Herzlich willkommen!

Erhebung, Analyse und Visualisierung digitaler Daten (EAVD)

(*Prozedurale Programmierung* / *Einführung in die Informatik*)

Prof. Dr. Sebastian Zug & Prof. Dr. Bernhard Jung

TU Bergakademie Freiberg, Wintersemester 2025/26

Organisatorische Einordnung / Motivation

Das vorliegende Material bedient die bisherigen Veranstaltungen

- *Prozedurale Programmierung* und
- *Einführung in die Informatik*

und firmiert diese als neukonzeptionierte übergreifenden Vorlesung *Erhebung, Analyse und Visualisierung digitaler Daten*.

> Die Veranstaltung richtet sich an Nicht-Informatiker aus verschiedenen ingenieurwissenschaftlichen Disziplinen mit keinen oder geringen Programmiervorkenntnissen.

Worum geht es?

Modulhandbuch

Qualifikationsziele / Kompetenzen:

Mit der erfolgreichen Teilnahme an der Veranstaltung sollen die Studierenden:

- verstehen, was Algorithmen sind und wie konkrete wissenschaftliche Aufgaben algorithmisch abgebildet werden können,
- Konzepte der prozeduralen und objektorientierten Programmierung in Python und C++ anzuwenden
- in der Lage sein, praktische Herausforderungen der Datenaggregation und Verarbeitung zu identifizieren und Umsetzungen zu realisieren
- Werkzeuge der Programmierung einordnen und nutzen zu können
- Datenstrukturen und algorithmische Konzepte anwenden zu können und über Wissen ausgewählter Standardalgorithmen verfügen.

Lernziele der Vorlesung im Kontext der Bloom'schen Taxonomie

Einordnung	Studierende ...
Entwickeln	• ... entwerfen eigenständige Lösungsansätze für einfache wissenschaftliche Datenerhebungen / Analysen. • ... kombinieren verschiedene Programmierkonzepte zu funktionsfähigen Anwendungen.
Bewerten	• ... können die Unterschiede der behandelten Programmiersprachen beurteilen . • ... beurteilen die Qualität eines Codes anhand grundlegender Metriken.
Analysieren	• ... können einen einfachen, fremden Code systematisch erschließen.
Anwenden	• ... realisieren kleiner Beispiele zur Datenerfassung bzw. -analysepipeline. • ... sind in der Lage eine wissenschaftliche Fragestellung auf eine Datenerfassungs- und Analysepipeline abzubilden. • ... wenden die Basis-Techniken der Codeentwicklung, des Debuggings und der Dokumentation an.
Verstehen	• ... erklären Basiskonzepte objektorientierter Programmierung (Vererbung, Kapselung). • ... können die Elemente prozeduraler Programmierung (Schleife, Verzweigung, Sprung, Funktion) beschreiben.
Erinnern	• ... kennen die grundlegende Syntaxelemente der behandelten Programmiersprachen. • ... beschreiben Grundkonzepte der Informatik wie Algorithmus, Sprache, Speicher usw.

Unsere Motivation

- **Anwendungssicht**

  Wir möchten Sie in die Lage versetzen einfache Messaufgaben (mit einem Mikrocontroller) zu entwerfen und die Daten auszuwerten.

- **Algorithmische Perspektive**

  Wir möchten Sie dazu ertüchtigen den Algorithmusbegriff der Informatik zu durchdringen und anwenden zu können.

- **Konzeptionelle Perspektive**

  Sie erlernen grundlegende Elemente der prozeduralen und der objektorientierten Programmierung.

- **Umsetzungssicht**

  Wir vermitteln Grundkenntnisse in den Programmiersprachen C++ und Python.

Wir sind kein "Programmierkurs" sondern vermitteln die Konzepte und Grundlagen der Softwareentwicklung sowohl abstrakt als auch praktisch.

	Phase 1	**Phase 2**
Programmiersprache	C++	Python
Framework / Packages	"Messrechner"	pandas/numpy/matplotlib
Ziel	Datenerhebung	Datenauswertung
Plattform	PC / Mikrocontroller	PC

Warum das Ganze?

