Ordner gespeichert ist.

*Schau im rot-markierten Feld nach, wo dein Sketchbook-Speicherort ist*

Merke dir den Pfad zu diesem Ordner, also den Ort wo dieser gespeichert ist.

1. Nun navigierst du in deinem Datei-Explorer zum Sketchbook-Speicherort (siehe 1.). Beachte, dass der Zielordner im Datei-Explorer am Sketchbook-Speicherort den Namen `Arduino` trägt. Öffne diesen Ordner. Innerhalb des `Arduino`-Ordners befindet sich ein Ordner mit dem Namen `libraries`. Innerhalb dieses Ordners befinden sich die senseBox-Libraries. Lösche den `libraries`-Ordner, um sie zu entfernen.

Wenn du ein erfahrener Arduino-Nutzer bist und in der Vergangenheit weitere externe Libraries eingebunden hast, die nicht zu den senseBox-Libraries gehören, gehe in den 'libraries'-Ordner und lösche alle Libraries, die nicht extern von dir eingebunden wurden, anstatt den gesamten Ordner zu löschen.

2. Schließe jetzt das Programm Arduino vollständig und starte es erneut, um das Löschen der alten senseBox-Libraries abzuschließen.

## Schritt 2: Neues Board-Support-Package einbinden

Um das neue Board-Support-Package einzubinden, geht ihr ähnlich vor, wie in den ersten Schritten, mit ein paar kleinen Änderungen.

1. Füge die folgende URL in deiner Arduino IDE unter Datei -> Voreinstellungen in das Feld für Zusätzliche Bordverwalter-URLs ein:

`https://github.com/sensebox/senseBoxMCU-core/raw/master/package_sensebox_index.json`

An der Stelle steht im Normalfall vorher schon folgende URL: `https://github.com/watterott/senseBox-MCU/raw/master/package_sensebox_index.json` diese sieht der obigen sehr ähnlich, ist aber nicht die gleiche URL. Sie muss aber unbedingt durch die oben stehende URL ausgetauscht werden.

*Öffne die Voreinstellungen und füge die URL ein*

2. Öffne nun den Boardverwalter unter Werkzeuge -> Board:"..." -> Boardverwalter und suche dort nach dem senseBox SAMD Boards-Package.

*Suche nach dem rot markierten Package*

Wenn ihr auf den Eintrag in der Liste klickt, erscheint dort ein Update-Button.

Wichtig ist, zuerst auf den Eintrag zu klicken. Ansonsten wird der Update-Button nicht angezeigt, auch wenn es bereits eine neue Version gibt.

1. Klicke auf diesen Button und gehe danach sicher, dass die installierte Version höher als 1.1.0 ist.



*Klicke auf 'Update', um das Board-Support-Package zu aktualisieren*

Da wir das senseBox SAMD Boards-Package für euch regelmäßig aktualisieren, solltet ihr immer mal wieder in den Boardverwalter gehen und nachschauen, ob das senseBox SAMD Boards-Package noch aktuell ist. Öffnet dafür wie oben beschrieben den Boardverwalter, sucht nach senseBox SAMD Boards und klickt dort ggf. auf `Update`.

### Schritt 1: Löschen der senseBox-Libraries aus dem Sketchbook-Folder

1. Öffne nun die Arduino IDE. Gehe auf `Arduino` -> `Einstellungen...`:



*Klicke 'Arduino' -> 'Einstellungen...'`*

und schaue im Feld unter `Sketchbook-Speicherort` nach an welchem Ort der Sketchbook-Ordner gespeichert ist.



*Schaue im rot-markierten Feld nach, wo dein Sketchbook-Speicherort ist*

Merke dir den Pfad zu diesem Ordner, also den Ort wo dieser gespeichert ist.

1. Nun navigierst du in deinem Datei-Explorer zum Sketchbook-Speicherort (siehe 1.). Beachte, dass der Zielordner im Datei-Explorer am Sketchbook-Speicherort den Namen `Arduino` trägt. Öffne diesen Ordner. Innerhalb des `Arduino`-Ordners befindet sich ein Ordner mit dem Namen `libraries`. Innerhalb dieses Ordners befinden sich die senseBox-Libraries. Lösche den `libraries`-Ordner, um sie zu entfernen.

