

# senseBox:home

## Dokumentation



---

# Inhaltsverzeichnis

Einleitung	1.1
Inventar & Grundaufbau	1.2
Softwareinstallation	1.3
openSenseMap Registrierung	1.4
senseBox:home Feinstaub-Erweiterung	2.1
Solarbetriebene senseBox:home	2.2
senseBox:home mit ESP8266	2.3
Softwareinstallation	2.3.1
Anpassen des Sketches	2.3.2
senseBox mit Wemos D1	2.3.3
Alle weiteren ESP8266	2.3.4
Beispielaufbauten	3.1
senseBox fibox	3.1.1
Botanischer Garten	3.1.2

---



## senseBox:home

Die senseBox:home ist ein Citizen Science DIY - Toolkit für die ortsbezogene Messung von Umweltdaten wie Temperatur, Luftfeuchte, Luftdruck, Beleuchtungsstärke und UV-Licht. Sie basiert auf der Arduino/Genuino Plattform und kann einfach in unsere Sensorweb-Plattform [openSenseMap<sup>1</sup>](https://opensensemapper.org) integriert werden, wo sie kontinuierlich Messdaten liefert.

Auf diesen Seiten - welche auch als [PDF<sup>2</sup>](#) verfügbar sind - befindet sich die Dokumentation und Aufbauanleitung zur senseBox:home.

Wie die Sensorstation programmiert wird, ist in den folgenden Kapiteln beschrieben:

Bei Fragen zum Aufbau wende dich bitte [per Mail<sup>3</sup>](mailto:support@sensebox.de) an uns. Das senseBox Team wünscht viel Spaß mit deiner Do-It-Yourself Sensorstation!

---

Die senseBox ist ein Open Source Projekt und befindet sich ständig in der Weiterentwicklung. Das heißt, dass auch diese Seiten nach und nach erweitert werden. Falls euch Fehler auffallen oder ihr bei der Entwicklung einsteigen wollt, meldet euch gerne bei uns [per Mail<sup>4</sup>](mailto:info@sensebox.de) oder öffnet ein Issue auf [GitHub<sup>5</sup>](https://github.com/sensebox/books/issues)!

Warnhinweise:

- Durch elektrostatische Entladung können die Bauteile beschädigt oder sogar zerstört werden! Daher solltet ihr euch z.B. an einem Heizungsrohr entladen, bevor ihr mit dem Aufbau anfängt.
  - Elektronische Bauteile und Leiterplatten können Chemikalien enthalten. Daher solltet ihr nach dem Aufbau oder Gebrauch euch die Hände waschen.
  - Elektronik sollte umweltfreundlich entsorgt werden und bei Sammelstellen abgegeben werden.
- 

<sup>1</sup>. <https://opensensemapper.org> ↵

<sup>2</sup>. [https://sensebox.github.io/books/senseBox:home\\_de.pdf](https://sensebox.github.io/books/senseBox:home_de.pdf) ↵

<sup>3</sup>. <mailto:support@sensebox.de> ↵

<sup>4</sup>. <mailto:info@sensebox.de> ↵

<sup>5</sup>. <https://github.com/sensebox/books/issues> ↵

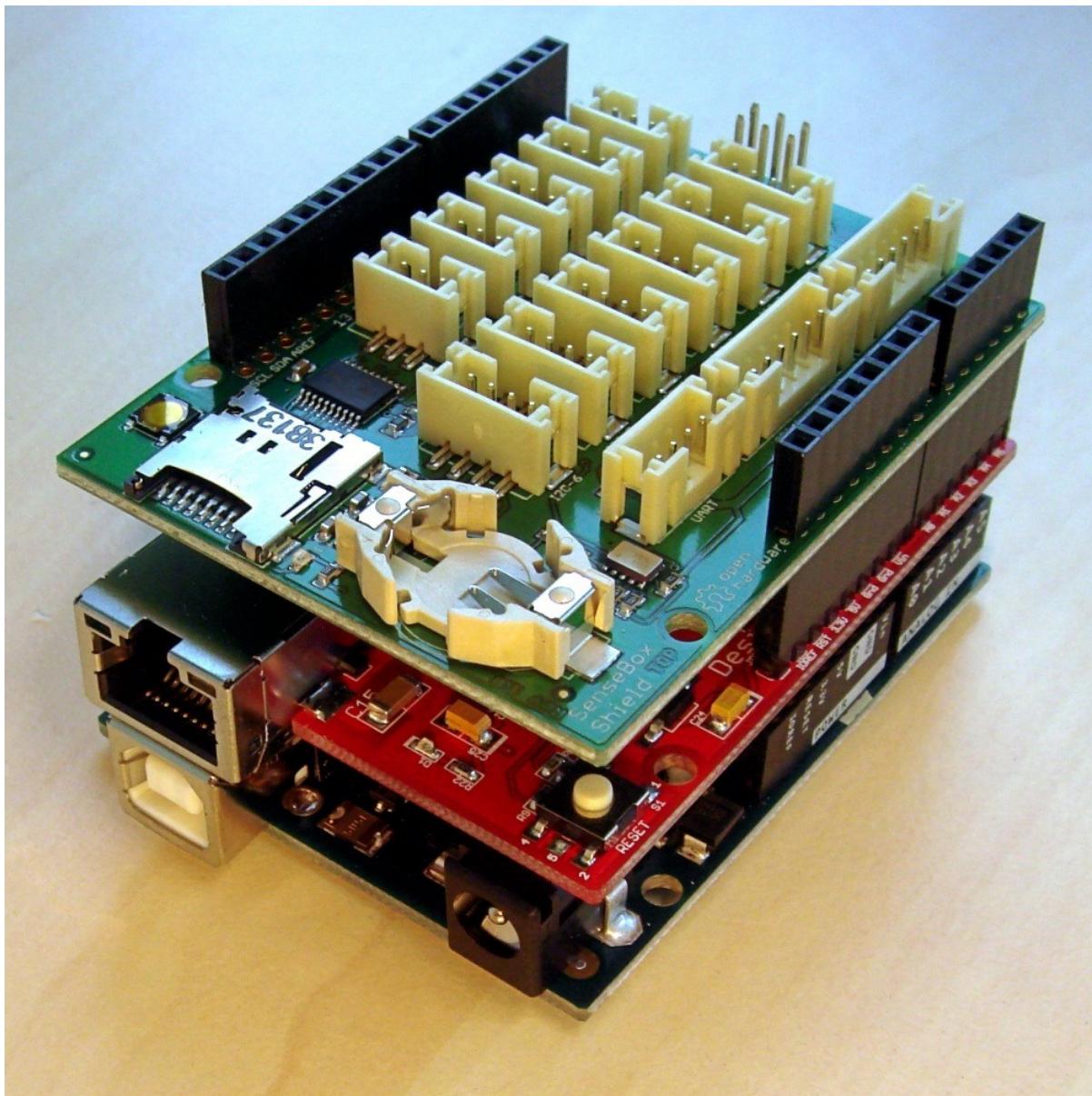
## Inventarliste

Bevor es los geht solltet ihr überprüfen ob alle Bauteile vorhanden sind.

### Inhalt der senseBox

Basisstation bestehend aus drei Platinen

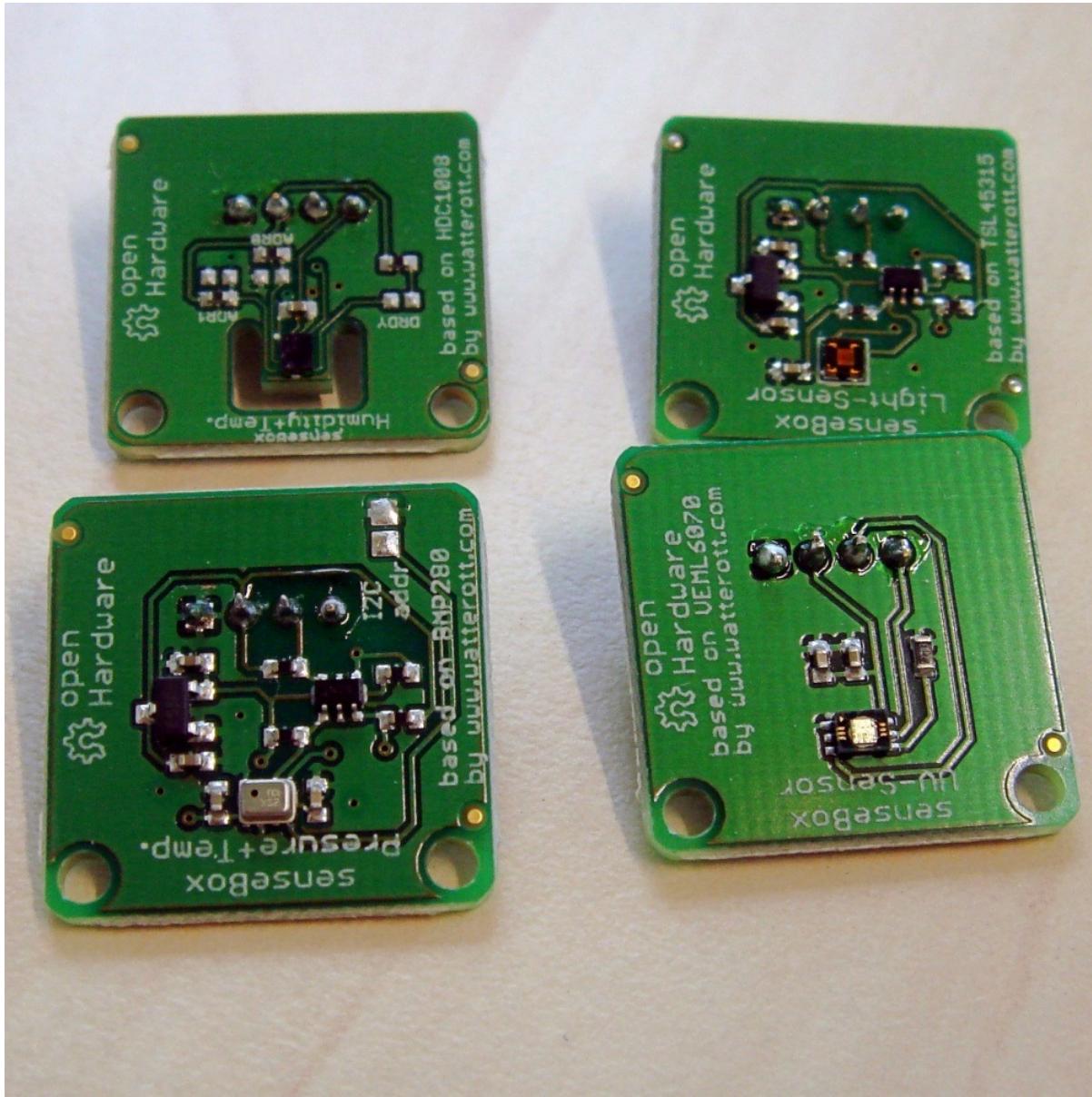
Die senseBox:home ist in zwei Ausgaben verfügbar: Einmal mit LAN-, und einmal mit WLAN-Netzwerkverbindung. Je nach Ausgabe ist ein W5500 Ethernet Shield, oder ein Watterott WLAN-Shield enthalten.



Platine	Beschreibung
Genuino Uno (unten)	Liest die angeschlossenen Sensoren aus und überträgt die Messungen ins Internet

W5500 Ethernet Shield oder Watterott WLAN-Shield (mitte)	Ist für die Internetverbindung zuständig
senseBox Shield (oben)	Hier werden die Sensoren angeschlossen

### Grundausstattung mit vier Sensoren



Sensor	Beschreibung
HDC1008	Temperatur in Grad Celsius (°C) und relative Luftfeuchte in Prozent (%)
BMP280	Luftdruck in Pascal (pa)
TSL45315	Beleuchtungsstärke des sichtbaren Lichts in Lux (lx)
VEML6070	Intensität der ultravioletten Strahlung in Mikrowatt pro Quadratcentimeter ( $\mu\text{W}/\text{cm}^2$ )

### Anschlusskabel für Sensoren und USB-Verbindung

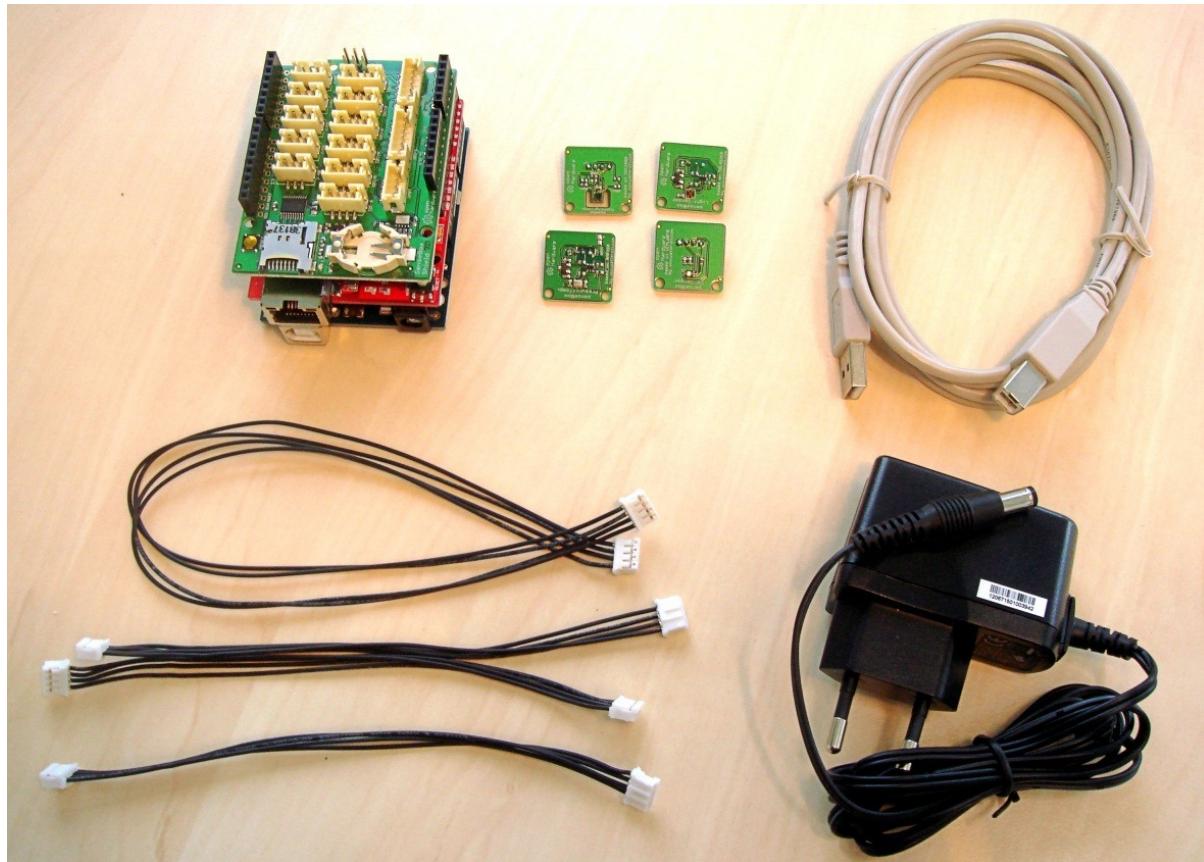
- 1x USB-Kabel für den Anschluss des Mikrocontrollers an den Computer

- 1x langes Verbindungskabel für kombiniertes Thermo- bzw. Hygrometer
- 3x kurzes Verbindungskabel für Barometer, Luxmeter und UV-Lichtsensor

## Netzteil

- 9V Netzteil (670mA)

## Gesamtüberblick:



## Zusätzliche Materialien (NICHT im Lieferumfang enthalten)

- LAN-Kabel für den Anschluss der senseBox an euren Router, falls die senseBox:home LAN vorliegt
- Gehäuse für eine wetterfeste Installation der Elektronik
- Werkzeuge für den Aufbau wie z.B. Heißklebepistole

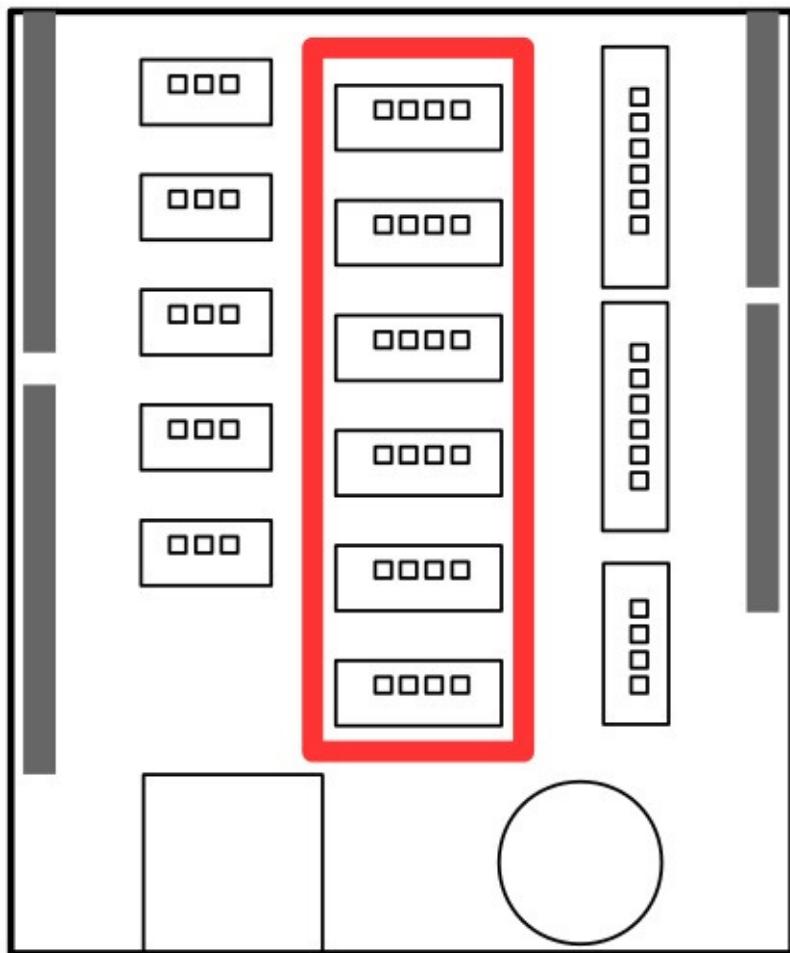
---

## Aufbau der senseBox

Du kannst deine Messstation kann in wenigen Schritten zusammenbauen.