Merke: Wissenschaftliches Arbeiten ist im Bereich der Natur- und Ingenieurwissenschaften ohne den Rechner (fast) nicht denkbar.

1. Erstellen einer Hypothese
2. Entwurf eines Vorgehens für die systematische Untersuchung der Fragestellung
3. Vorbereitung des Experimentes
4. Durchführung
5. Analyse UND Bewertung der erlangten Daten

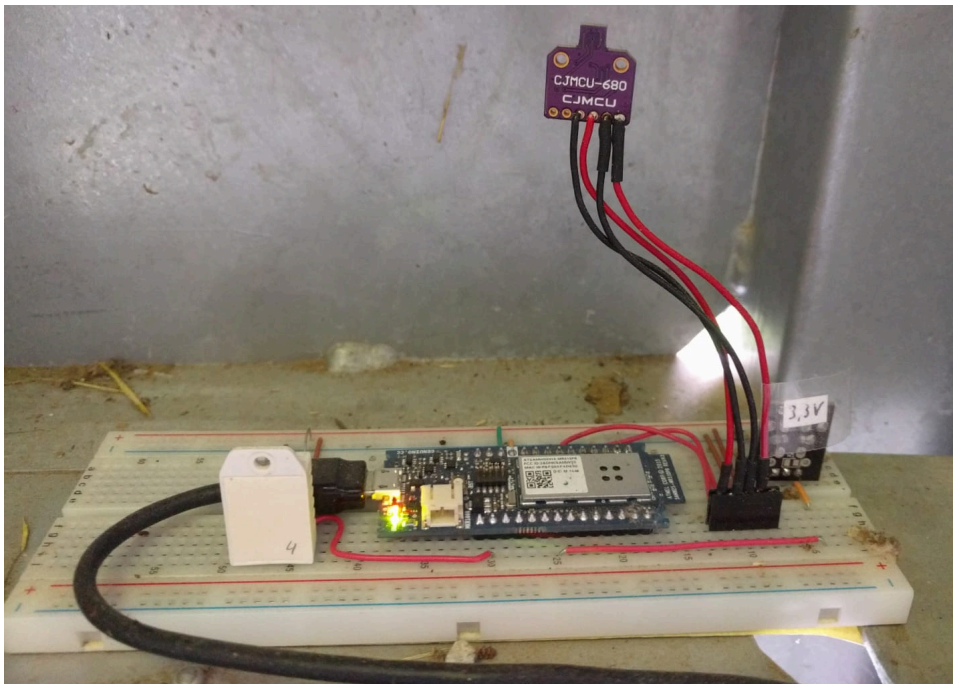
Beispiel

1. Hypothese: Am Wochenende scheint die Sonne häufiger als unter der Woche.

2. Konzeption eines Experimentes: ...

Frage: Wie würden Sie vorgehen?

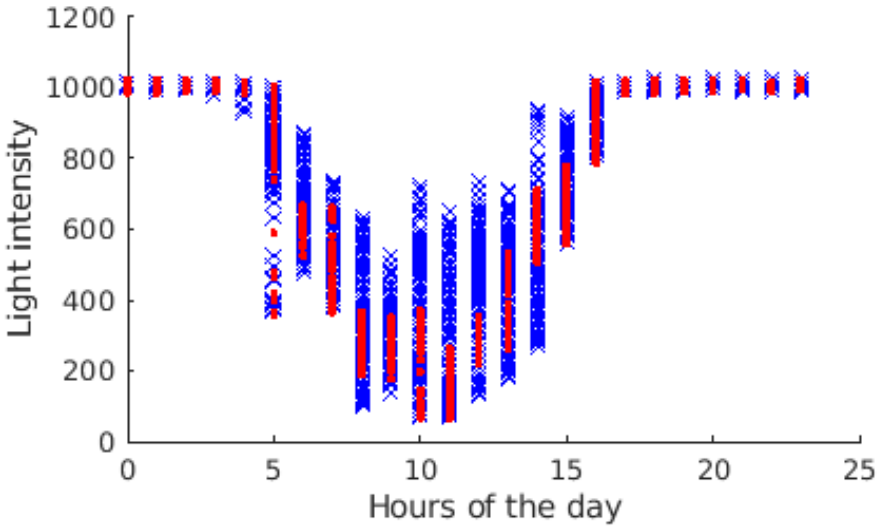
3. Vorbereitung/4. Durchführung des Experimentes



