

Bau einer Zählstation mit Hilfe der senseBox

Seit mindesten 50000 Jahren zählen wir Menschen. Wir zählen unsere eignen Sachen, Menschen, Treppenstufen, den Verkehr und, und und. Wir erhoffen uns dadurch Informationen über uns und unsere Umwelt.

Fragen die wir uns stellen könnten zum Beispiel sein:

- Wie viele Menschen leben in unserer Stadt/Dorf?
- Wie viele Menschen sind auf einer Veranstaltung?
- Wie viele Autos fahren auf einer Straße
- Wie viele Autos fahren auf den Schulhof?
- Wie oft wird ein Vogelnest verlassen und wieder betreten?

Bei all diesen Fragen ist nicht nur der Wert selber interessant, sondern auch, wie er sich über die Zeit hinweg verändert. So kann man Rückschlüsse über das Verhalten des Objektes und Einflüsse auf es untersuchen.

Auf diesem Arbeitsblatt werden wir im besonderen beachten, wie man einen solchen Zähler mit Hilfe der sensebox bauen kann und wie man ihn so baut, dass man nicht selber jedes einzelne zu zählende Objekt aufschreiben muss.

① Suche dir eine Person mit der du zusammenarbeiten möchtest!

- Schaut euch gemeinsam die nächsten Seiten an und überlegt dann, was genau ihr zählen möchtet.

Die Projektidee:

Das wollen wir mit Hilfe der **senseBox** messen:

Wir wollen gerade dieses Phänomen messen, da :

Die Umsetzung mit Hilfe der senseBox:

② Nehmt euch nun eine sensebox!

- Welche Teile braucht ihr für euer Projekt und warum?
- Tragt alle Teile in die untenstehende Liste ein und findet heraus, wozu die jeweiligen Teile gut sind.

Teil	Funktion	Benötigt	Anmerkung

Teile der sensebox

Los geht's!

**Diese Teile braucht ihr:**

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

③ Sucht euch die oben genannten Komponenten heraus.

- Ihr nutzt den Ultraschall-Distanzsensor um mit Hilfe des Schalls die Entfernung von Objekten zu messen. Der Sensor sendet einen Impuls aus und misst die Zeit, bis er das Echo des Impulses wieder empfängt. Aus dieser Zeit wird mit Hilfe der Schallgeschwindigkeit die Entfernung des Objektes berechnet.

**Verkabelung**

Das rote Kabel und das schwarze Kabel sind für die Stromversorgung zuständig(5V)

Das grüne Kabel und das gelbe Kabel sind für die Datenübertragung zuständig.

④ Baut nun die Komponenten zusammen!

- Verbindet den Ultraschallsensor mit einem JST-Kabel mit der senseBox MCU.
- Sucht auf der senseBox MCU den Steckplatz Digital A und steckt das weiße Ende des Kabels hinein.
- Steckt nun den Ultraschallsensor vertikal in die Spalte e auf das weiße Steckbrett der senseBox MCU. Achtet hierbei darauf, dass die Beschriftung HC-SR04 nicht in Richtung der MCU zeigt.
- Verbindet nun die Ultraschallsensor mit dem Digital A Port indem ihr das rote Kabel in der Spalte d an die Stelle steckt, wo auf dem Ultraschallsensor Vcc steht, das grüne Kabel in die Spalte c steckt, wo beim Ultraschallsensor Trig steht, das gelbe Kabel in die Spalte c steckt, wo beim Ultraschallsensor echo steht und das schwarze Kabel in die Spalte d steckt, wo beim Ultraschallsensor Gnd steht.

Wenn der Anschluss erfolgt ist, werdet ihr nun überlegen, wie ihr die Daten visualisieren könnt.

- ⑤ Um die Daten zu visualisieren muss die **sensebox** programmiert werden. Dazu nutzt die Programmierumgebung blockly:
(<https://blockly.sensebox.de/ardublockly/?lang=de&board=sensebox-mcu>)
- Macht euch mit blockly vertraut. Welche Blöcke könnt ihr finden? Was machen diese Blöcke und welche Blöcke braucht ihr für euren Abstandsmesser?
 - Sucht als erstes alle Blöcke heraus, die ihr benötigt um das Display programmieren zu können.



Diese Bausteine braucht ihr

The screenshot shows the Blockly Sensebox IDE interface. On the left is a library of blocks categorized by function (Sensoren, Ausgabe, Display, etc.). The main workspace contains a sequence of blocks: an 'Arduino run first:' block, an 'Arduino loop forever:' block, a 'Display initialisieren' block, a 'Zeige auf dem Display' block, a 'Display löschen' block, and a 'Schriftfarbe Weiß' block. The right panel displays the compiled Arduino C++ source code, which includes headers for SPI, Wire, Adafruit_GFX, Adafruit_SSD1306, and senseBoxIO, and defines the OLED_RESET pin. The setup() function initializes the display and sets the font color to white. The loop() function calls display.display() and display.clearDisplay().

```
{ } Arduino Quellcode

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

#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_RESET);

void setup() {
  senseBoxIO.powerI2C(true);
  delay(2000);
  display.begin(SSD1306_SWITCHCAPVCC, 128);
  display.display();
  delay(100);
  display.clearDisplay();
}

void loop() {
  display.display();
  display.clearDisplay();
}
```