Die senseBox wird entweder über das USB-Kabel oder über das Netzteil mit Strom versorgt. Für den temporären Betrieb wird das Netzteil also nicht benötigt.

Im Bausatz der senseBox:home befinden sich vier kleinen Platinen mit den Sensoren. Die eigentlichen Sensoren sind nur wenige Millimeter groß und befinden sich auf der Oberseite der Platinen. Um einer Beschädigung vorzubeugen, solltest du die kleinen Sensoren nicht berühren, sondern die Platinen nur am Rand anfassen. Der Anschluss der Sensoren ist denkbar einfach: Benutze die Verbindungskabel, um die Sensoren mit den mittleren Steckplätzen auf der Basisstation zu verbinden. Welcher Anschluss dabei gewählt wird spielt keine Rolle.



Das lange Verbindungskabel ist für den HDC1008 gedacht, um ihn außerhalb eines Gehäuses anbringen zu können!

## Treiber und Softwareinstallation

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.

Falls etwas bei dem Testlauf schief geht, melde dich am besten [per Mail<sup>1</sup>](#) bei unserem Support.

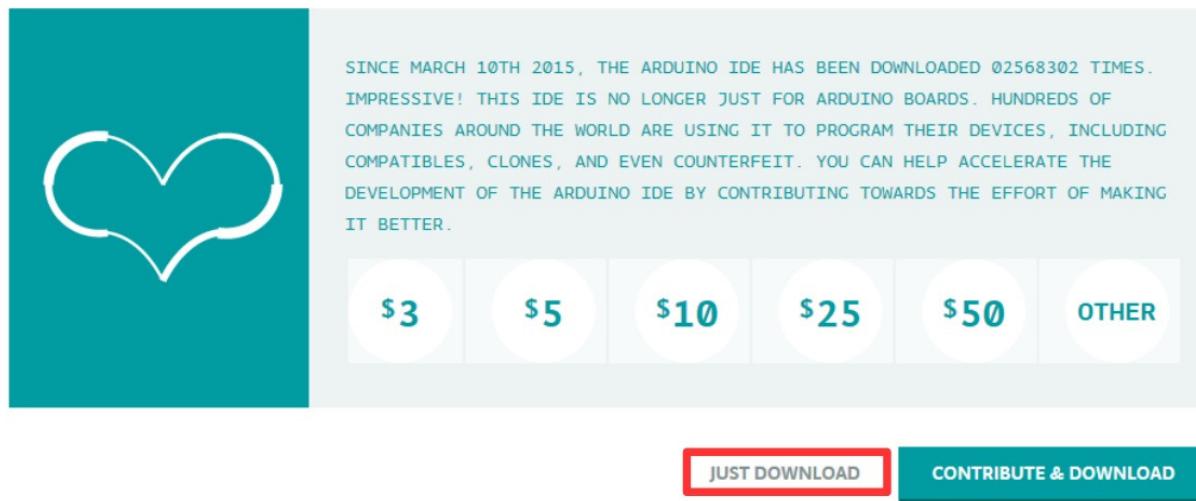
## Arduino Software herunterladen

Für einen reibungslosen Ablauf bitte Arduino 1.8.3 oder höher nutzen.

Das Mainboard der senseBox ist eine modifizierte Version des Arduino Uno Mikrocontrollers. Um ein Programm auf das Board zu laden, brauchst du die integrierte Entwicklungsumgebung von Arduino, kurz Arduino IDE. Lade die neueste Version als zip-Datei von der [Arduino Homepage<sup>2</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 starten der Datei `arduino.exe` kann die IDE gestartet werden.

## 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 die folgenden Terminal-Befehle:

```
tar -xvf arduino-1.8.3-linux64.tar.xz
cd arduino-1.8.3
./install.sh
```

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:

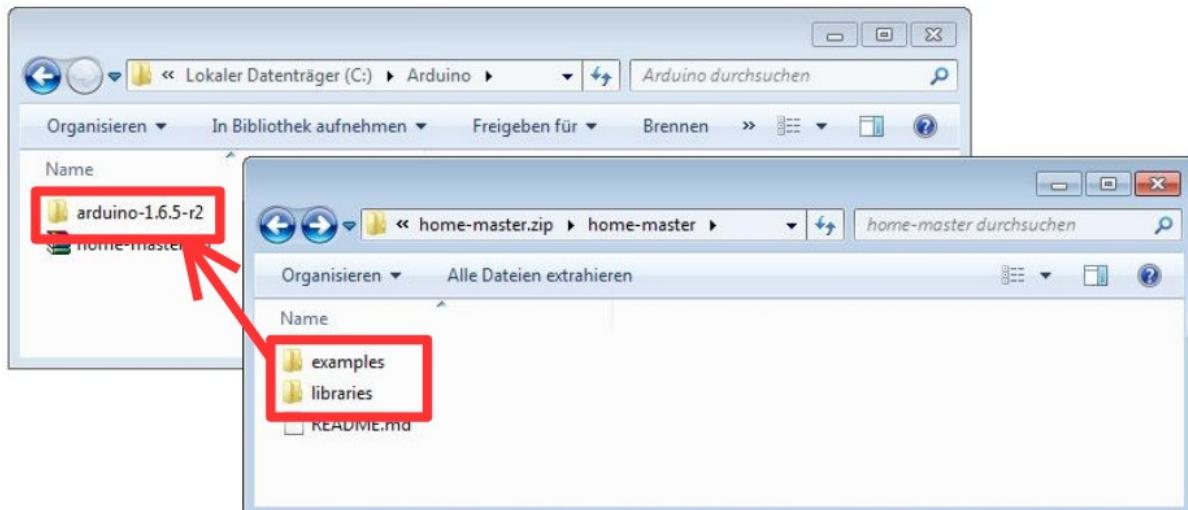
```
sudo usermod -a -G dialout $(whoami)
sudo chmod a+r /dev/<device-id>
```

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

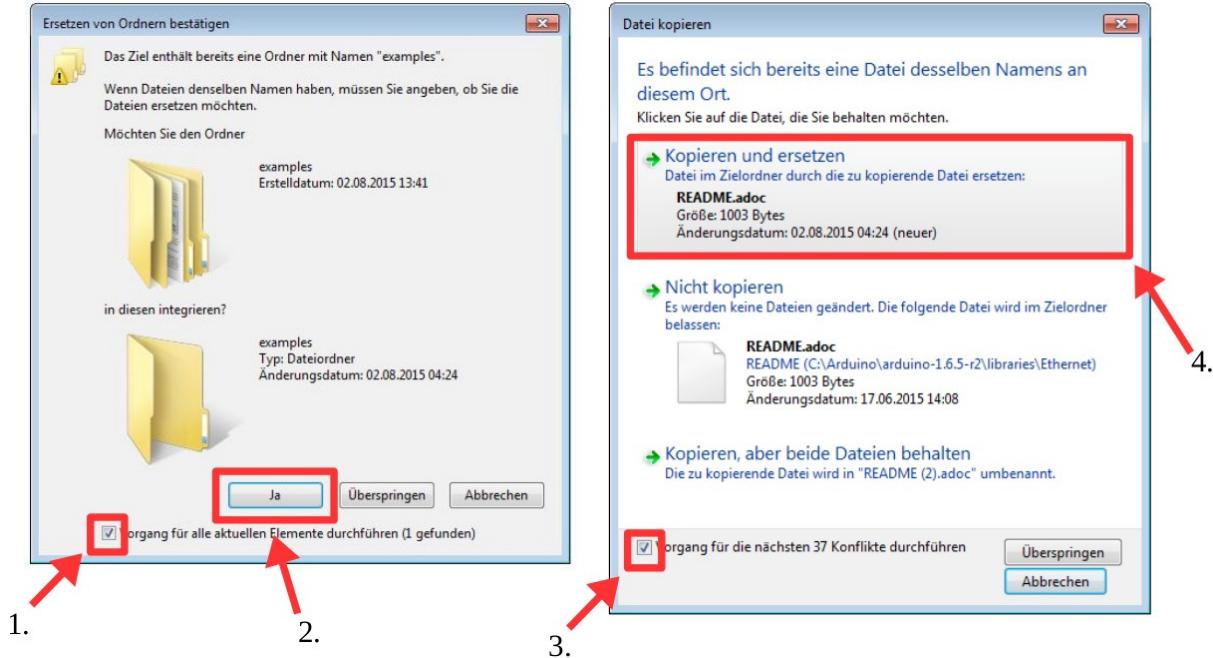
## Arduino Bibliotheken installieren

Um die Sensoren und die Netzwerkkarte nutzen zu können, müssen noch ein paar Bibliotheken installiert werden. Ein zip-Archiv mit allen benötigten Bibliotheken findest du [hier](#)<sup>3</sup>.

Lade dieses zip-Archiv herunter und integriere nun die beiden Ordner `examples` und `libraries` aus dem Archiv in deinen Arduino Ordner. Wenn ihr gefragt werdet ob bestehende Dateien ersetzt werden sollen, folge den Anweisungen unten auf der Seite.



Setze nun wie unten dargestellt im ersten Dialogfeld den Haken unten und bestätigt mit „Ja“. Daraufhin öffnet sich ein neues Fenster, in dem du wieder den Haken setzt, und „Kopieren und ersetzen“ auswählst.



Das folgende Video zeigt den Kopiervorgang noch einmal im Detail:

Eingebettetes Video: <https://www.youtube.com/watch?v=j-hdRJp2o4k>

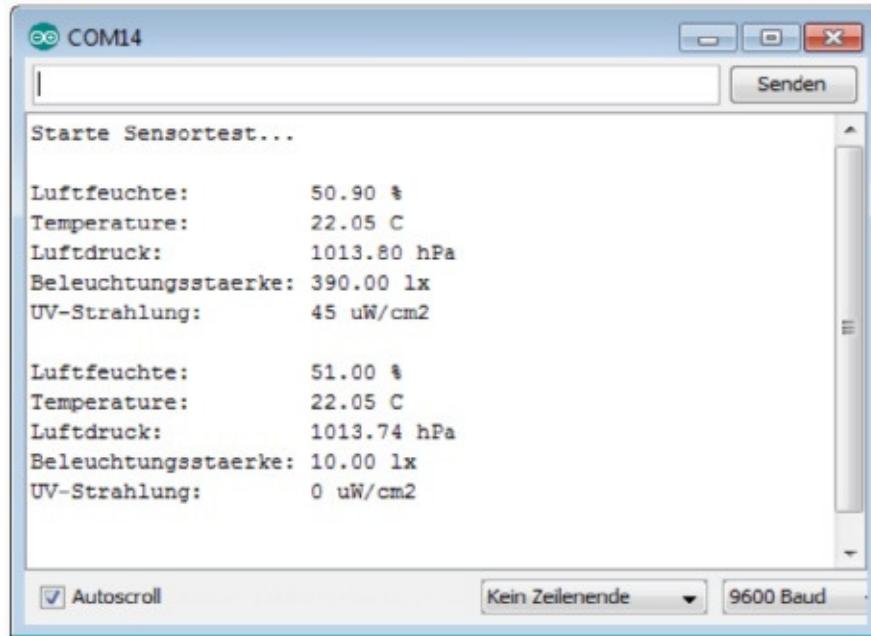
## Sensoren testen

Die Sensoren der senseBox:home können nun auf funktionstüchtigkeit und getestet werden. Hierzu gibt es einen Sketch, welcher mit den zuvor Arduino-Bibliotheken installiert wurde. Voraussetzung für den test ist natürlich, dass die Sensoren mit dem Arduino verbunden wurden ;)

1. Arduino Anwendung starten
  - Es kann sein, dass nach dem Start eine Meldung über neue Updates erscheint. Fall du die Version 1.6.5 oder höher installiert hast, kannst du dies überspringen.
2. unter Werkzeuge → Board das Arduino Uno auswählen
3. unter Werkzeuge → COM-Port den entsprechenden Anschluss wählen
  - Falls mehrere Auswahlmöglichkeiten angezeigt werden, musst du zuerst den richtigen COM-Port im Geräte Manager finden, oder alle Ports ausprobieren.

Ladet nun das Programm, um die Sensoren zu testen und überträgt es auf die Messstation:

1. in der Menüleiste Datei → Beispiele → senseBox → \_01\_sensor\_test auswählen
2. das Programm über das Pfeil Icon auf den Mikrocontroller laden
3. warten bis das Programm übertragen wurde
4. den seriellen Monitor über das Lupe Icon öffnen



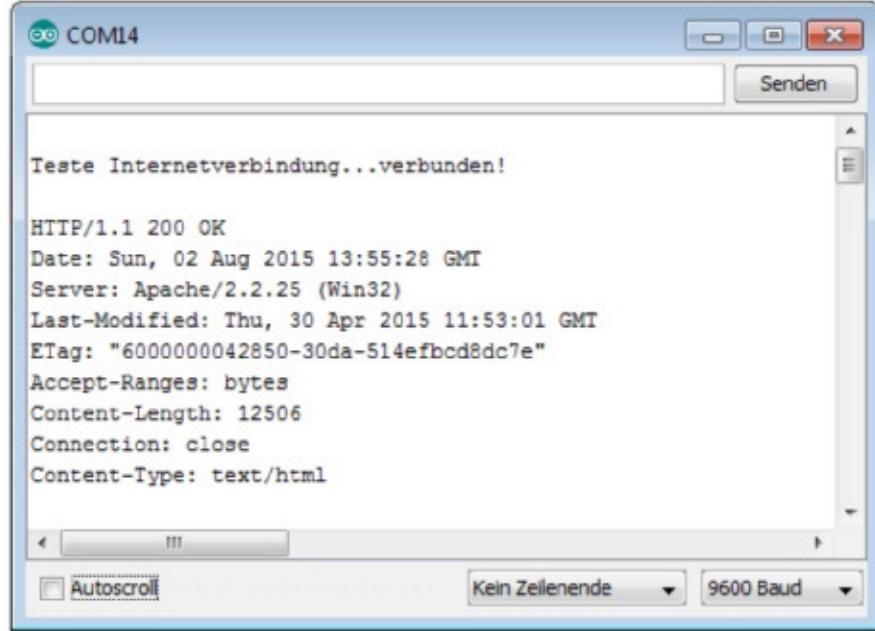
Du kannst durch experimente prüfen, ob sich die gemessene Temperatur, Luftfeuchtigkeit oder Beleuchtungsstärke verändern. Der Luftdruck lässt sich nicht ohne weiteres beeinflussen. Er sollte grob, je nach Höhenlage und Wetterverhältnissen, zwischen 600 hPa und 1000 hPa liegen. Die Intensität des UV-Lichts kann nur mit speziellen Lampen oder durch direkte Sonneneinstrahlung getestet werden. In einem geschlossen Raum sollte keine bzw. nur minimale UV-Strahlung gemessen werden können.

## Verbindung zur openSenseMap testen

Nach dem selben Schema kann noch die Internetverbindung der senseBox:home getestet werden:

1. den seriellen Monitor (Fenster mit den Messwerten) schließen
2. ein Netzwerkkabel von eurem Heimnetzwerk mit der senseBox verbinden
3. in Menüleiste Datei → Beispiele → senseBox → \_02\_network\_test auswählen
4. das Programm über das Pfeil Icon auf den Mikrocontroller laden
5. den seriellen Monitor über das Lupe Icon starten

Wenn die Verbindung klappt, bekommst du eine entsprechende Meldung im seriellen Monitor angezeigt.



- 
- 1. mailto:support@sensebox.de ↵
  - 2. <https://www.arduino.cc/en/main/software> ↵
  - 3. [https://github.com/sensebox/resources/raw/master/libraries/senseBox\\_Libraries.zip](https://github.com/sensebox/resources/raw/master/libraries/senseBox_Libraries.zip) ↵

## Web-Integration

Hier wird die Einbindung der senseBox in unser Sensornetzwerk durch die Registrierung auf der openSenseMap beschrieben.

### Registrierung auf der openSenseMap

Unter <https://opensesemap.org/register> kann eine neue Sensorstation für das openSenseMap Sensornetzwerk registriert werden. Während der Registrierung wirst du nach einem Hardware Setup gefragt. Wähle dort die „senseBox:home“ aus und setze danach je nach Variante den Haken bei „senseBox:home (Ethernet)“ oder „senseBox:home (WLAN)“.

Ein Software-Programm für einen Arduino Mikrocontroller ist an der Dateiendung `.ino` zu erkennen. Eine solche Datei wird benötigt, um eure senseBox mit der openSenseMap im Internet zu verbinden. Den passenden Sketch bekommst du zusammen mit einer E-Mail zugeschickt, wenn die Station auf der openSenseMap registriert wurde.

### Programm auf die Station laden

Nachdem du den `.ino` Anhang der Email heruntergeladen hast, muss dieses Programm auf deine senseBox geladen werden. Wie man genau ein Programm auf den Mikrocontroller lädt, ist [hier<sup>1</sup>](#) ausführlich erklärt worden. Hier die Schritte in der Übersicht:

- Arduino Anwendung öffnen
- In der Menüleiste `Datei` → `Öffnen` auswählen und die `sensebox.ino` Datei auswählen
- Dialog mit "Ja" bestätigen
- Das Programm über das Pfeil Icon auf den Mikrocontroller laden
- Warten bis das Programm übertragen wurde

Wenn alles richtig gelaufen ist, kannst du nun auf der openSenseMap deine Station auswählen und verfolgen, wie Messungen kontinuierlich übertragen werden!