Prototypische Wetterstation im Keller des Humboldt-Baus

5. Analyse UND Bewertung

Das Diagramm zeigt die Darstellung der Lichtintensität über den Stunden eines Tages für eine Woche. Blau sind die Wochentage markiert, rot der Samstag und der Sonntag.



Frage: Welche Kritik sehen Sie an dieser Methodik?

Weitere Beispiele:

- Erfassen und Analysieren Sie das Insektenaufkommen auf dem Campus der TU Freiberg!
- In welchem Zusammenhang steht der Energieverbrauch der Gebäude der Bergakademie mit der Außentemperatur?
- An welchen Stellen in Freiberg muss unser Roboter häufiger stoppen, um Passantinnen queren zu lassen?

Keine Angst vor Code!

Das Beispiel zeigt ein "Hello World" Beispiel in C++.

Hello World - Beispiele Phase 1

main.cpp

```cpp
#include <iostream>
using namespace std;

int main (){
  int i = 0;
  int max = 0;

  cout << "How many hellos: ";
  cin >> max;

  for(i=0; i<max; i++)
    cout << "Hello, world " << i << endl;

  return 0;
}
```

```
How many hellos: How many hellos:
```

Die Programmiersprache Python offeriert eine Vielzahl von Paketen für unterschiedlichste Zwecke. Wir werden uns auf die Visualisierung und Datenauswertung konzentrieren.

Hello World - Beispiele Phase 2

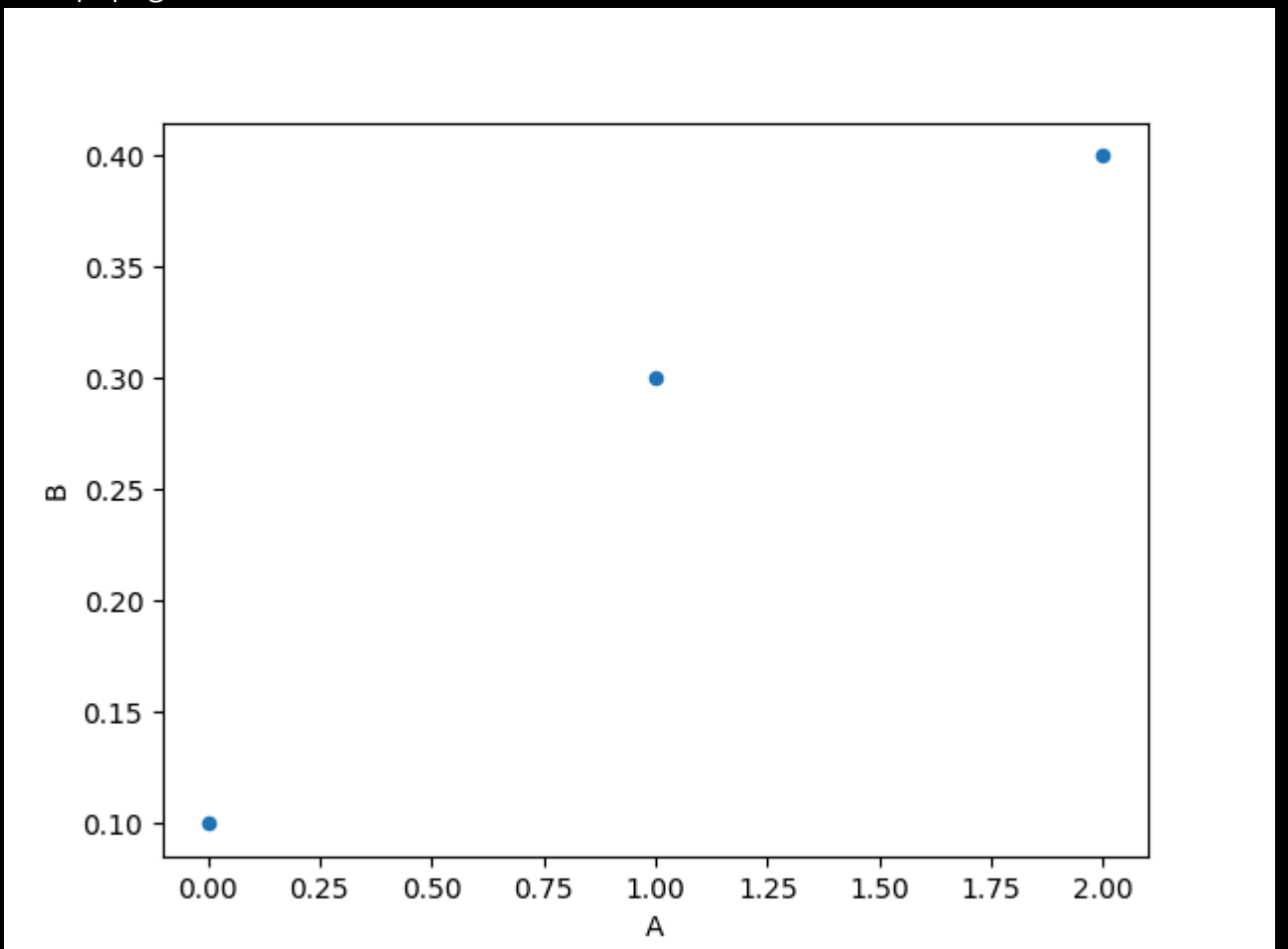
data.csv

```
A,B,C
0,0.1,3
1,0.3,5
2,0.4,2
```