Wenn du ein erfahrener Arduino-Nutzer bist und in der Vergangenheit weitere externe Libraries eingebunden hast, die nicht zu den senseBox-Libraries gehören, gehe in den `'libraries'`-Ordner und lösche alle Libraries, die nicht extern von dir eingebunden wurden, anstatt den gesamten Ordner zu löschen.

2. Schließe jetzt das Programm Arduino vollständig und starte es erneut, um das Löschen der alten senseBox-Libraries abzuschließen.

## Schritt 2: Neues Board-Support-Package einbinden

Um das neue Board-Support-Package einzubinden, geht ihr ähnlich vor, wie in den ersten Schritten, mit ein paar kleinen Änderungen.

1. Füge die folgende URL in deiner Arduino IDE unter `Arduino -> Einstellungen...` in das Feld für Zusätzliche Bordverwalter-URLs ein:

```
https://github.com/sensebox/senseBoxMCU-core/raw/master/package_sensebox_index.json
```

An der Stelle steht im Normalfall vorher schon folgende URL: [https://github.com/watterott/senseBox-MCU/raw/master/package\\_sensebox\\_index.json](https://github.com/watterott/senseBox-MCU/raw/master/package_sensebox_index.json) diese sieht der obigen sehr ähnlich, ist aber nicht die gleiche URL. Sie muss aber unbedingt durch die oben stehende URL ausgetauscht werden.

*Öffne die Voreinstellungen und füge dort die URL ein*

2. Öffne nun den Boardverwalter unter Werkzeuge -> Board:"..." -> Boardverwalter und suche dort nach dem senseBox SAMD Boards-Package.

*Suche nach dem rot markierten Package*

Wenn ihr auf den Eintrag in der Liste klickt, erscheint dort ein Update-Button.

Wichtig ist, zuerst auf den Eintrag zu klicken. Ansonsten wird der Update-Button nicht angezeigt, auch wenn es bereits eine neue Version gibt.

1. Klicke auf diesen Button und gehe danach sicher, dass die installierte Version höher als 1.1.0 ist.

*Klicke auf 'Update', um das Board-Support-Package zu aktualisieren*

Da wir das senseBox SAMD Boards-Package für euch regelmäßig aktualisieren, solltet ihr immer mal wieder in den Boardverwalter gehen und nachschauen, ob das senseBox SAMD Boards-Package noch aktuell ist. Öffnet dafür wie oben beschrieben den Boardverwalter, sucht nach senseBox SAMD Boards und klickt dort ggf. auf `Update`.

## Schritt 1: Löschen der senseBox-Libraries aus dem Sketchbook-Folder

1. Öffne nun die Arduino IDE. Gehe auf `Datei` -> `Voreinstellungen`:



*Klicke 'Datei' -> 'Voreinstellungen'*

und schaue im Feld unter `Sketchbook-Speicherort` nach an welchem Ort der Sketchbook-Ordner gespeichert ist.



*Schaue im rot-markierten Feld nach, wo dein Sketchbook-Speicherort ist*

Merke dir den Pfad zu diesem Ordner, also den Ort wo dieser gespeichert ist.

1. Nun navigierst du in deinem Datei-Explorer zum Sketchbook-Speicherort (siehe 1.). Beachte, dass der Zielordner im Datei-Explorer am Sketchbook-Speicherort den Namen `Arduino` trägt. Öffne diesen Ordner. Innerhalb des `Arduino`-Ordners befindet sich ein Ordner mit dem Namen `libraries`. Innerhalb dieses Ordners befinden sich die senseBox-Libraries. Lösche den `libraries`-Ordner, um sie zu entfernen.

**Wenn du ein erfahrener Arduino-Nutzer bist und in der Vergangenheit weitere externe Libraries eingebunden hast, die nicht zu den senseBox-Libraries gehören, gehe in den 'libraries'-Ordner und lösche alle Libraries, die nicht extern von dir eingebunden wurden, anstatt den gesamten Ordner zu löschen.**

2. Schließe jetzt das Programm Arduino vollständig und starte es erneut, um das Löschen der alten senseBox-Libraries abzuschließen.