---

1. [https://books.sensebox.de/books/osem/de/osem\\_registrierung.html](https://books.sensebox.de/books/osem/de/osem_registrierung.html) ↵

1. Siehe [1.3 Softwareeinstallation](#) ↵

## Feinstaub-Erweiterung für die senseBox:home

In dieser Anleitung wird das Anschließen, Aufbauen und Registrieren der Feinstaub-Erweiterung für die senseBox:home beschrieben. Die Feinstaub-Erweiterung ergänzt die senseBox:home um die Möglichkeit, Feinstaubmessungen durchzuführen. Dazu wird ein SDS011 Sensor von Inovafit genutzt, welcher die Feinstaubkonzentration in den Messgrößen PM10 (Partikel kleiner 10 Mikrometer) und PM2.5 (Partikel kleiner 2,5 Mikrometer) ermöglicht. 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. Weitere Informationen findet ihr auf [dieser Seite](#)<sup>1</sup>.

### Benötigte Bauteile:

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



### Registrierung auf der openSenseMap

- Option a: Bestehende senseBox:home Falls ihr eure senseBox:home mit einem Feinstaubsensor erweitern wollt, müsst ihr auf der [openSenseMap](#)<sup>2</sup> die Einstellungen eurer Station bearbeiten. Dies ist nach dem Login im Dashboard unter senseBox verwalten möglich. Nun könnt ihr unter dem Reiter Erweiterungen die Feinstaub-Erweiterung aktivieren. Nachdem die Einstellungen gespeichert wurden (oben rechts), erhaltet ihr einen aktualisierten Sketch per Email. Dieser muss anschliessend auf die senseBox hochgeladen werden!

- Option b: Neue senseBox:home mit Feinstaub-Erweiterung Falls ihr eine neue senseBox:home inklusive Feinstaub-Erweiterung registrieren wollt, aktiviert bei der Registrierung auf der [openSenseMap<sup>3</sup>](#) unter senseBox:Erweiterungen die Feinstaub-Erweiterung. Ihr bekommt darauf hin eine Mail mit einem angepassten Sketch zugeschickt.

## Aktualisierung der Arduino Software

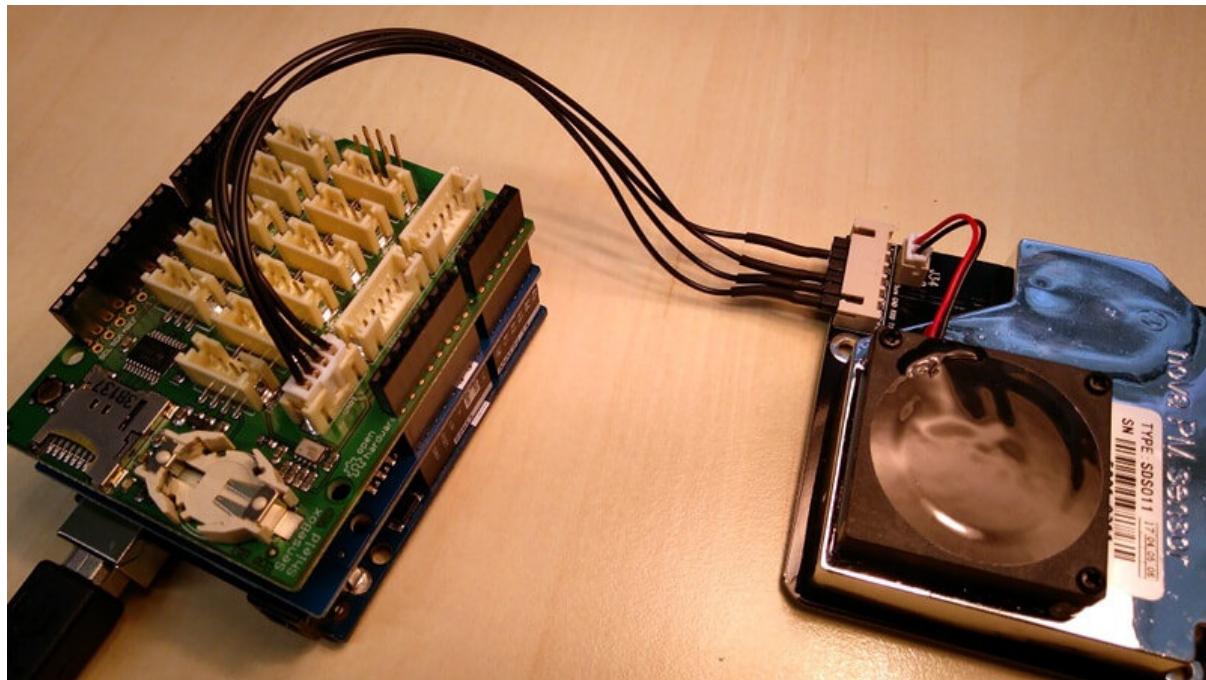
Um den Sensor korrekt auszulesen muss der erhaltene Arduino-Sketch auf die senseBox geladen werden.

Hinweis: Zur Kommunikation zwischen dem SDS011 und dem Arduino/Genuino Mikrokontroller wird eine einfache serielle Datenverbindung aufgebaut. Diese serielle Schnittstelle wird auch genutzt, um über die USB Verbindung mit dem Computer zu kommunizieren. Daher dürfen die digitalen Pins 0 (RX) und 1 (TX), bzw. der UART Steckplatz auf dem senseBox Shield nicht verbunden sein, wenn ein Sketch über die Arduino IDE auf das Arduino/Genuino Board geladen wird!

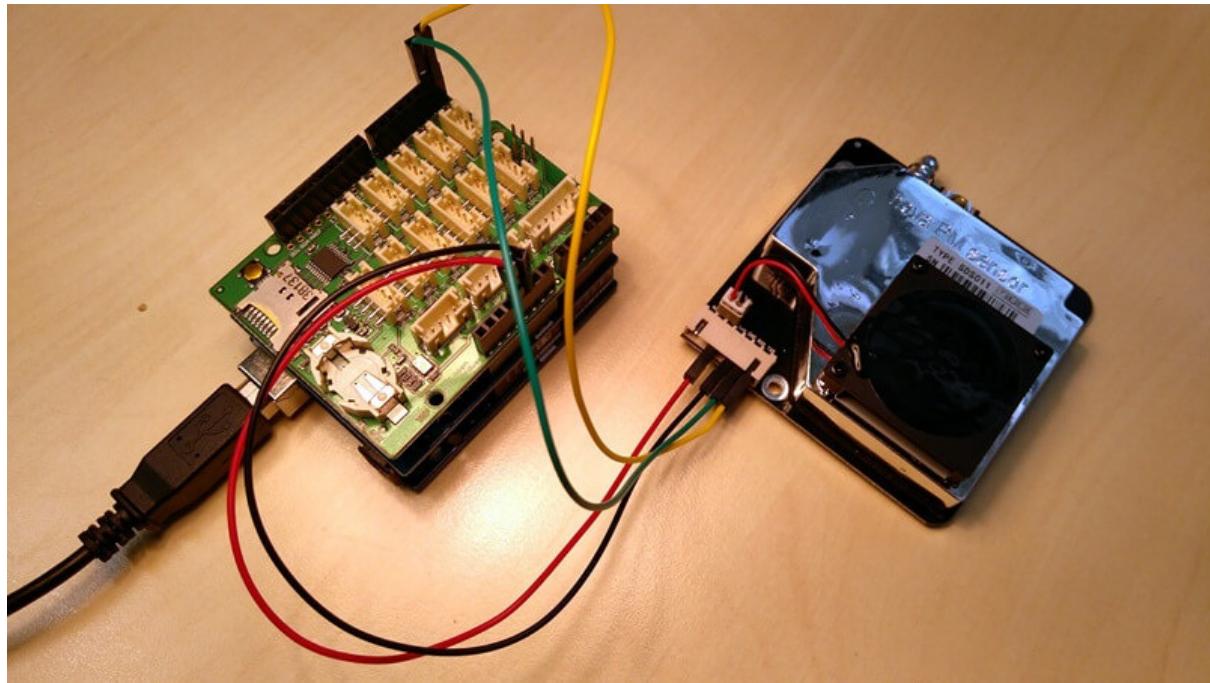
1. Installiert zunächst die Bibliothek für den Feinstaubsensor. Ladet dazu [dieses zip-Archiv<sup>4</sup>](#) herunter und kopiert den Inhalt in euer arduino/libraries Verzeichnis.
2. Startet nun die Ardunio IDE neu und öffnet den soeben erhaltenen Sketch.
3. Passt den Sketch für eure Netzwerkverbindung an. Dazu gehört das Eintragen von Wifi Name und Passwort oder statische IP.
4. Trennt die Kabelverbindung zwischen dem Feinstaubsensor und dem Arduino/Genuino Board und überträgt den Sketch auf das Board. Als letztes schließt ihr das Kabel des Sensors wieder an.

## Anschluss Feinstaubsensor

Dieser Abschnitt kann übersprungen werden, falls das passende Kabel Feinstaubsensor ↔ senseBox Shield vorliegt.

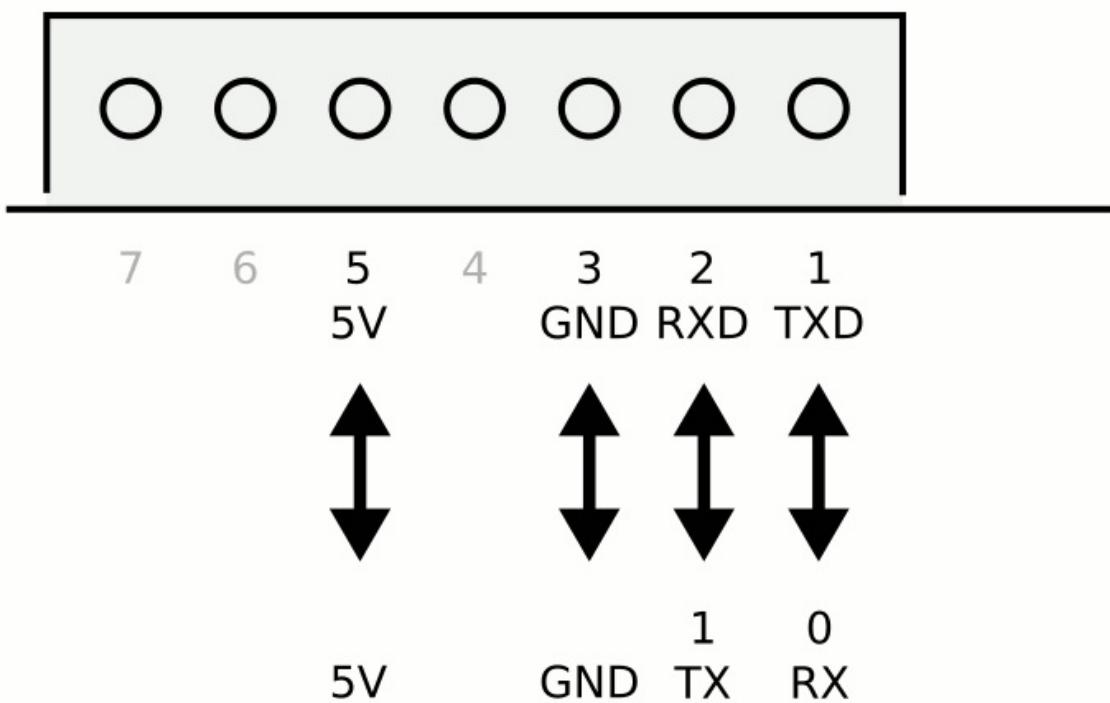


Der Feinstaubsensor muss mit vier Kabeln mit dem Arduino verbunden werden. Dafür können ganz normale Dupont Kabel mit jeweils einem Stecker und einer Buchse verwendet werden.



Die 5 Volt (5V) und GND Pins des SDS011 werden mit den gleichen Pins am Arduino verbunden. Danach werden die Pins TxD und RxD gekreuzt (RX ↔ TX) an die Pins 0 (RX) und 1 (TX) am Arduino angeschlossen.

## SDS011



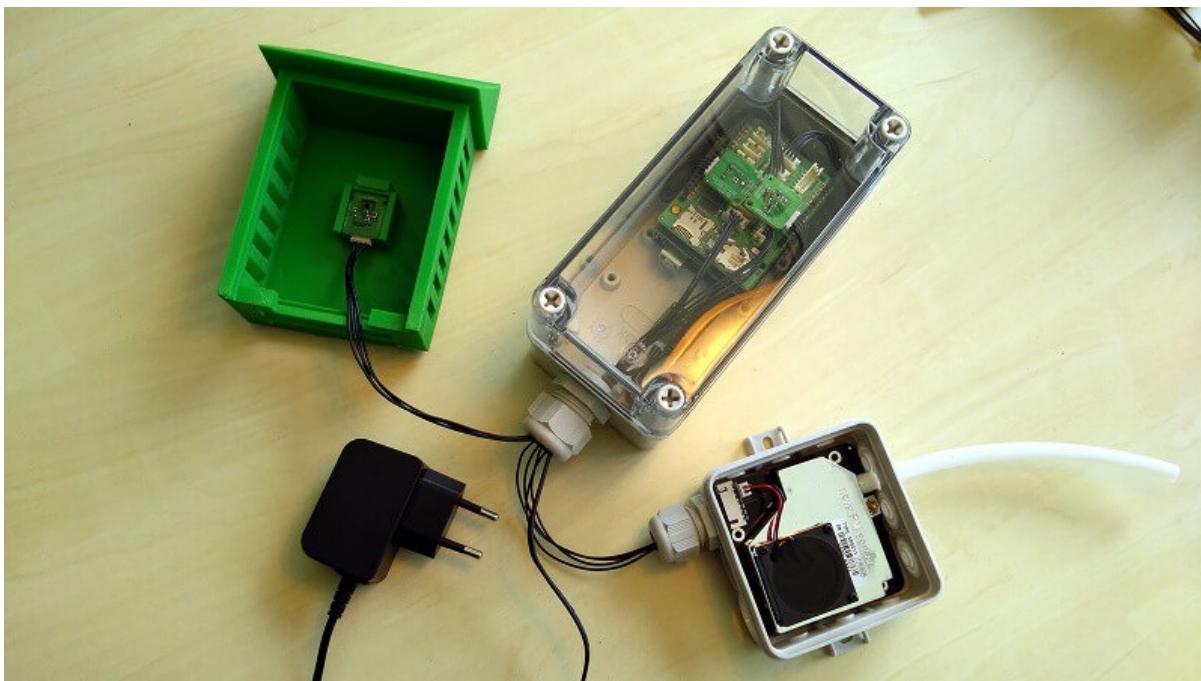
## Arduino

Gehäuse

Als Gehäuse wird hier eine einfache Verteilerdose verwendet. Es müssen jeweils Löcher für den Ansaugschlauch (8mm), Luftauslass (16mm) und Kabelverschraubung (16mm) gebohrt werden.



Jetzt legt ihr den Sensor in das Gehäuse und schließt den Teflonschlauch und das Kabel an. Am Ende sollte alles so aussehen wie in dem unteren Bild:



Viel Spaß beim Messen und Forschen!

---

1. <http://aqicn.org/sensor/sds011/de/> ↵

2. <https://opensesem.org/> ↵

3

- | 3. <https://opensesemap.org/account/register> ↵
- | 4. <https://github.com/sensebox/SDS011-select-serial/archive/master.zip> ↵

# Solarbetriebene senseBox:home

Hinweis: Diese Anleitung richtet sich an Fortgeschrittene und beschreibt das Thema keinesfalls allumfassend. Die hier beschriebenen Methoden und vorgestellte Schaltung dienen nur als erster Anhaltspunkt und sind an vielen Stellen beliebig optimierbar!

Die handelsübliche senseBox:home ist darauf ausgelegt, ständig durch ein Netzteil mit Strom versorgt zu werden. Durch Änderungen im Code und an der Hardware lassen sich die Messwerte jedoch auch vom Stromnetz getrennt und autark an die openSenseMap übertragen. Handelsübliche Solarladeregler für LiPo Akkus helfen dabei, eine Solarzelle und einen Akku mit einem Microcontroller zu verbinden.

## Umfang dieser Anleitung

Die Anleitung befasst sich zuerst mit den benötigten zusätzlichen Bauteilen und wie diese zusammengebaut werden müssen. Dies wird danach beispielhaft für die senseBox:home und senseBox:home mit ESP8266 Codebeispielen behandelt. Die besten Ergebnisse werden mit einer fertigen und funktionierenden senseBox:home mit ESP8266 oder Arduino Nano erzielt. Der Arduino Uno hat einen vergleichsweise hohen Stromverbrauch. Es funktioniert, führt aber dazu, dass der Akku schneller leer ist.

Achtung: Bitte die Anleitung vor dem Umsetzen einmal aufmerksam durchlesen. An mehreren Stellen muss über bestimmte Details der Umsetzung entschieden und dementsprechend andere Schritte durchgeführt werden.

## Benötigte Teile und Werkzeuge

Die hier verlinkten Shops sind als Beispiele anzusehen. Die benötigten Teile und Werkzeuge können dort beschafft werden, wo sie am günstigsten sind/am schnellsten lieferbar sind etc.

### Werkzeuge und Verbrauchsmaterialien

- Seitenschneider
- Lötkolben, Lötzinn
- Drähte/Litzen
- Alte USB-Kabel passend zum Microcontroller

### Teile

- Akkuladeregler
- Solarzelle passend zum Laderegler (siehe Absatz [Auswahl des Ladereglers, Solarzelle und Akku](#)<sup>1</sup>)
- LiPo Akku 3.7V mind 2000mAh, besser mehr
  - entweder 18650 Bauweise mit Batteriehalter
  - oder mit JST Stecker
- LiPo Safe

### Auswahl des Ladereglers, Solarzelle und Akku

Achtung: Es ist sehr wichtig darauf zu achten, passende Laderegler und Solarzellen zu verwenden. Unpassende Kombinationen führen zur Zerstörung vom Laderegler oder der Solarzelle oder beidem.