readCSV.py

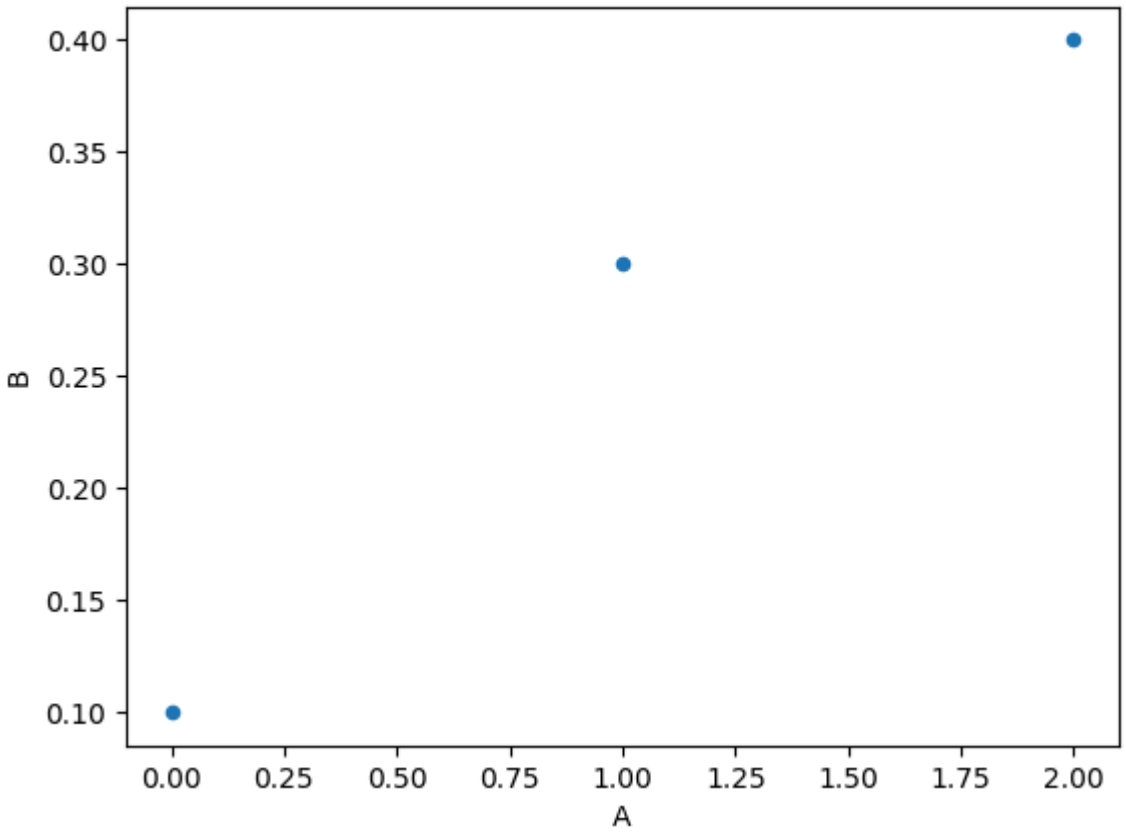
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('data.csv', header = 0)
df.plot.scatter(x='A', y='B')
plt.savefig('temp.png')
```

temp.png



temp.png main.py data.csv

temp.png



temp.png main.py data.csv

"Python ist ja nett, aber C++ ist ..."

Foreneinträge aus Veranstaltungen anderer Hochschulen:

> *"Viele haben bei uns wegen dem Info-Grundlagenmodul gewechselt. Allerdings hängt das auch von dir und deinem Talent ab. Das Tempo ist rasant. Jede Art von Vorerfahrung hilft dir eigentlich sehr. Also wenn du noch Zeit hast vorm Studienbeginn, schnapp dir ein gutes Buch zur gelehrten Sprache, und fange schonmal bissl an kleine Sachen zu programmieren."*
>
> *"Ich habe es gerade irgendwie selbst gelöst, aber keine Ahnung warum es funktioniert hat."*

Frage: Was sind die besonderen Herausforderungen bei der Programmierarbeit?

- Abstrakte Denkweise
- Penible Beachtung der Syntax
- Ungewohnte Arbeitsmittel

Warum ist Syntax so wichtig? Computer verstehen nur exakte Anweisungen - ein kleiner Tippfehler kann das gesamte Programm zum Absturz bringen!

fehlerhaft.py

```
  print("Einfaches Programm zur Temperaturumrechnung")

def celsius_to_fahrenheit(celsius):
    fahrenheit = celsius * 9/5 - 32      # C * 9/5 + 32 (aus Wikipedia
    return fahrenheit

for i in range(0, 35, 5):
    print("Temperatur in Celsius:", i, end=""
    result = Celsius_to_Fahrenheit(i)
    print(" = ", result, "Grad Fahrenheit")
```

```
  File "/tmp/tmpn5giak1w/main.py", line 1
    print("Einfaches Programm zur Temperaturumrechnung")
IndentationError: unexpected indent
```

main.py