## Schritt 2: Neues Board-Support-Package einbinden

Um das neue Board-Support-Package einzubinden, geht ihr ähnlich vor, wie in den ersten Schritten, mit ein paar kleinen Änderungen.

1. Füge die folgende URL in deiner Arduino IDE unter Datei -> Voreinstellungen in das Feld für Zusätzliche Bordverwalter-URLs ein:

`https://github.com/sensebox/senseBoxMCU-core/raw/master/package_sensebox_index.json`

An der Stelle steht im Normalfall vorher schon folgende URL: `https://github.com/watterott/senseBox-`

MCU/raw/master/package\_sensebox\_index.json diese sieht der obigen sehr ähnlich, ist aber nicht die gleiche URL. Sie muss aber unbedingt durch die oben stehende URL ausgetauscht werden.



*Öffne die Voreinstellungen und füge die URL ein*

2. Öffne nun den Boardverwalter unter Werkzeuge -> Board:"..." -> Boardverwalter und suche dort nach dem senseBox SAMD Boards-Package.



*Suche nach dem rot markierten Package*

Wenn ihr auf den Eintrag in der Liste klickt, erscheint dort ein Update-Button.

Wichtig ist, zuerst auf den Eintrag zu klicken. Ansonsten wird der Update-Button nicht angezeigt, auch wenn es bereits eine neue Version gibt.

1. Klicke auf diesen Button und gehe danach sicher, dass die installierte Version höher als 1.1.0 ist.



*Klicke auf 'Update', um das Board-Support-Package zu aktualisieren*

Da wir das senseBox SAMD Boards-Package für euch regelmäßig aktualisieren, solltet ihr immer mal wieder in den Boardverwalter gehen und nachschauen, ob das senseBox SAMD Boards-Package noch aktuell ist. Öffnet dafür wie oben beschrieben den Boardverwalter, sucht nach senseBox SAMD Boards und klickt dort ggf. auf `Update`.

# Contributing

## Contributors-only

Hier findest du eine Sammlung von Vorlagen für das Styling, um an diesem Gitbook mitzuarbeiten.

---

Darunter eine [Vorlage für eine neue Seite](#)<sup>1</sup> mit Titlelement und Beschreibungsbox.

---

[Hint-Boxes](hint-template.md)[^2]:

>>INFO-HINT<<

>>SUCCESS-HINT<<

>>WARNING-HINT<<

>>ERROR-HINT<<

---

Bildunterschriften<sup>3</sup>:



*Dies ist eine Beispiel-Unterschrift*

Tabs<sup>4</sup>:

- [Erster Tab](#)
- [Zweiter Tab](#)
- [Dritter Tab](#)

## Erster Tab

Das ist der erste Tab.

Innerhalb der Tabs kann Markdown normal verwendet werden.

## Zweiter Tab

Das ist der zweite Tab.

Innerhalb der Tabs kann Markdown normal verwendet werden.

## Dritter Tab

Das ist der dritte Tab.

Innerhalb der Tabs kann Markdown normal verwendet werden.

---

Collapsibles<sup>5</sup>:

▼ [my list](#)

- list 1
- list 2
- list 3
  - list 3.1

---

1. Siehe [6.7.4 Neue Seite](#) ↵

2. Siehe [6.7.1 Hint-Box](#) ↵

3. Siehe [6.7.2 Bildunterschriften](#) ↵

4. Siehe [6.7.3 Tabs](#) ↵

5. Siehe [6.7.5 Collapsible](#) ↵



## Hint-box/Hinweis-Box:

Um Hinweise kenntlich zu machen kannst du eine unserer 4 Hinweis-Boxen benutzen. Achte darauf, dass es sich um HTML Tags handelt, das bedeutet, innerhalb des Tags muss auch HTML benutzt werden. Solange nur Text innerhalb der Boxen stehen soll ist das egal, solltest du aber Links, Bilder oder andere Elemente benutzen sollten diese HTML Elemente sein um später richtig dargestellt zu werden.