Bei der Auswahl des Ladereglers sollte gleichzeitig auch die Wahl der Solarzelle beachtet werden. Solarzellen gibt es in vielen verschiedenen Ausführungen. Für diese Anleitung sind Solarzellen mit etwa 5 Volt Ausgangsspannung interessant. Meist ist die Beschaffung der Solarzelle schwieriger, deswegen sollte zuerst eine Solarzelle ausgesucht werden und dann passend dazu ein Laderegler.

## Solarzelle

Wie schon erwähnt, sollte die Solarzelle etwa 5 bis 6 Volt Ausgangsspannung liefern. Eine Ausgangsleistung von etwa 3-4 Watt ist ausreichend, kann aber je nach Exponiertheit zu viel oder zu wenig sein. Teilweise werden Solarzellen bereits mit Spannungsregulierung verkauft. Dies sind oft Zellen mit USB Ausgang. Diese können genauso verwendet werden wie Solarzellen, die noch mit Kabeln versehen werden müssen. Im Abschnitt [Aufbau](#)<sup>432</sup> wird eine Solarzelle von Ebay ohne Kabel verwendet.

## Laderegler

Nachdem eine passende Solarzelle ausfindig gemacht wurde, kann sich der Auswahl eines Ladereglers gewidmet werden. In jedem Fall sollten sich Ausgangsleistung und Spannung der Solarzelle innerhalb der erlaubten Eingangsspannung und Leistung des Ladereglers befinden. Diese können den Datenblättern entnommen werden. Hierbei ist darauf zu achten, dass die Ausgangsspannung des Ladereglers meist bei 5 Volt liegt. Diese Spannung sollte dem Microcontroller unbedingt über den USB Port zugeführt werden.

Die folgende Liste enthält Vorschläge zu Laderegbern. Im Abschnitt [Aufbau](#) wird ein Laderegler des Typs TP4056 verwendet.

- TP4056 TE420 ([1](#), [2](#), [3](#))
- Adafruit USB / DC / Solar Lithium Ion/Polymer charger - v2 ([1](#), [2](#), [5](#))
- SparkFun Sunny Buddy - MPPT Solar Charger ([1](#), [2](#), [3](#), [8](#))

## Akku

Vorsicht: Li-Ion/LiPo Akkus können bei falscher oder unsachgemäßer Behandlung explodieren oder giftige Gase freisetzen! Bevor der Akku mit dem Stromkreis verbunden wird, sollte man sich immer vorher vergewissert haben, dass kein Kurzschluss vorliegt. Auch sollten die Akkus niemals mit spitzen Gegenständen perforiert, geschnitten oder gesägt werden. Außerdem darf der Akku auf keinen Fall ins Feuer geworfen werden! Bitte dazu auch die Abschnitte [Hinweise zum Umgang mit Lithium-Ionen-Akkus](#)<sup>9</sup> und [Gefahren beim Umgang mit Lithium-Ionen-Akkus](#)<sup>10</sup> auf Wikipedia lesen!

Der verwendete Akku sollte mindestens so viel Kapazität besitzen, wie der Laderegler an Ladestrom bereitstellt. Mit einer Zelle größer 2500mAh ist man meistens auf der sicheren Seite. Die Bauform des Akkus ist relativ egal. Es sollte das gekauft werden, was passt. Es gibt auch Akkus mit bereits eingebauten Schutzschaltungen. Auch diese können verwendet werden. Die Art, wie der Akku mit dem Laderegler verbunden wird, entscheidet sich auch nach Geschmack. Der Stromkreis zwischen Akku und Laderegler sollte aber trennbar sein und in jedem Fall sollte ein LiPo Safe verwendet werden. Dieser sorgt bei einem Kurzschluss für Sicherheit gegenüber Explosionen. Im Abschnitt [Aufbau](#) wird ein LiPo in 18650 Bauweise mit Batteriehalter verwendet.

## Aufbau

Der in diesem Beispielaufbau verwendete Laderegler bietet einen Eingang für eine Stromquelle (in unserem Fall eine Solarzelle), einen Anschluss für den LiPo Akku und einen Ausgang für einen Verbraucher. Dieser ist in diesem Beispiel der Microcontroller. Der Eingang für die Stromquelle ist bei diesem Laderegler sowohl durch verlötbare Pins, als auch durch einen Micro-USB Anschluss ausgeführt. Durch diesen Micro-USB Anschluss kann der Akku mit einem Steckernetzteil,

z.B. von einem Handy, aufgeladen werden, bevor alles sich selbst bzw. der Sonne überlassen wird. Dabei sollte eine Höchststromstärke nicht überschritten werden. (Steht auf dem Netzteil) Der Laderegler sorgt dafür, dass der Akku weder überladen noch zu stark entladen wird.

Achtung: Die maximale Stromstärke des Laderegels sollte niemals überschritten werden. Diese ist dem Datenblatt zu entnehmen.

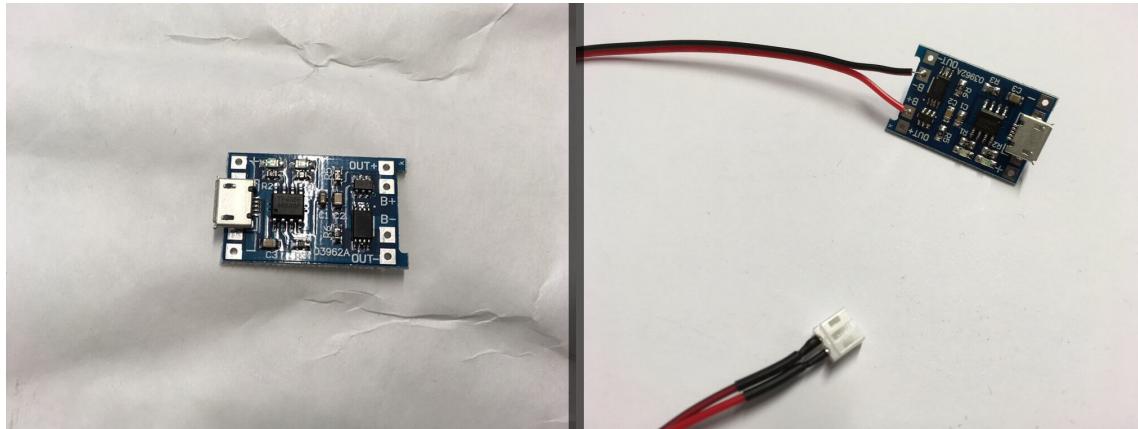


Abbildung 1: Solarladeregler mit angelötetem JST Kabel

Im Ersten Schritt sollte man sich nun Gedanken über die Spannungsversorgung des Microcontrollers machen. Der Laderegler hat eine Ausgangsspannung von 5 Volt. Dies erlaubt bei Arduino Nanos mit USB die direkte Versorgung über den USB Port, bei ESP8266 die Versorgung über entsprechende Vin Pins. Je nach verwendetem Microcontroller muss man sich nun eine Leitung für die Spannungsversorgung basteln. Dafür kann entweder ein altes USB-Kabel (aufschneiden) oder zwei separate Drähte, die dann per Pins den Microcontroller versorgen, verwendet werden. Bei der Variante mit dem USB Kabel muss mit Hilfe eines Multimeters die passenden Vcc und GND Leitungen ermittelt werden.

Die Stromversorgungsleitung kann nun mit einem Lötkolben an den Laderegler angelötet werden. Dabei muss auf die Polarität (Plus und Minus) geachtet werden.

Der Akku sollte nicht direkt untrennbar mit dem Laderegler verbunden werden. Um dies zu erreichen, gibt es mehrere Möglichkeiten: Entweder man besorgt sich einen passenden Akkuhalter oder, falls man einen Akku mit JST Stecker besitzt, besorgt man sich ein Kabel mit JST Buchse. In jedem Fall hat man eine Vcc und eine GND Leitung, welche passend mit dem Laderegler verlötet werden muss.

Achtung: Den Akku immer nur im LiPo Safe betreiben!

An dieser Stelle kann man nun mit einem Micro-USB Handynetzteil den Akku laden. Dazu einfach nur den Akku an den Laderegler klemmen und auf der Stromeingangsseite das Handynetzteil anschließen. Dabei kann der Laderegler allerdings sehr heiß werden.

Solarzellen gibt es entweder ohne Drähte, mit Drähten, mit JST Stecker oder mit USB Stecker. Die Variante mit dem USB-Stecker ist am einfachsten, da man einfach nur die Solarzelle in den Laderegler stecken muss und damit schon fertig ist. In den anderen Fällen muss man jeweils am Ende dafür sorgen, dass Vcc und GND mit den korrekten Eingangspins des Laderegels verbunden sind. Wichtig ist hierbei, dass die Solarzelle elektrisch zum Laderegler passt und wasserdicht ist!

Nun kann überlegt werden, wie der Microcontroller mit Spannung versorgt werden soll. Der hier verwendete Laderegler hat eine Ausgangsspannung von 5 Volt. Bei normalen Arduinos sollten diese 5 Volt am sichersten über den USB Port angelegt werden. Deswegen sollte man sich an dieser Stelle informieren, wie man seinen Microcontroller am besten mit den 5 Volt versorgt.

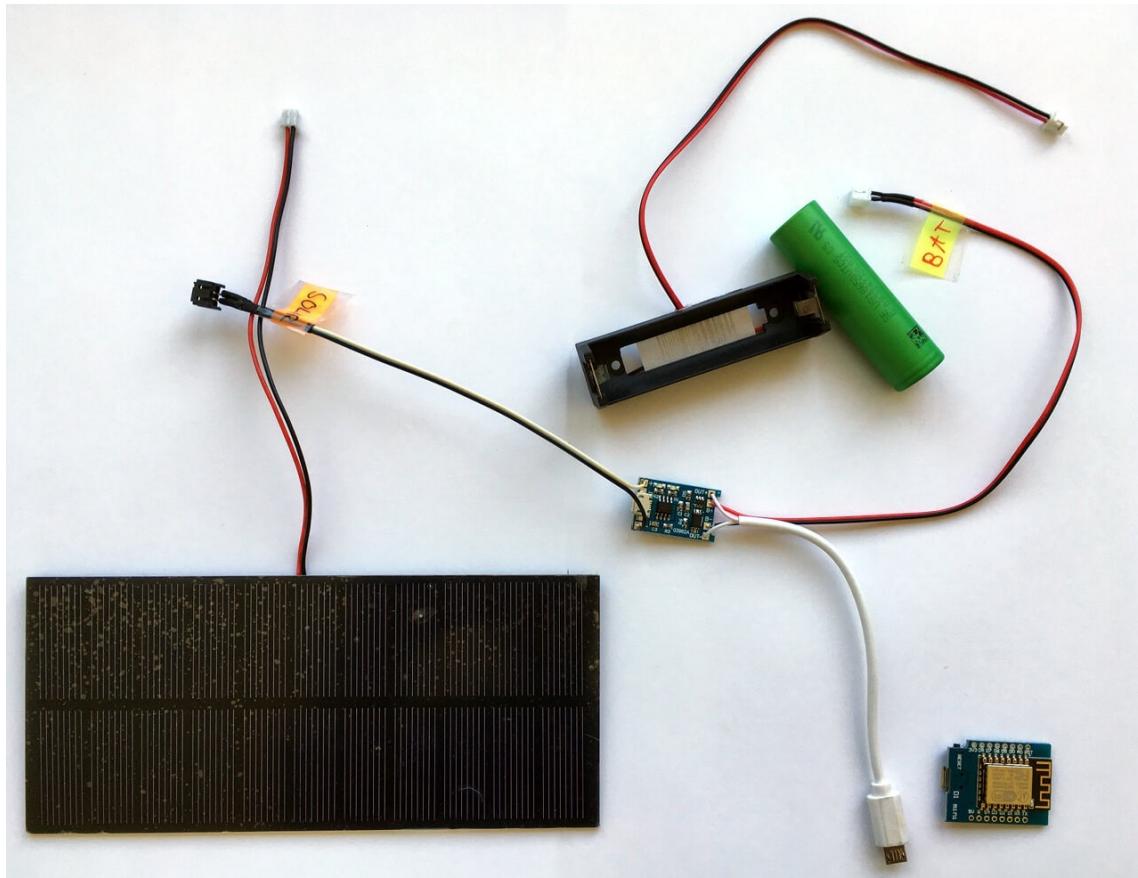


Abbildung 2: Solarladeregler mit Solarzelle, Akku und Versorgungskabel für Microcontroller

Eine Möglichkeit ist es, ein passendes USB-Kabel aufzutrennen und die Spannungsversorgungsleitungen an den Laderegler zu löten. Als eine andere Möglichkeit, sofern es der verwendete Microcontroller es erlaubt, kann man sich auch eine andersartige Steckverbindung basteln.

Achtung: Damit der Microcontroller korrekt mit Spannung versorgt wird, muss auf jeden Fall auf die richtige Polarität geachtet werden (Plus und Minus). Die alleinige Verantwortung liegt bei dir, nicht bei uns!

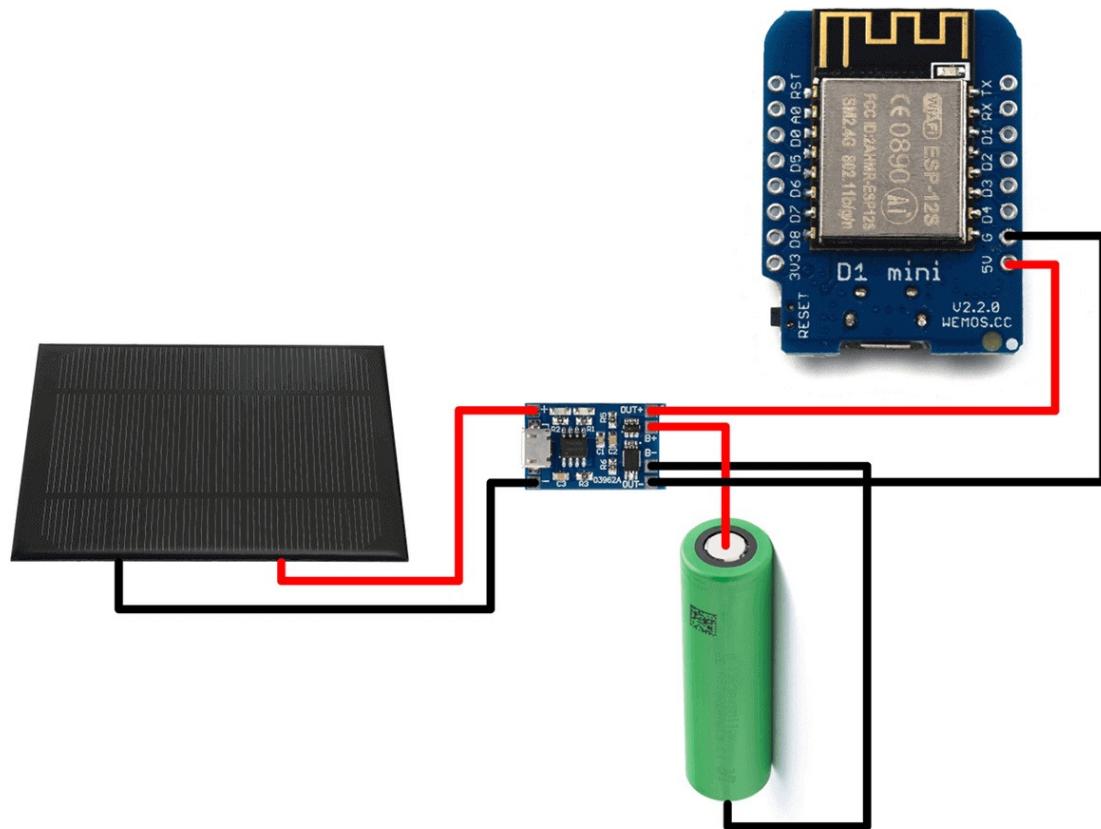


Abbildung 3: Beispielschaltung

Die Schaltung (siehe Abbildung 3) sollte nun soweit korrekt laufen und kann ausprobiert werden. Zuerst sollte der Akku, danach der Microcontroller und dann die Solarzelle verbunden und ins Licht gehalten werden. Leuchtet eine zusätzliche LED am Laderegler auf, ist alles korrekt und die Programmierung des Microcontroller kann angepasst werden.

Die gesamte Schaltung sollte wasserundurchlässig und witterungsgeschützt aufgestellt werden. Dabei ist es auch wichtig, dass die Solarzelle idealerweise in Richtung Süden aufgestellt wird, damit immer genügend Strom zum Laden des Akkus vorhanden ist.

## Microcontroller programmieren

Damit der Akku niemals leer wird muss die Programmierung auf geringen Stromverbrauch getrimmt werden. Beim ESP8266 kann dies durch die Verwendung des DeepSleep und beim Arduino durch die Verwendung der Library Narcoleptic erreicht werden. Insgesamt ist es wichtig, dass der Microcontroller möglichst kurze Arbeitsphasen hat und dazwischen wieder schläft. Deswegen macht es Sinn, die Standard-Übertragungsrate von einer Minute durch 5 oder 10 minütige Intervalle zu ersetzen.

### ESP8266

Nicht alle ESP8266 sind gleich, so eignet sich z.B. der Wemos D1 Mini besser für diese Anleitung als der in Arduino Uno Bauform gebaute Wemos D1 R2. Dies ist begründet durch sparsamere und einer geringeren Anzahl an Bauteilen. Ein Nachteil des D1 Mini liegt darin, dass man die Stecker der senseBox:home Sensoren nicht wiederverwenden kann. Man muss also die Sensoren anderweitig mit dem Microcontroller verbinden, was jedoch nicht Inhalt dieser Anleitung sein soll.