Bausteine für das Display

- ⑥ Ihr habt nun die richtigen Bauteile um Dinge auf dem Display anzeigen zu können, Doch was wollt ihr überhaupt anzeigen?
- überlegt euch welche Sensoren ihr nutzt und wie ihr es schaffen könnt, dass diese bei bestimmten Messungen die Anzahl der Zählung erhöhen
 - schreibt eure Vermutungen in Pseudocode auf und überlegt, ob ihr euren Pseudocode so mit den Blocklyblöcken umsetzen könnt

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Pseudocode

Pseudocode ist ein informelles Werkzeug, das benutzt werden kann, um einen Algorithmus zu planen. Es ist ein verbaler Schritt-für-Schritt-Entwurf deines Codes, der später in eine Programmiersprache umgeschrieben werden kann. Es ist eine Kombination aus menschlicher und Programmiersprache: Es ahmt den Satzbau tatsächlicher Computercodes nach, sorgt sich aber mehr um die Lesbarkeit als um technische Genauigkeit.

**Diese Bausteine braucht ihr**

The screenshot shows the Arduino IDE interface. On the left, the block-based code editor contains the following logic:

- Arduino run first:** A block to "erhöhe Element um 1" (increase Element by 1).
- Arduino loop forever:** A loop containing:
 - A "wenn mache" (if) block with a "falsch" (false) condition.
 - Inside the "if" block, a "wenn mache" (if) block with a "wahr" (true) condition.
 - Inside the "if" block, a "und" (and) block connecting two conditions: "0 >= 0" and "Element = (char)(0)".
 - When the "and" block is true, it triggers a "Schreibe Element" (write Element) block.
- After the "if" block, there is another "Schreibe Element" block.
- Finally, an "als Zeichen" (as character) block is connected to the second "Schreibe Element" block.

On the right, the C++ source code is displayed:

```
{ } Arduino Quellcode

int Element;

void setup() {
}

void loop() {
  if (false) {
  }

  Element += 1;

  Element = (char)(0);

  0 >= 0;

  false;

  if (false) {
  }

  0 <= 0;

  Element = 0;

  true;

  (char)(0);
}
```

At the bottom of the IDE window, it says "Arduino IDE Ausgabe".

Bausteine für den Code

⑦ Passt nun euren Pseudocode an diese Bausteine an.

- Achtet darauf, dass ihr sinnvolle Variablennamen wählt
- Ist der Abstand des Sensors richtig eingestellt? (der Sensor misst in cm und seine Reichweite beträgt ca. 3m)



So könnte euer Programm aussehen:

The screenshot shows the senseBox Blockly IDE interface. The main workspace contains a program with the following logic:

- Arduino run first:** Display initialisieren
- Arduino loop forever:**
 - wenn** (if): Ultraschall Abstandssensor an Port A $\geq$ 40 (Trigger D1, Echo D2)
 - mache** (do): Schreibe Spurfrei = wahr als Boolean
 - wenn** (if): Spurfrei = wahr und Ultraschall Abstandssensor an Port A $\leq$ 40 (Trigger D1, Echo D2)
 - mache** (do): erhöhe Fahrzeuge um 1; Schreibe Spurfrei = falsch als Boolean
 - Zeige auf dem Display** (Show on display): Schriftfarbe Weiß, Schriftgröße 1, x=0, y=0, Wert Fahrzeuge
 - Display löschen** (Clear display)

The right sidebar shows the Arduino Quellcode:

```
{ } Arduino Quellcode

#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <senseBoxIO.h>
#include "SenseBoxMCU.h"

boolean Spurfrei;
int Fahrzeuge;

#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_RES

Ultrasonic UltrasonicA(1,2);

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

void loop() {
  if (UltrasonicA.getDistance() >
```

At the bottom, there is a green bar labeled "Arduino IDE Ausgabe".

Beispielhafte Programmlösung

- ⑧ Nun werdet ihr das Display an die **sensebox** anschließen
 - Steckt dafür den Anschluss des Displays an einen der I2C-Wire PLätze
- ⑨ Wenn euer Code fertig ist und ihr alle benötigten Sensoren an der **sensebox** befestigt habt solltet ihr nun euren Code auf die **sensebox** laden und testen, ob alles so funktioniert, wie ihr es euch vorgestellt habt.
Um den Code hochzuladen:
 - schließt die **sensebox** mit dem USB-Kabel an den Computer an
 - kompiliert euren Code indem ihr bei der Programmieroberfläche Blockly auf den Haken oben rechts (Sketch kompilieren) drückt
 - tätigt einen Doppelklick auf dem roten reset-Knopf der **sensebox**
 - zieht den heruntergeladenen Code auf die **sensebox**, die als Wechseldatenträger auf dem Computer auftauchen sollte
- ⑩ Um eure Daten über einen längeren Zeitraum zu dokumentieren tragt sie in die untere Tabelle ein.

[illegible]

Messungen über einen längeren Zeitraum