## Code:

```
<div class="box_info">
    <i class="fa fa-info fa-fw" aria-hidden="true" style="color: #42acf3;"></i>
    >>INFO-HINT<<
</div>

<div class="box_success">
    <i class="fa fa-check fa-fw" aria-hidden="true" style="color: #50af51;"></i>
    >>SUCCESS-HINT<<
</div>

<div class="box_warning">
    <i class="fa fa-exclamation-circle fa-fw" aria-hidden="true" style="color: #f0ad4e;"></i>
    >>WARNING-HINT<<
</div>

<div class="box_error">
    <i class="fa fa-exclamation-triangle fa-fw" aria-hidden="true" style="color: #d9534f;"></i>
    >>ERROR-HINT<<
</div>
```

## Result:

>>INFO-HINT<<

>>SUCCESS-HINT<<

>>WARNING-HINT<<

>>POSSIBLE-ERROR-HINT<<



## Bildunterschriften

Um eine Bildunterschrift zu erstellen füge einfach einen Titel zum Bild hinzu, dieser wird dann automatisch als Bildunterschrift dargestellt.

---

Um eine Bildunterschrift zu erstellen füge einfach einen Titel zum Bild hinzu, dieser wird dann automatisch als Bildunterschrift dargestellt.

### Bild mit Unterschrift:

```
![Das LoraBee passend zur SenseBox!](../../../../../pictures/LoraBee%20top.png)
```



*Das LoraBee passend zur SenseBox!*

### Bild ohne Unterschrift:

```

```



## Tabs

Tabs helfen dir wenn du Textabschnitte auf verschiedene Grundbedingungen seitens des Nutzers anpassen willst. Ist ein Textabschnitt zum Beispiel abhängig von dem Betriebssystem, welches der Nutzer verwendet, so kannst du Tabs benutzt um dem Nutzer die Ansicht auf das für ihn relevante zu reduzieren.

Um Tabs zu benutzen, verwende den Tabs-Tag:

```
{% tabs first="Erster Tab", second="Zweiter Tab", third="Dritter Tab" %}  
{%- content "first" %}  
# Erster Tab  
Das ist der erste Tab.  
> Innerhalb der Tabs kann Markdown normal verwendet werden.  
  
{%- content "second" %}  
# Zweiter Tab  
Das ist der zweite Tab.  
> Innerhalb der Tabs kann Markdown normal verwendet werden.  
  
{%- content "third" %}  
# Dritter Tab  
Das ist der 3. Tab.  
> Innerhalb der Tabs kann Markdown normal verwendet werden.  
  
{%- endtabs %}
```

- [Erster Tab](#)
- [Zweiter Tab](#)
- [Dritter Tab](#)

## Erster Tab

Das ist der erste Tab.

Innerhalb der Tabs kann Markdown normal verwendet werden.

## Zweiter Tab

Das ist der zweite Tab.

Innerhalb der Tabs kann Markdown normal verwendet werden.

## Dritter Tab

Das ist der dritte Tab.

Innerhalb der Tabs kann Markdown normal verwendet werden.



# Überschrift

Hier steht eine einleitende Beschreibung der Seite/des Themas.

## Weitere Überschrift

Inhalt der Seite ...

### Markdown Code der Vorlage

```
#Überschrift{#head}

<div class="description">
    Hier steht eine einleitende Beschreibung der Seite/des Themas.
</div>
<div class="line">
    <br>
    <br>
</div>

##Weitere Überschrift
Inhalt
der
Seite
...
```

# Collapsible

Um die Übersicht zu verbessern und Text nur dann zu zeigen, wenn er dem User auch nutzt/interessiert kannst du sogenannte "Collapsibles" erstellen. Sieh dir an wie sie funktionieren und was du damit machen kannst.

Collapse Modul ohne Titel:

```
{% collapse %}

* list 1
* list 2
* list 3
    * list 3.1

{% endcollapse %}
```



- list 1
- list 2
- list 3
  - list 3.1

Modul mit Titel:

```
{% collapse title="my list" %}

* list 1
* list 2
* list 3
    * list 3.1

{% endcollapse %}
```



- list 1
- list 2
- list 3
  - list 3.1

```
{% collapse title="my non-markdown list", process=false %}

* list 1
* list 2
* list 3
    * list 3.1

{% endcollapse %}
```

Modul in dem Markdown nicht kompiliert wird:

► my non-markdown lis