Damit der ESP8266 in den DeepSleep wechseln und danach auch wieder Zeitgesteuert aufwachen kann, muss eine elektrische Verbindung zwischen GPIO Pin 16 und Pin RST bestehen. Dies kann durch eine kleine Drahtbrücke erreicht werden. Deswegen eignen sich alle ESP8266 Ausführungen, die Pin 16 einfach zugänglich machen am besten.

Bevor man hier weitermacht, sollte mindestens einmal die [Anleitung ESP8266<sup>11</sup>](#) gelesen werden. Hier wird auch erklärt, wie man Treiber installiert und die Arduino IDE vorbereitet.

Der angepasste Sketch basiert im Großen und Ganzen auf dem Standard senseBox:home Sketch, der Unterschied liegt aber darin, dass durch den DeepSleep die `loop` Methode niemals erreicht wird. Der ESP8266 startet sich nach dem DeepSleep jedes Mal neu. Der Sketch muss also die Messungen und das Übertragen an die openSenseMap alles im `setup` abhandeln.

Einen Beispielsketch für eine senseBox:home mit ESP8266 und DeepSleep befindet sich [hier<sup>12</sup>](#).

## Arduino

Der Arduino Uno oder Nano hat leider keinen Sleep Modus, kann aber durch die [Narcoleptic Library<sup>13</sup>](#) zu einem geringeren Stromverbrauch angeregt werden. Diese deaktiviert alle unnötigen Vorgänge des Arduino.

Der originale senseBox:home Sketch kann fast 1:1 weiterbenutzt werden. Es muss nur die Narcoleptic Library eingebunden und aktiviert werden.

```
#include <Narcoleptic.h>
```

Am Ende der `setup` Methode:

```
Narcoleptic.disableTimer1();
Narcoleptic.disableTimer2();
Narcoleptic.disableSerial();
Narcoleptic.disableADC();
Narcoleptic.disableSPI();
```

In der Methode `loop` muss noch der Aufruf `sleep(postingInterval);` durch `Narcoleptic.delay(postingInterval);` ersetzt werden.

## Abschließende Worte

Die in dieser Anleitung gezeigten Schaltungen und Techniken umreißen den Aufbau einer energieautarken senseBox:home. Wie Eingangs erwähnt, bildet dieses Vorgehen nur einen Weg von vielen, um das Ziel einer solarbetriebenen senseBox:home zu erreichen ab. Die konkrete Durchführung hängt stark von den verwendeten Teilen ab.

<sup>1</sup>. Siehe [2.2 Solarbetriebene senseBox:home > auswahl-des-ladereglers-solarzelle-und-akku](#) ↵

<sup>2</sup>. Siehe [2.2 Solarbetriebene senseBox:home > aufbau](#) ↵

<sup>3</sup>. Siehe [2.2 Solarbetriebene senseBox:home > aufbau](#) ↵

<sup>1</sup>. <http://www.watterott.com/de/TP4056-Micro-USB-5V-1A-Lithium-Battery-Charger-with-Protection> ↵

<sup>2</sup>. <http://www.ebay.de/itm/231820310558> ↵

<sup>3</sup>. <https://www.amazon.de/dp/B00QI6TR54> ↵

<sup>4</sup>. <http://www.exp-tech.de/adafruit-usb-dc-solar-lithium-ion-polymer-charger-v2> ↵

<sup>5</sup>. <https://eckstein-shop.de/Adafruit-USB-DC-Solar-Lithium-Ion-Polymer-charger> ↵

- 6. <http://www.exp-tech.de/sparkfun-sunny-buddy-mppt-solar-charger> ↵
- 7. <https://www.amazon.de/dp/B01BTV90UE> ↵
- 8. <https://eckstein-shop.de/SparkFun-Sunny-Buddy-MPPT-Solar-Charger-EN> ↵
- 9. [https://de.wikipedia.org/wiki/Lithium-Ionen-Akkumulator#Hinweise\\_zum\\_Umgang\\_mit\\_Lithium-Ionen-Akkus](https://de.wikipedia.org/wiki/Lithium-Ionen-Akkumulator#Hinweise_zum_Umgang_mit_Lithium-Ionen-Akkus) ↵
- 10. [https://de.wikipedia.org/wiki/Lithium-Ionen-Akkumulator#Gefahren\\_beim\\_Umgang\\_mit\\_Lithium-Ionen-Akkus](https://de.wikipedia.org/wiki/Lithium-Ionen-Akkumulator#Gefahren_beim_Umgang_mit_Lithium-Ionen-Akkus) ↵
- 4. Siehe [2.2 Solarbetriebene senseBox:home > aufbau](#) ↵
- 11. <https://home.books.sensebox.de/de/esp8266/> ↵
- 12. <https://github.com/ubergesundheit/senseBox-ESP8266/blob/master/senseBox-ESP8266-Solar.ino> ↵
- 13. <https://github.com/doc62/narcoleptic> ↵

## senseBox:home mit ESP8266

Alternativ zum Arduino/Genuino lassen sich auch andere Plattformen als Microcontroller der Sensorstation nutzen. Hier wird beispielhaft der ESP8266 verwendet, welcher ein schnellerer Mikrokontroller mit eingebautem WLAN ist. Er lässt sich mit der Arduino IDE programmieren, was die Anpassungsarbeit der Software sehr einfach macht. Es gibt relativ viele Varianten dieses Mikrocontrollers.

### Achtung

- Leider funktioniert das Barometer (BMP280) nicht korrekt mit der `BMP280.h` Bibliothek, stattdessen kann die [BME280.h Bibliothek<sup>1</sup>](#) verwendet werden.
- Diese Anleitung beschreibt die Möglichkeit den `ESP8266` mit der Arduino IDE zu programmieren. Es existieren jedoch viele weitere Möglichkeiten Software für den ESP8266 zu schreiben.
- Der ESP8266 kann nur mit 3.3V betrieben werden. Der 3.3V Pin des Arduinos stellt jedoch nicht genügend Strom bereit, um den ESP zu betreiben, eine externe Stromversorgung ist nötig!

1. Alle ESP8266 Varianten: [Installation der Software<sup>1</sup>](#)
2. Alle ESP8266 Varianten: [Anpassen des senseBox:home Sketch<sup>2</sup>](#)
3. Am einfachsten: [Wemos D1 mit senseBox Shield<sup>3</sup>](#)
4. [Alle anderen ESP8266 Varianten](#)

---

<sup>1</sup>. <https://github.com/finitespace/BME280> ↵

<sup>1</sup>. Siehe [2.3.1 Softwareinstallation](#) ↵

<sup>2</sup>. Siehe [2.3.2 Anpassen des Sketches](#) ↵

<sup>3</sup>. Siehe [2.3.3 senseBox mit Wemos D1](#) ↵

## Installation von **ESP8266 Arduino core**

Um den ESP8266 über die Arduino IDE programmierbar zu machen, wird eine passende Toolchain benötigt. Eine solche wurde als `ESP8266 Arduino core` von der ESP-Community entwickelt.

Zuallererst sollten die Softwarebibliotheken für die Sensoren installiert werden, falls dies noch nicht getan wurde (siehe [Anleitung<sup>1</sup>](#)).

Danach muss die `ESP8266 Arduino core` Toolchain installiert werden. Diese Anleitung ist analog zur Anleitung aus dem [Github Repository<sup>1</sup>](#):

1. Arduino starten
  2. Im Menü unter Datei -> Voreinstellungen unter 'Additional Board Manager URLs' die Adresse  
`http://arduino.esp8266.com/stable/package_esp8266com_index.json` einfügen
  3. Im Menü Werkzeuge -> Platine -> Boards Manager kann man nun mit der Suche oben rechts im Fenster `ESP8266 by ESP8266 Community` installieren
  4. Das entsprechende Board unter Werkzeuge -> Platine auswählen. Sollte nicht explizit das vorhandene Board zur Auswahl stehen, ist "Generic ESP8266" eine gute Wahl.
- 

<sup>1</sup>. Siehe [1.3 Softwareinstallation](#) ↵

<sup>1</sup>. <https://github.com/esp8266/Arduino#installing-with-boards-manager> ↵

# openSenseMap-Sketch für den ESP8266 anpassen

Damit der senseBox-Sketch auf dem ESP8266 läuft, sind ein paar Anpassungen nötig. Zuallererst benötigt man seinen senseBox-Sketch. Entweder man registriert eine neue senseBox ([Anleitung hier<sup>1</sup>](#)), oder man verwendet seinen bestehenden Sketch. Wichtig ist dass man seine senseBox-ID und Sensor-IDs kennt.

Damit das Barometer (BMP280) korrekt in mit dem ESP8266 funktioniert, wird eine andere Bibliothek ([BME280 Version 1.0.02<sup>1</sup>](#)) zum ansteuern des Sensors benötigt. Diese kann [hier<sup>2</sup>](#) heruntergeladen werden. Die Installation erfolgt analog zu unseren Bibliotheken, wie [hier<sup>2</sup>](#) beschrieben ist.

Hier findet ihr einen komplett angepassten Sketch. Dort müssen nur noch WiFi-Name und -Passwort, senseBox-ID und Sensor-IDs ersetzt werden.

## Kompletter Sketch

```
/*
senseBox Home - Citizen Sensingplatform
Version: 2.4
Date: 2017-Mar-06
Homepage: https://www.sensebox.de https://www.opensensemaps.org
Author: Institute for Geoinformatics, University of Muenster
Note: Sketch for senseBox:home
Email: support@sensebox.de
*/

#include <Wire.h>
#include "HDC100X.h"
#include <BME280.h>
#include <Makerblog_TSL45315.h>
#include <SPI.h>
#include <ESP8266WiFi.h>

// Wifi Configuration
const char* server = "ingress.opensensemaps.org";
WiFiClient client;
const char* ssid = "dein-wifi-name";
const char* password = "dein-wifi-passwort";

typedef struct sensor {
    const uint8_t ID[12];
} sensor;

uint8_t sensorsIndex = 0;

// Number of sensors
static const uint8_t NUM_SENSORS = 5;

// senseBox ID and sensor IDs
const uint8_t SENSEBOX_ID[12] = { 0x58, 0xd9, 0x1a, 0x6f, 0x68, 0x91, 0xfd, 0x00, 0x0f, 0x8a, 0xf7, 0xed };

// Do not change order of sensor IDs
const sensor sensors[NUM_SENSORS] = {
    // Temperatur
    { 0x58, 0xd9, 0x1a, 0x6f, 0x68, 0x91, 0xfd, 0x00, 0x0f, 0x8a, 0xf7, 0xee },
    // rel. Luftfeuchte
    { 0x58, 0xd9, 0x1a, 0x6f, 0x68, 0x91, 0xfd, 0x00, 0x0f, 0x8a, 0xf7, 0xef },
    // Luftdruck
    { 0x58, 0xd9, 0x1a, 0x6f, 0x68, 0x91, 0xfd, 0x00, 0x0f, 0x8a, 0xf7, 0xf0 },
    // Beleuchtungsstärke
    { 0x58, 0xd9, 0x1a, 0x6f, 0x68, 0x91, 0xfd, 0x00, 0x0f, 0x8a, 0xf7, 0xf1 },
    // UV-Intensität
    { 0x58, 0xd9, 0x1a, 0x6f, 0x68, 0x91, 0xfd, 0x00, 0x0f, 0x8a, 0xf7, 0xf2 }
}
```

```

};

uint8_t contentLength = 0;
float values[NUM_SENSORS];

// Load sensors
Makerblog_TSL45315 TSL = Makerblog_TSL45315(TSL45315_TIME_M4);
HDC100X HDC(0x43);
BME280 BME;
uint8_t pressureUnit(1); // hPa
#define UV_ADDR 0x38
#define IT_1 0x1

const unsigned int postingInterval = 60000;

void setup() {
    sleep(2000);
    Serial.begin(9600);
    // start the Wifi connection:
    Serial.println("SenseBox Home software version 2.1");
    Serial.println();
    Serial.print("Starting wifi connection...");
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("done!");
    sleep(1000);
    Serial.print("Initializing sensors...");
    Wire.begin();
    Wire.beginTransmission(UV_ADDR);
    Wire.write((IT_1 << 2) | 0x02);
    Wire.endTransmission();
    sleep(500);
    HDC.begin(HDC100X_TEMP_HUMI, HDC100X_14BIT, HDC100X_14BIT, DISABLE);
    TSL.begin();
    BME.begin();
    Serial.println("done!");
    HDC.getTemp();
    Serial.println("Starting loop.\n");
}

void loop() {
   .addValue(HDC.getTemp());
    sleep(200);
   .addValue(HDC.getHumi());
   .addValue(BME.ReadPressure(pressureUnit));
   .addValue(TSL.readLux());
   .addValue(getUV());

    submitValues();

    sleep(postingInterval);
}

void.addValue(const float& value) {
    values[sensorsIndex] = value;
    sensorsIndex = sensorsIndex + 1;
}

uint16_t getUV() {
    byte msb = 0, lsb = 0;
    uint16_t uvValue;
    Wire.requestFrom(UV_ADDR + 1, 1); // MSB
    sleep(1);
    if (Wire.available())
        msb = Wire.read();
}

```

```

Wire.requestFrom(UV_ADDR + 0, 1); // LSB
sleep(1);
if (Wire.available())
    lsb = Wire.read();
uvValue = (msb << 8) | lsb;
return uvValue * 5;
}

int printHexToStream(const uint8_t* data,
                      uint8_t length,
                      Print& stream) // prints 8-bit data in hex
{
    byte first;
    int j = 0;
    for (uint8_t i = 0; i < length; i++) {
        first = (data[i] >> 4) | 48;
        if (first > 57) {
            stream.write(first + (byte)39);
        } else {
            stream.write(first);
        }
        j++;
    }
    first = (data[i] & 0x0F) | 48;
    if (first > 57) {
        stream.write(first + (byte)39);
    } else {
        stream.write(first);
    }
    j++;
}
return j;
}

int printCsvToStream(Print& stream) {
    int len = 0;
    for (uint8_t i = 0; i < sensorsIndex; i++) {
        if (!isnan(values[i])) {
            len = len + printHexToStream(sensors[i].ID, 12, stream);
            len = len + stream.print(",");
            // do not print digits for illuminance und uv-intensity
            if (i < 3)
                len = len + stream.println(values[i]);
            else
                len = len + stream.println(values[i], 0);
        }
    }
    return len;
}

// millis() rollover fix -
// http://arduino.stackexchange.com/questions/12587/how-can-i-handle-the-millis-rollover
void sleep(unsigned long ms) { // ms: duration
    unsigned long start = millis(); // start: timestamp
    for (;;) {
        unsigned long now = millis(); // now: timestamp
        unsigned long elapsed = now - start; // elapsed: duration
        if (elapsed >= ms) // comparing durations: OK
            return;
    }
}

void waitForResponse() {
    // if there are incoming bytes from the server, read and print them
    sleep(100);
    String response = "";
    char c;
    boolean repeat = true;
    do {

```

```

        if (client.available())
            c = client.read();
        else
            repeat = false;
        response += c;
        if (response == "HTTP/1.1 ")
            response = "";
        if (c == '\n')
            repeat = false;
    } while (repeat);

    Serial.print("Server Response: ");
    Serial.print(response);

    client.flush();
    client.stop();
}

void submitValues() {
    // close any connection before send a new request.
    // This will free the socket on the WiFi shield
    Serial.println("_____ \n");
    if (client.connected()) {
        client.stop();
        sleep(1000);
    }
    // if there's a successful connection:
    if (client.connect(server, 80)) {
        Serial.println("connecting...");
        // send the HTTP POST request:

        client.print(F("POST /boxes/"));
        printHexToStream(SENSEBOX_ID, 12, client);
        client.println(F("/data HTTP/1.1"));

        // !!!!! DO NOT REMOVE !!!!
        // !!!!! NICHT LÖSCHEN !!!!
        // print once to Serial to get the content-length
        int contentLen = printCsvToStream(Serial);
        // !!!!! DO NOT REMOVE !!!!
        // !!!!! NICHT LÖSCHEN !!!!

        // Send the required header parameters
        client.print(F("Host: "));
        client.println(server);
        client.print(
            F("Content-Type: text/csv\nConnection: close\nContent-Length: "));
        client.println(contentLen);
        client.println();
        printCsvToStream(client);
        client.println();
        Serial.println("done!");

        waitForResponse();

        // reset index
        sensorsIndex = 0;
    } else {
        // if you couldn't make a connection:
        Serial.println("connection failed. Restarting System.");
        sleep(5000);
        ESP.restart();
    }
}

```

<sup>1</sup>. Siehe [1.4 openSenseMap Registrierung](#) ↵

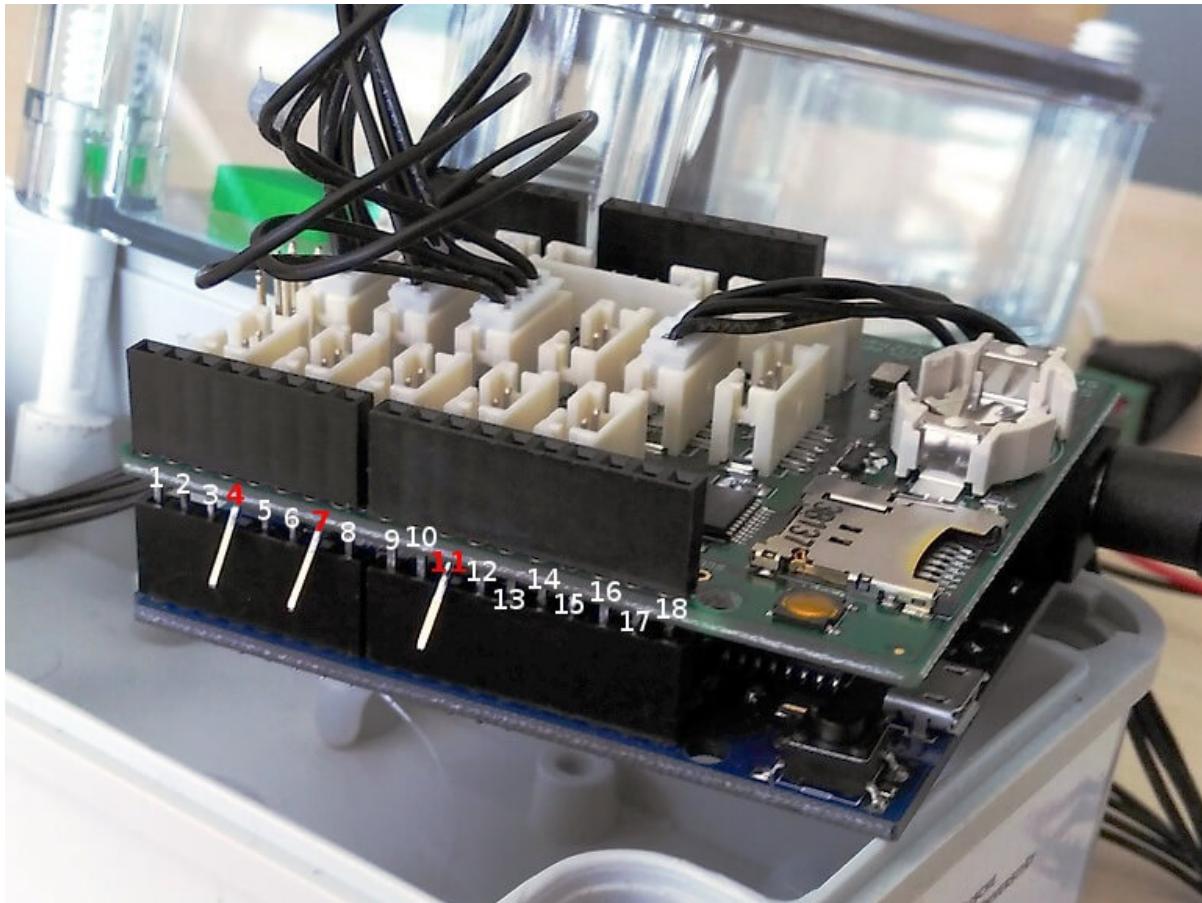
- | 1. <https://github.com/finitespace/BME280/tree/fa4a6d4412dfb46f135cb68c15680a57ff68dfda> ↵
- | 2. [https://github.com/finitespace/BME280/archive/Release\\_Version\\_1.0.02.zip](https://github.com/finitespace/BME280/archive/Release_Version_1.0.02.zip) ↵
- | 2. Siehe [1.3 Softwareinstallation](#) > arduino-bibliotheken-installieren ↵

## senseBox:home mit Wemos D1 und senseBox Shield

Der [Wemos D1](#)<sup>1</sup> ist ein Arduino Uno kompatibles Board mit einem ESP8266 als Mikrokontroller.

### Anpassungen am senseBox Shield

Im Prinzip ist das senseBox:home Shield kompatibel mit dem Wemos D1. Damit das senseBox Shield mit dem Wemos D1 zusammenarbeitet, müssen ein paar der Pins abgebogen werden. Auf diesem Bild siehst du welche:



Achtung: Für defekte Shields oder Sensoren übernehmen wir keine Haftung!

---

<sup>1</sup>. <https://www.wemos.cc/product/d1.html> ↵

## senseBox:home mit anderen ESP8266 Varianten

Wie Eingangs schon erwähnt, ist der ESP8266 Mikrokontroller weitestgehend mit dem Arduino kompatibel. Die senseBox verwendet den I<sup>2</sup>C Bus um mit den Sensoren zu kommunizieren. Dieser verwendet standardmäßig Pin 4 ( SDA ) und 5 ( SCL ).

### ESP8266-01

Das Modell 01 des ESP8266 hat nur Pin 0 und 2 auf der Hardware verfügbar. Hier muss zum initialisieren des I<sup>2</sup>C Bus `Wire.begin(0, 2)` verwendet werden.

#### Hinweise zum Debuggen

- Leuchtet die Blaue LED auf, kommen Daten auf dem RX Pin an
- Beim Booten gibt der Bootloader des ESP Debuginformationen mit 74880 Baud aus, anschließend läuft NodeMCU (sofern installiert) mit 115200 Baud.

Der ESP hat verschiedene Bootmodi, welche über die Spannung an GPIO 0 und 2 gewählt werden. Um den ESP normal zu booten, sodass dieser sein Programm ausführt, benötigen beide Pins einen Pullup auf 3.3V. Um ein Programm auf den ESP zu schreiben muss der UART-Modus verwendet werden, welcher mit einem Pulldown auf GPIO 0 und einem Pullup auf GPIO 2 aktiviert wird.

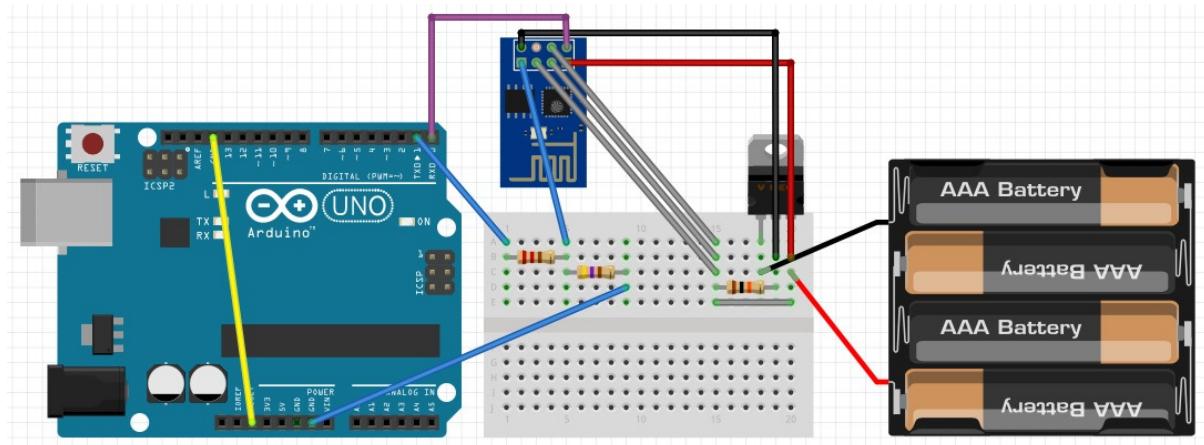
- Siehe [hier<sup>1</sup>](#) für Informationen zu Pullup-Widerständen.
- Siehe [hier<sup>2</sup>](#) für Informationen zu den Bootmodi des ESP8266.

### TTL-bridge über Arduino

Die kompakteren ESP Modelle haben meist keinen USB Anschluss über welchen der serielle Port des Mikrocontrollers angesprochen werden könnte. Um solche Boards zu programmieren ist eine separater USB-TTL-converter notwendig. Falls gerade keine dedizierte Hardware vorliegt, kann ein Arduino dazu zweckentremdet werden!

Legt man den RST Pin des Arduinos auf GND, wird effektiv der Prozessor des Arduinos deaktiviert. Dadurch geben die Pins 0 und 1 die Signale des seriellen Busses unverändert weiter.

Da Arduinos meist mit 5V und ESPs mit 3,3V arbeiten, muss die Spannung der Signale auf ein einheitliches Level gebracht werden: Am einfachsten wird dazu der RX -Output des Arduinos durch einen Voltage-Divider auf 3,3V gebracht (blaue Kabel). Die TX Leitung (lila Kabel) kann direkt verbunden werden, da der Arduino auch 3,3V-Spannungen als HIGH versteht. Die vollständige Schaltung - inklusive Pullups (graue Kabel) und Stromversorgung des ESP8266 - sieht also wie folgt aus:



Hinweis: Der 3.3V -Pin des Arduinos stellt nicht genügend Strom bereit, um den ESP8266 zu betreiben. Hierzu sollte eine separate Stromversorgung (3,3V!) verwendet werden! Siehe [hier<sup>3</sup>](#) für Informationen zu Voltage Dividern.

Nun kann der Rechner mit dem ESP direkt kommunizieren. Zum Testen einfach den seriellen Monitor der Arduino IDE öffnen, und Kommandos an den ESP schicken!

1. <https://learn.sparkfun.com/tutorials/pull-up-resistors> ↵

2. <https://github.com/esp8266/Arduino/blob/master/doc/boards.md#boot-messages-and-modes> ↵

3. <https://learn.sparkfun.com/tutorials/voltage-dividers> ↵

## Beispielaufbauten und Bauanleitungen

Hier sind verschiedene exemplarische Aufbauten der senseBox:home dokumentiert. Da wir zur Zeit noch an einer Lösung für ein Gehäuse für den Outdoor-Betrieb arbeiten, musst du diesbezüglich selber kreativ werden. Nutze die folgenden Anleitungen als Empfehlung und Inspiration!

Titel	Autor	Datum
Fibox Gehäuse mit PoE <sup>1</sup>	senseBox Team	2015
MyOSD WiFi (PDF)	MyOSD	Okt. 2016
Botanischer Garten <sup>2</sup>	Felix	März. 2017

Wenn du selbst einen Aufbau entwickelt und dokumentiert hast, fügen wir diesen gern zu dieser Liste hinzu! Erstelle hierzu einen Pull Request auf [GitHub](#)<sup>1</sup>, oder sende uns die Dokumentation per E-Mail<sup>2</sup> (idealerweise im [Markdown Format](#)<sup>3</sup>) zu!

<sup>1</sup>. Siehe [3.1.1 senseBox fibox](#) ↵

<sup>2</sup>. Siehe [3.1.2 Botanischer Garten](#) ↵

<sup>1</sup>. <https://github.com/sensebox/books> ↵

<sup>2</sup>. [mailto:info@sensebox.de?subject=Aufbau senseBox:home](mailto:info@sensebox.de?subject=Aufbau%20senseBox:home) ↵

<sup>3</sup>. <http://five.squarespace.com/display>ShowHelp?section=Markdown> ↵

## Aufbau mit FIBOX-Gehäuse

Dieser Outdoor-Aufbau nutzt ein wasserfestes Gehäuse, ein separates 3D-druckbares Temperatursensor-Gehäuse, sowie Power-over-Ethernet Adapter zum Betrieb der senseBox. Die zusätzlich zur senseBox benötigten Materialien sind:

- ein wasserfestes Gehäuse mit klarem Deckel ([dieses Gehäuse<sup>1</sup>](#), [hier<sup>2</sup>](#) bestellbar) mit den Mindestmaßen 5.5cm x 10cm



x 5cm (BxTxH)

- ein vor Spritzwasser und Strahlung schützendes, luftdurchlässiges Gehäuse. Wir haben [dieses 3D-druckbare MoDell<sup>3</sup>](#) speziell für das Breakout Board des HDC1008 entworfen.
- einen Satz [Power-over-Ethernet \(PoE\) Adapter<sup>4</sup>](#)
- ein (Flachband)-Netzwerkkabel, Länge je nach Aufbauort.

Die Kosten für die Materialien belaufen sich auf ca. 20€.

Bei dem Gehäuse ist darauf zu achten, dass es einen transparenten Deckel ohne Lichtfilterwirkung hat, damit sinnvolle Lichtmessungen gemacht werden können.

Zudem braucht ihr noch eine Heißklebepistole, Bohrmaschine, Schraubendreher sowie ein paar Kabelbinder zur Befestigung.

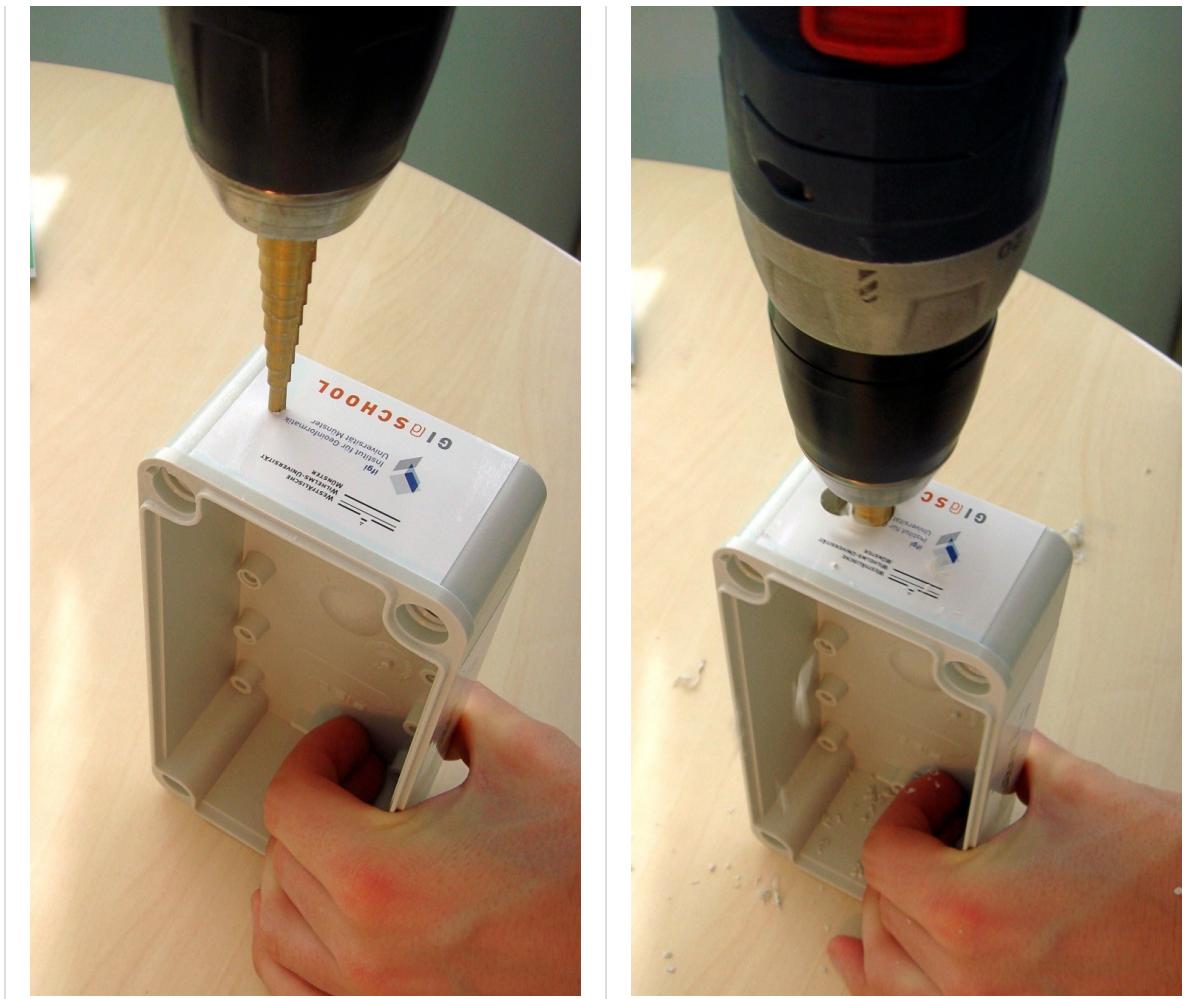
Im folgenden Video wird der Zusammenbau gezeigt, weiter unten folgt eine Schritt-für-Schritt Anleitung:

Eingebettetes Video: <https://www.youtube.com/watch?v=TgEQvMPjrMA>

## Kabelführung bohren

Durch eine ca. 15mm breite Bohrung im Boden des Gehäuses werden Strom- und Netzwerkkabel gelegt, sowie das lange Verbindungskabel für den HDC1008 Temperatur-/Luftfeuchtesensor:

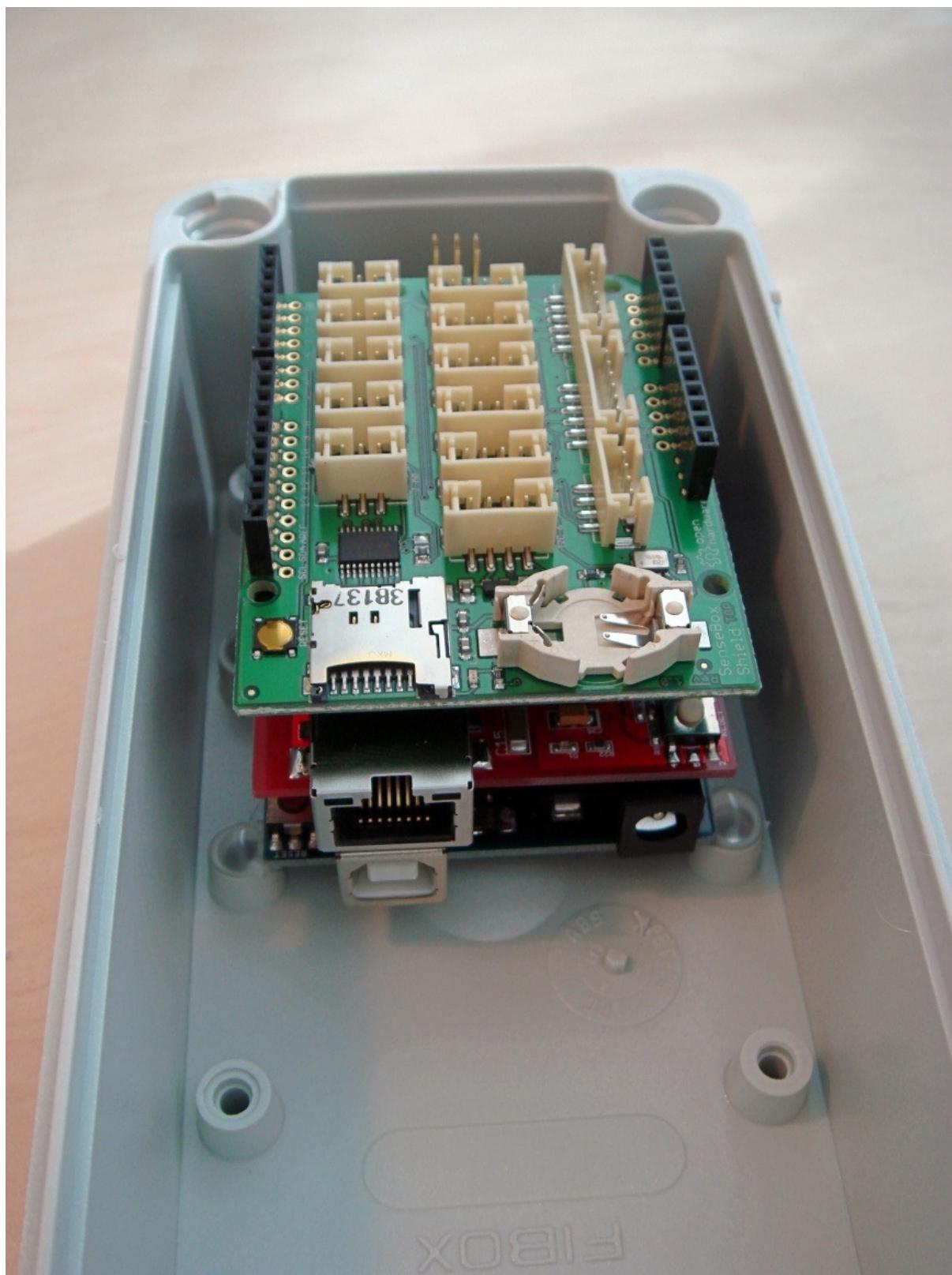




Im Video an der Stelle 00:00

## Hauptplatinen festkleben

Dazu setzen wir im Gehäuse einige Klebepunkte mit dem Heißkleber und drücken die Hauptplatine an, bis der Kleber getrocknet ist:



Im Video an der Stelle 00:50

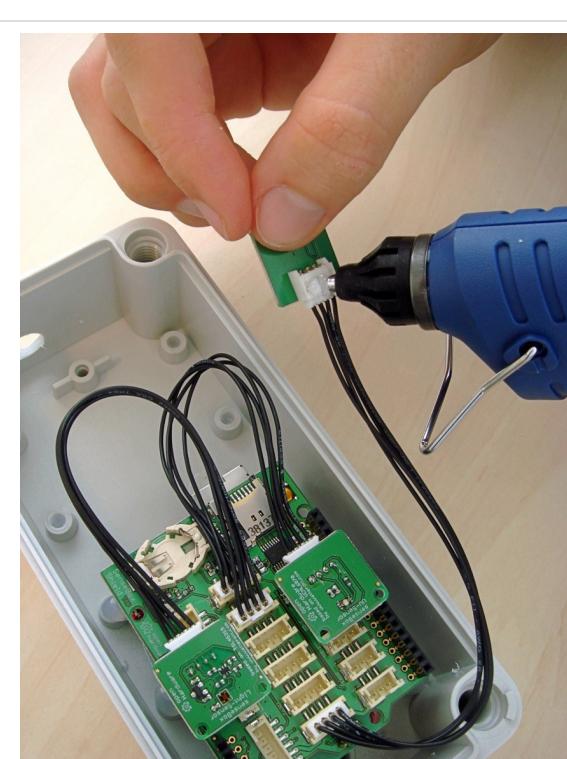
## Sensoren befestigen

Es sollte darauf geachtet werden, dass keine Kleberreste auf die Oberseite der Sensorplatten kommt! Beim festkleben der Sensoren reicht schon ein wenig Heißkleber aus.

Die beiden Lichtsensoren oben auf das senseBox-Shield kleben. Die beiden Lichtsensoren sollten "freie Sicht" zum transparenten Deckel haben und nicht von den Kabeln bedeckt werden!



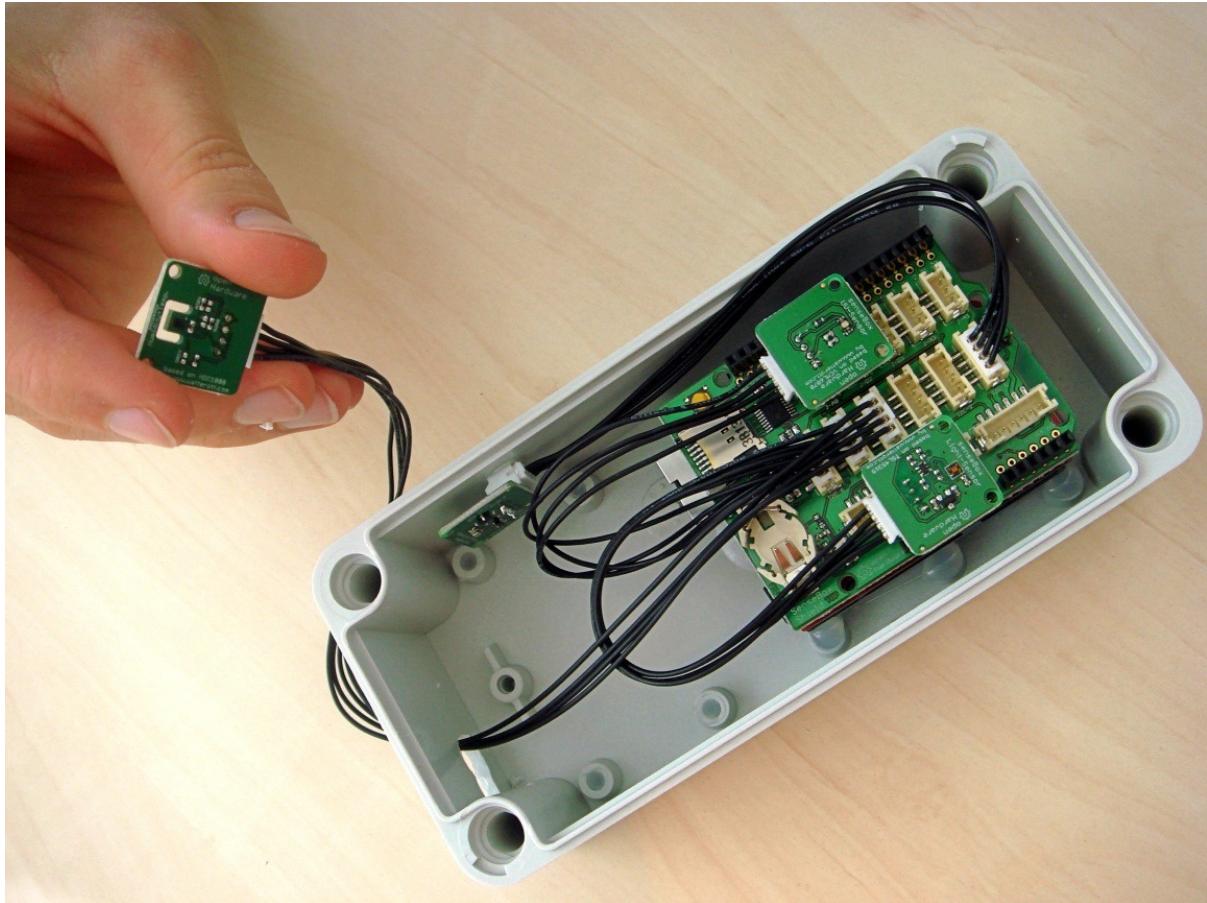
Den Luftdrucksensor ebenfalls im Gehäuse mit etwas Heißkleber weiter vorne befestigen:



Im Video an der Stelle [02:02](#)

## Temperatursensor anbringen

Im Gehäuse werden Temperatur und Luftfeuchte durch die Eigenwärme des Mikrocontrollers beeinflusst. Daher muss der HDC1008 außerhalb in einem zweiten Gehäuse angebracht werden, in dem er vor Regen oder Spritzwasser geschützt ist. Dazu führen wir das lange Sensor-Verbindungskabel durch die Bohrung nach außen und verbinden es mit dem Sensor.

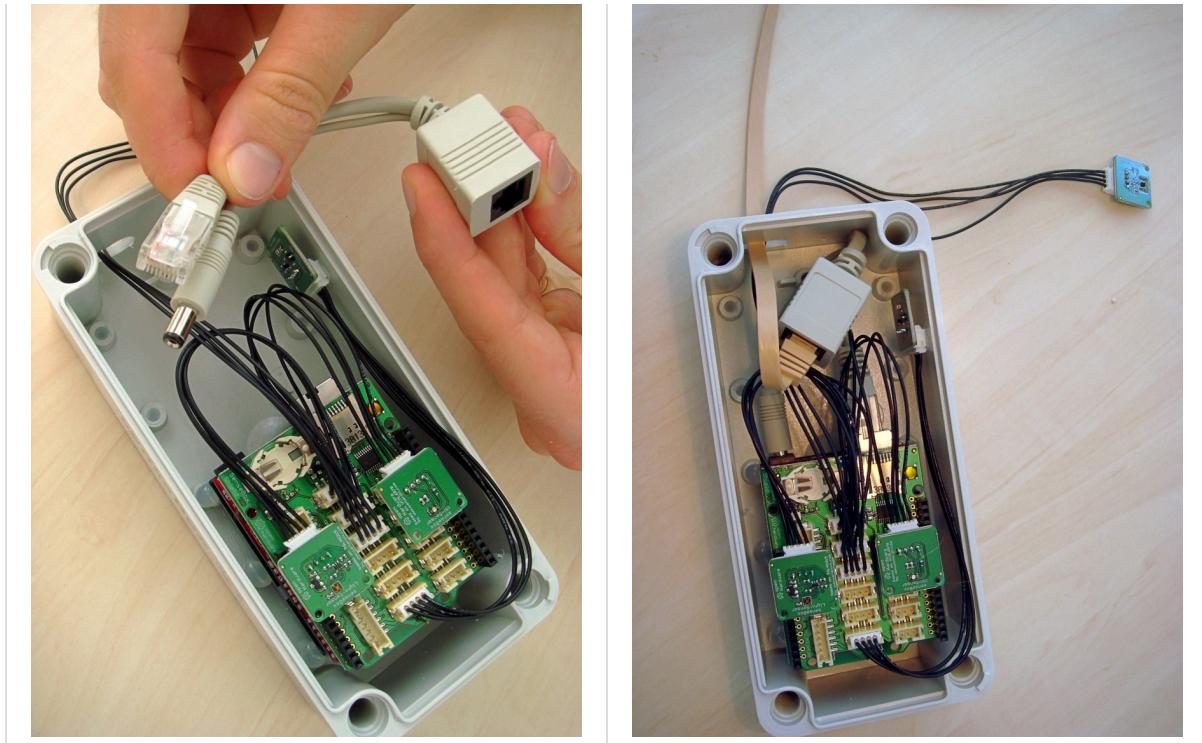


Im Video an der Stelle [04:20](#)

## Strom- und Netzwerkanschluss

Um die senseBox mit Strom zu versorgen, kann ein Power-over-Ethernet-Adapter (POE) verwendet werden. Dieser wird an den Netzwerk- und Stromanschluss der Hauptplatten angeschlossen. Danach kann das Ethernetkabel durch das Bohrloch geführt und in den Adapter gesteckt werden.



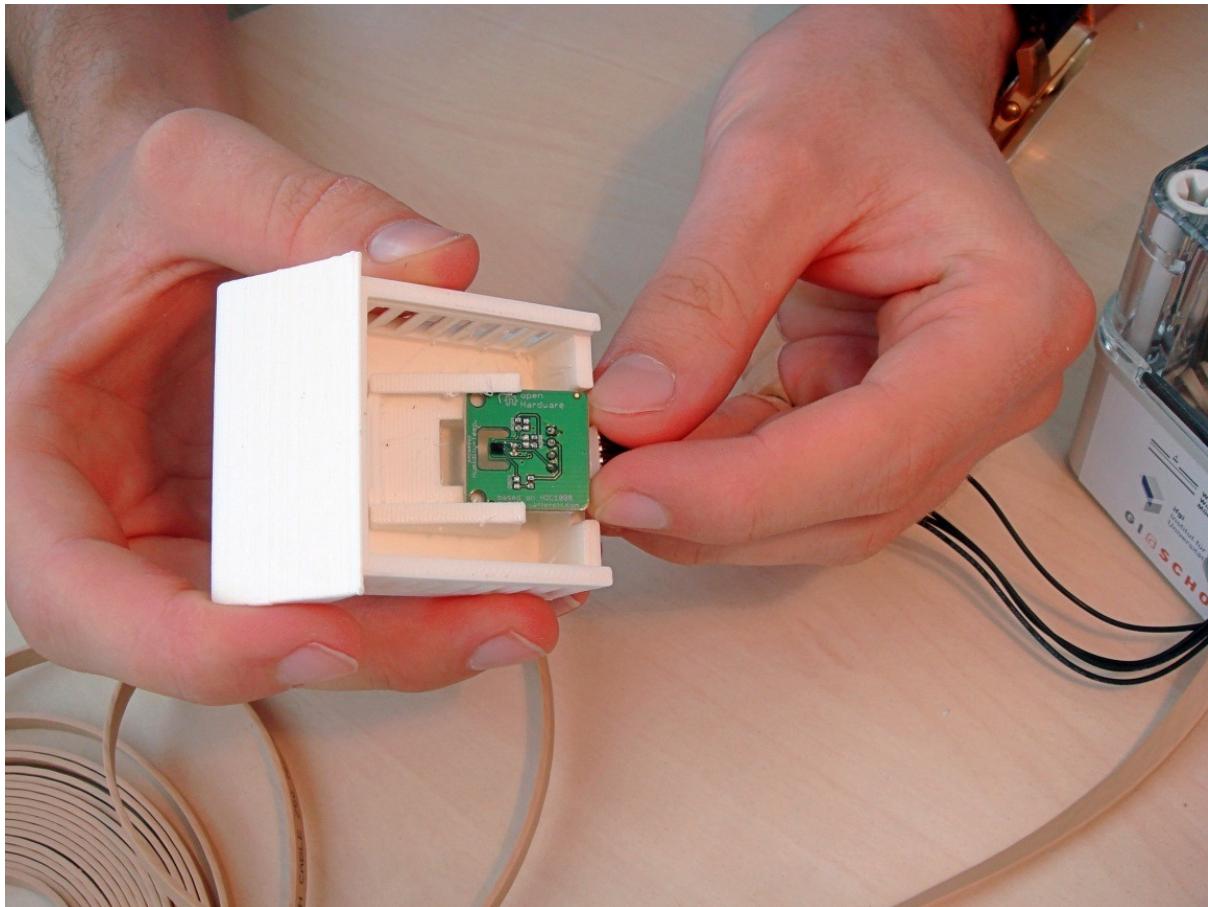


Im Video an der Stelle [05:00](#)

Nun kann das Gehäuse verschlossen werden.

## Temperatursensorgehäuse

Damit der Temperatur- und Luftfeuchtesensor durch Regen und Schmutz geschützt ist, wird er in einem eigenen Gehäuse untergebracht. Dazu den Sensor einfach in das [3D-gedruckte Gehäuse<sup>5</sup>](#) schieben und danach die Klappe aufschieben.



Das Gehäuse kann danach etwa mit Heißkleber an das Haupt-Gehäuse geklebt werden.

Im Video an der Stelle [07:12](#)

## Kabelzugang abdichten

Damit kein Wasser durch die Kabelführung ins Innere des Gehäuses eindringt, muss diese Öffnung abgedichtet werden.  
Dazu eignet sich beispielsweise Dichtungsmasse oder auch Heißkleber:



Im Video an der Stelle 08:08

## Endergebnis

Zuletzt noch den zweiten POE-Adapter mit dem Ende des Ethernetkabels, und diesen mit Router und Netzteil verbinden.

fertig!



Sucht euch einen Standort für die Station, an dem ihr eure Daten aufnehmen wollt. Im besten Falle sollte dieser Standort exponiert unter freiem Himmel sein. Da die Lage der Station allerdings durch Länge der Kabel begrenzt ist, werden die privaten Stationen in der Regel auf einem Balkon oder an einer Häuserwand befestigt.

Achtung: Falls der Stromstecker nach draußen verlängert werden muss, ist unbedingt darauf zu achten, dass eine wetterfeste Verteilerdose verwendet wird! Diese sollte mindestens die **Schutzart IP65<sup>6</sup>** erfüllen.

<sup>1</sup>. [https://raw.githubusercontent.com/sensebox/resources/master/datasheets/datasheet\\_case\\_FIBOX\\_piccolo\\_pc-d-85-t.pdf](https://raw.githubusercontent.com/sensebox/resources/master/datasheets/datasheet_case_FIBOX_piccolo_pc-d-85-t.pdf) ↵

<sup>2</sup>. <http://de.farnell.com/fibox/pc-d-85-t/box-polycarbonat-ip67-deckel-klar/dp/1004124> ↵

<sup>3</sup>. <https://www.thingiverse.com/thing:1523429> ↵

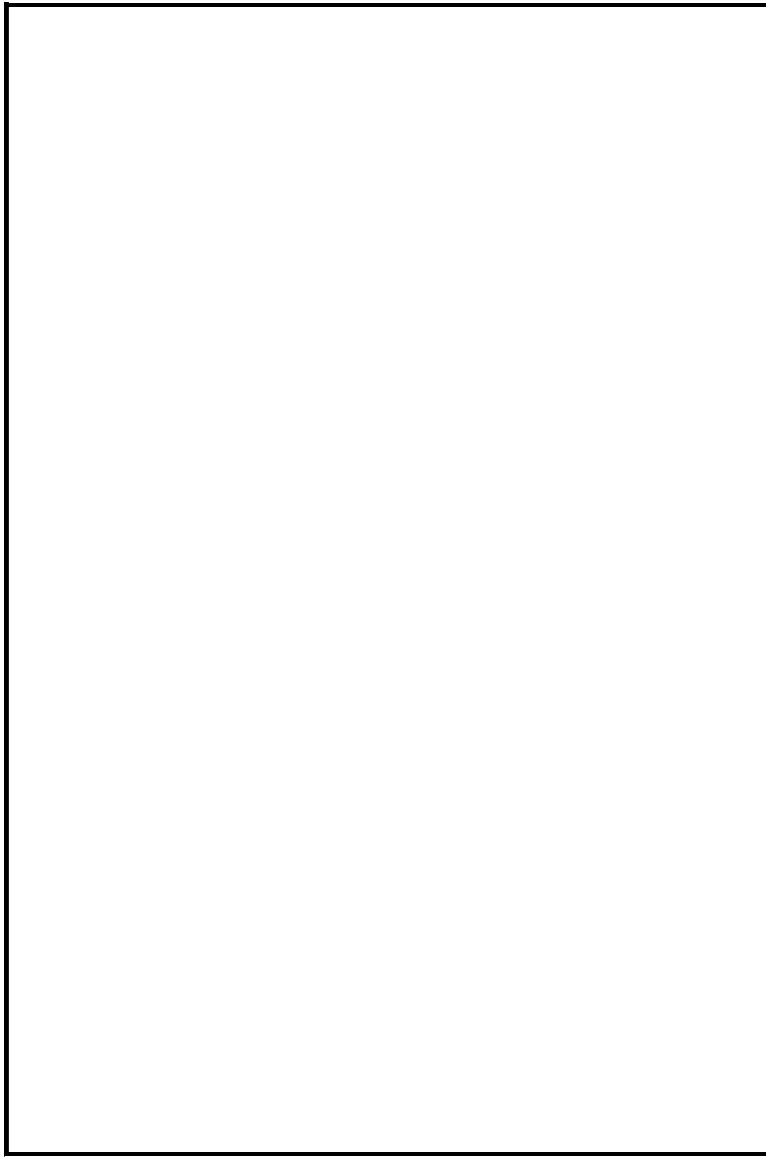
<sup>4</sup>. <https://geizhals.de/digitus-dn-95001-a1214858.html> ↵

<sup>5</sup>. <https://www.thingiverse.com/thing:1523429> ↵

<sup>6</sup>. [http://www.ncps.de/NetCare\\_Trier/Lexikon/IP\\_Schutzarten/](http://www.ncps.de/NetCare_Trier/Lexikon/IP_Schutzarten/) ↵

## senseBox Botanischer Garten

Die senseBox im Botanischen Garten in Münster besteht aus einer erweiterten senseBox:home. Sie misst die üblichen Phänomene wie Temperatur, Luftfeuchtigkeit, Luftdruck, Helligkeit und UV-Index. Zusätzlich werden die Windrichtung, Windgeschwindigkeit, Niederschlagsmenge sowie die Wolkenbedeckung gemessen.



## Komponenten

Die senseBox besteht aus folgenden Komponenten:

- senseBox:home
- [Wetter-Messeinheit<sup>1</sup>](#)
- [senseBox:cloud<sup>2</sup>](#)
- SHT15 Temperatur-/Luftfeuchtesensor

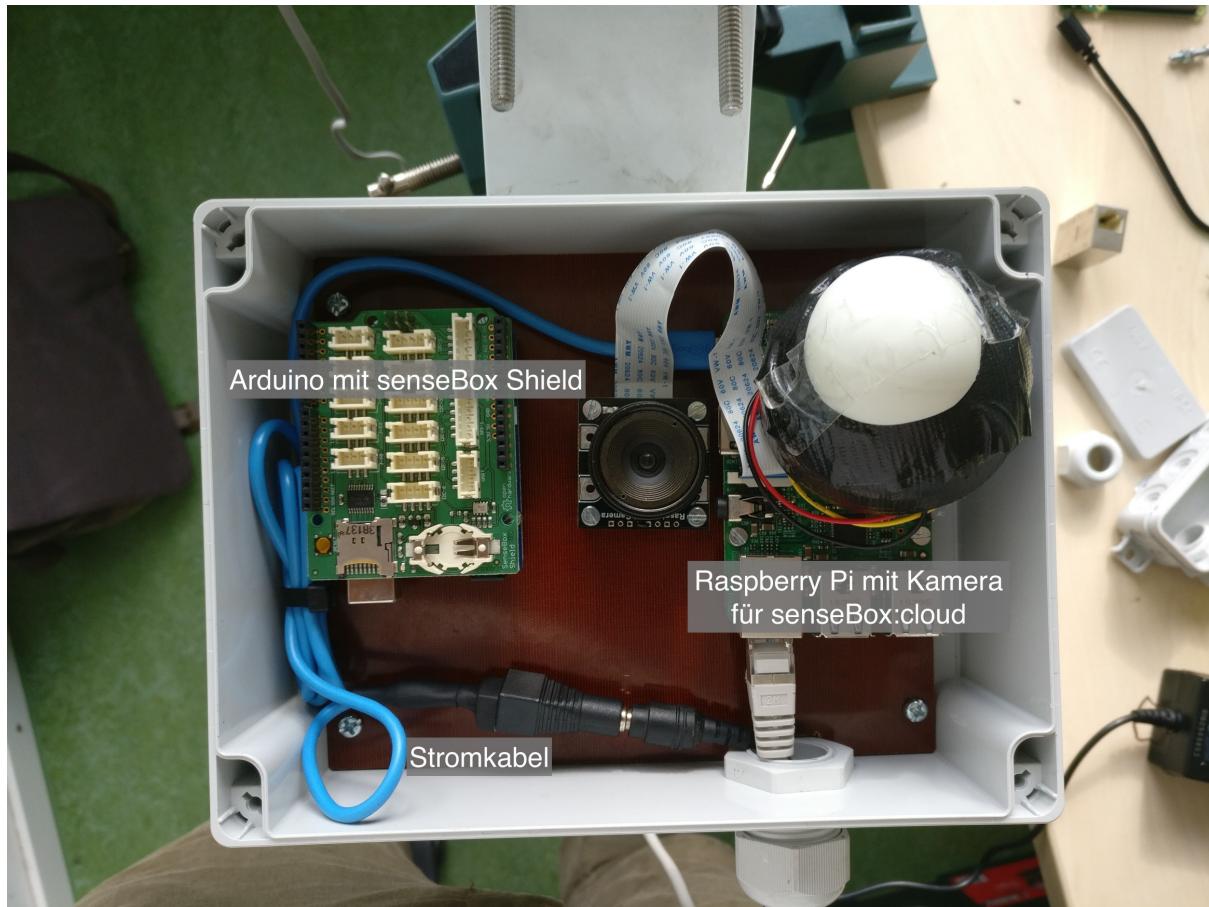
## Aufbau

Alle Komponenten sind an der Wetter-Messeinheit befestigt. Die senseBox:cloud und senseBox:home wurden an einem Arm befestigt, damit die Kamera einen uneingeschränkten Blick auf den Himmel hat.



Gehäuse

Als Gehäuse wird ein Installationsgehäuse mit transparentem Deckel genutzt (in diesem Fall: <https://www.conrad.de/de/installations-gehaeuse-202-x-152-x-90-abs-licht-grau-ral-7035-spelsberg-tg-abs-2015-9-to-1-st-533295.html>). Im Gehäuse sind die senseBox:cloud und die senseBox:home untergebracht.

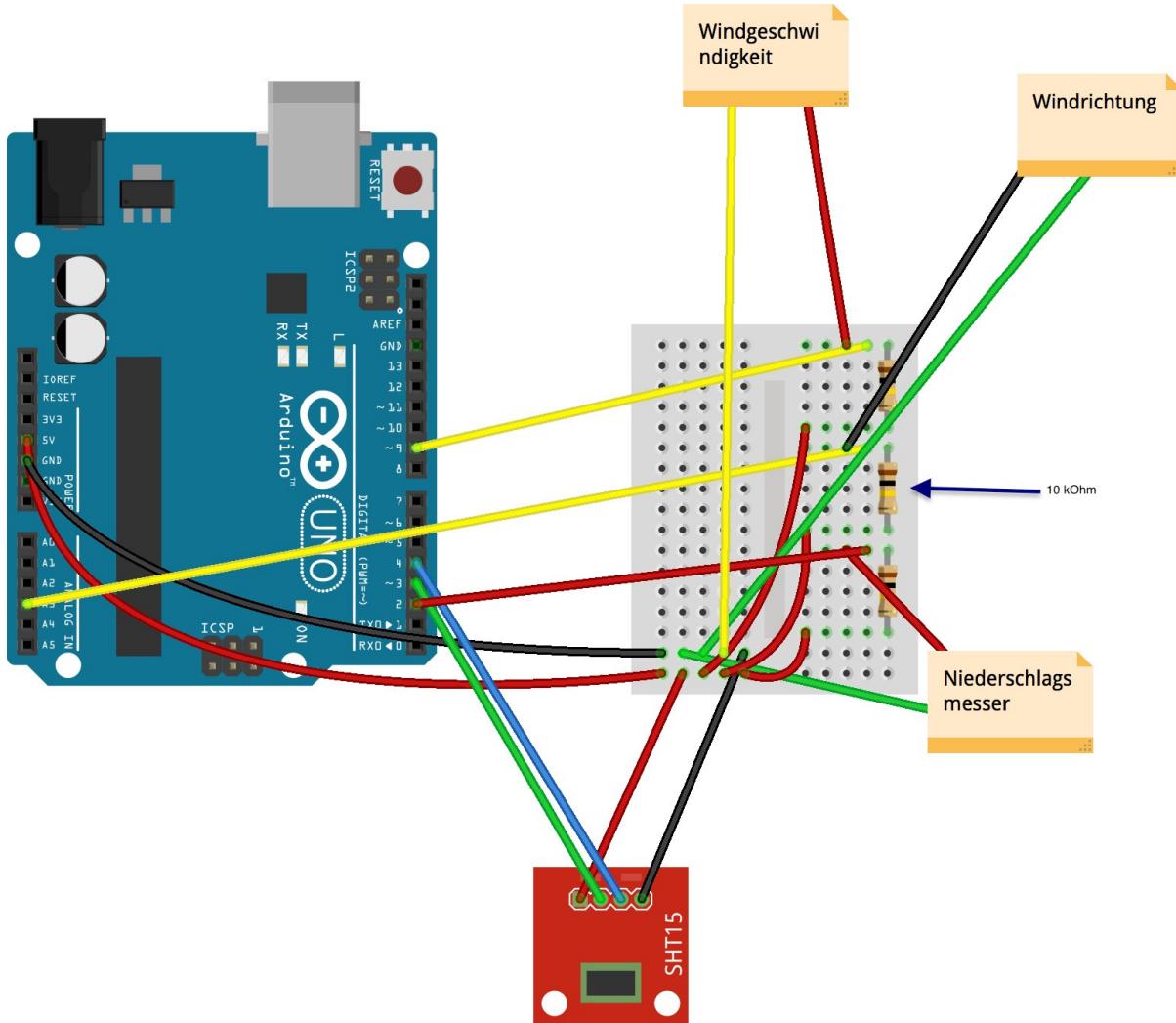


## Stromversorgung

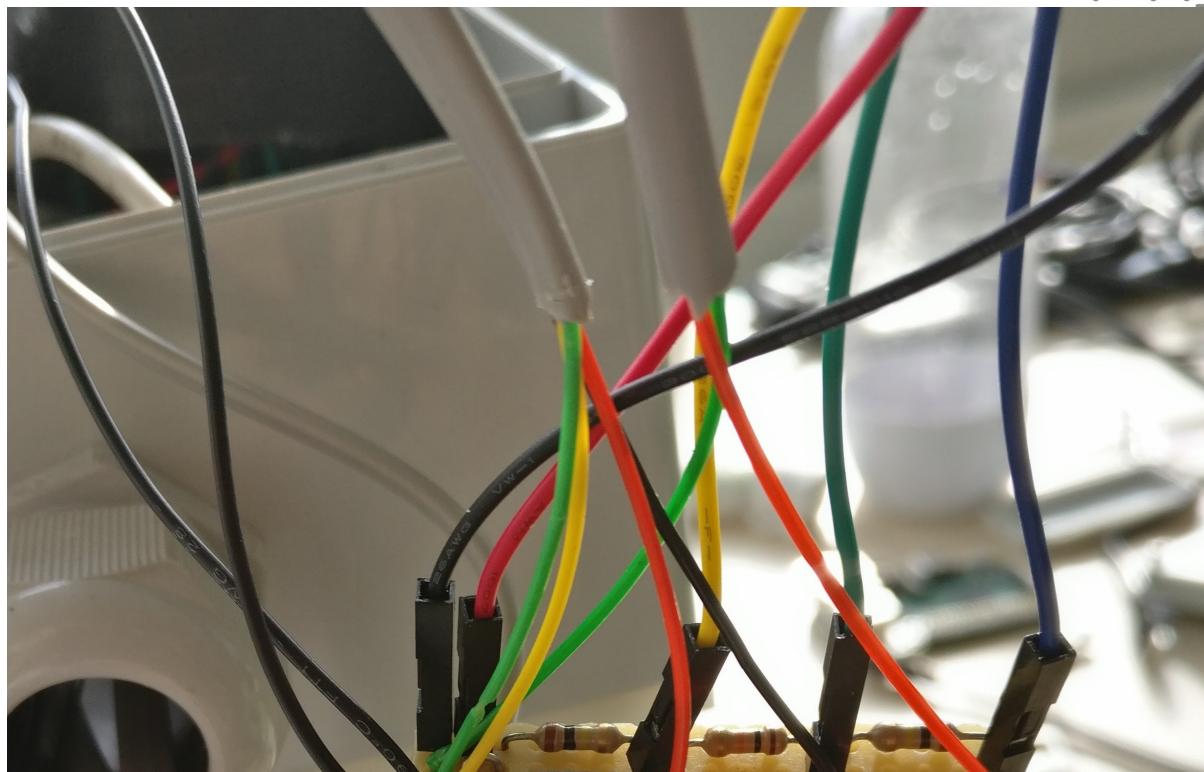
Zur Stromversorgung wird von diesem Netzteil bereitgestellt: <https://www.distrelec.de/de/netzgeraete-mean-well-gs25b05-p1j/p/16997623?q=366457&page=1&origPos=1&origPageSize=25&simi=97.0&no-cache=true>. Dazu wird etwa dieser Adapter benötigt um den Raspberry Pi mit Strom zu versorgen (<https://www.adafruit.com/product/2789>). In diesem Fall wurde ein Adapter jedoch selbst zusammen gelötet.

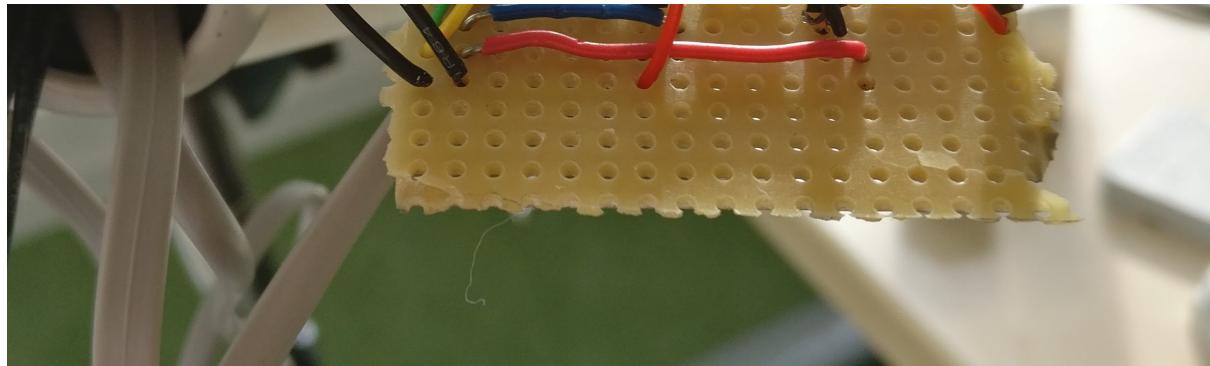
## Wetter Messeinheit

Neben den üblichen Wetterphänomenen misst die senseBox:home auch die Daten der Wetter-Messeinheit. Dazu wurden die Sensoren folgendermaßen verbunden:



fritzing





## Software

Auf dem Raspberry Pi der senseBox:cloud läuft neben dem Code zur Aufnahme der Bilder ebenfalls ein Python Script um die Daten der senseBox:home zu empfangen und zur OpenSenseMap zu senden.

Das Python Script befindet sich [hier](#)<sup>3</sup>

Der Code zur Wolkenbedeckung sowie das Python Script werden beim Boot des Raspberry Pi gestartet.

Die senseBox:home sendet somit die Daten nicht selbst zum OpenSenseMap, sondern zum Raspberry Pi über die serielle Schnittstelle USB.

Der Arduino Sketch befindet sich [hier](#)<sup>4</sup>

## Standort

Die komplette Wetterstation wurde im Botanischen Garten in Münster in etwa 3m Höhe aufgestellt. Sie hängt an einem Masten, an welchem vorher eine Webcam installiert war. Der Masten ist umgeben von Bambusstangen.

---

1. <http://www.watterott.com/de/Wetter-Messeinheit> ↵

2. <https://github.com/felixerdy/senseBox-cloud> ↵

3. [https://github.com/sensebox/resources/blob/master/code/Botanischer\\_Garten/readArduinoPostOSeM.py](https://github.com/sensebox/resources/blob/master/code/Botanischer_Garten/readArduinoPostOSeM.py) ↵

4. [https://github.com/sensebox/resources/blob/master/code/Botanischer\\_Garten/Weatherstation.ino](https://github.com/sensebox/resources/blob/master/code/Botanischer_Garten/Weatherstation.ino) ↵