```
  File "/tmp/tmp79idj_92/main.py", line 1
    print("Einfaches Programm zur Temperaturumrechnung")
IndentationError: unexpected indent
```

main.py

Entdecken Sie die 3 Syntaxfehler! Diese kleinen Fehler verhindern, dass der Computer das Programm überhaupt ausführen kann.

Aufgabe Was fällt Ihnen neben den Syntaxfehlern auf?

Wo stehen Sie?

Wer von Ihnen hat bereits Programmiererfahrung? Was würden Sie auf die Frage antworten, wie viele Codezeilen Sie bereits geschrieben haben?

- ◯ Keine einzige Zeile
- ◯ 20 Zeilen in der Schule
- ◯ n Zeilen in eigenen Programmierprojekten
- ◯ sehr viele Zeilen in Open Source Projekten

Organisatorisches

Wer sind *wir*?

Name	Email	Rolle
Prof. Dr. Sebastian Zug	sebastian.zug@informatik.tu-freiberg.de	Vorlesungen bis Februar
Prof. Dr. Bernhard Jung	bernhard.jung@informatik.tu-freiberg.de	Vorlesungen ab Januar
Florian Richter	flo.richter@informatik.tu-freiberg.de	
Johannes Kohl	johannes.kohl@informatik.tu-freiberg.de	
Johannes Vater	johannes.vater@informatik.tu-freiberg.de	
Adrian Köppen	adrian.koeppen@student.tu-freiberg.de	Tutor

Strukurierung der Veranstaltung

Nr.	Datum		Inhalt
0	20.10.2024		Motivation, Organisation
1	27.10.2024	C++	C++ Programmstrukturen / Entwicklungsprozess
2	03.11.2024	C++	Datentypen / Ein- und Ausgabe
3	10.11.2024	C++	Kontrollstrukturen
4	17.11.2024	C++	Zeiger und Arrays
5	24.11.2024	C++	Funktionen
6	01.12.2024	C++	Objekte
7	08.12.2024	C++	Vererbung
8	15.12.2024	C++	Anwendungen
9	05.01.2025	Python	Python Grundlagen
10	12.01.2025	Python	Python Grundlagen
11	19.01.2025	Python	Objekte
12	26.01.2025	Python	Visualisierung
13	02.02.2025	Python	Datenanalyse
14	09.02.2025	Python	Übergreifende Anwendungen

Achtung: C++ wird ohne die spezifische Verwendung des Mikrocontrollers verwendet. Vielmehr erfolgt die Programmierung auf dem Desktoprechner. In freiwilligen Tutorien haben Sie Gelegenheit die Hardware auszuwechseln und mit dem Mikrocontroller zu arbeiten.

Übungen

Die Übungen vertiefen das erlernte anhand praktischer Programmieraufgaben:

```
1. Maximumsberechnung:
    a) Nach der Eingabe zweier reellen Zahlen ist mit Hilfe einer if-
       Anweisung das Maximum zu bestimmen und auszugeben.
    b) Bestimmen Sie nach einer Programmerweiterung das Maximum von
       drei beliebigen reellen Zahlen, wobei zur Problemlösung eine
       geschachtelte if-Anweisung eingesetzt werden soll.
   Bei Gleichheit der drei Zahlen soll zusätzlich eine Ausschrift
   „Zahlen gleich“ ausgegeben werden.
```

Frage: Ist jemand noch nicht für die Übungen eingeschrieben?

Empfehlung: Für die Arbeit am Programmcode empfehlen wir den freien Editor Visual Studio Code. Dieser kann sowohl für die Arbeit mit den C++ als auch Python Programmbeispielen genutzt werden.

Mikrocontroller Tutorials

Ein Mikrocontroller-Tutorial wird im Dezember / Januar angeboten. Es dient interessierten Studierenden der weiteren Vertiefung ihrer Kenntnisse.

Mikrocontrollerbezogene Inhalte sind nicht Gegenstand der Prüfungen!

Vorlesungsmaterialien

Die Vorlesungsunterlagen selbst sind unter

https://tubaf-ifi-liascript.github.io/prozprog.html

verfügbar. Diese können entweder in der Markdown-Syntax oder als interaktives Dokument betrachtet werden.

Achtung! Es handelt sich um "lebende" Materialien.

- der Inhalt wird sich ggf. anhand Ihrer Verbesserungsvorschläge verändern
- die Dokumente enthalten eine Vielzahl von ausführbarem Code.

Sie finden die PDF Versionen der Vorlesungen im SHRIMP Portal der Universität Leipzig.

Das Passwort wird in der Vorlesung bekannt gegeben.

Zeitaufwand

Der Zeitaufwand beträgt 180h und setzt sich zusammen aus 60h Präsenzzeit und 120h Selbststudium. Letzteres umfasst die Vor- und Nachbereitung der Lehrveranstaltungen, die eigenständige Lösung von Übungsaufgaben sowie die Prüfungsvorbereitung.

Sie müssen, insbesondere wenn Sie noch keine Programmiererfahrung haben, ggf. deutlich mehr Zeit in den Kurs investieren. Dies macht gemeinsam mehr Spaß als allein!

Prüfungsinhalte

Die Prüfung besteht aus einer Klausur und ggf. einer praktischen Arbeit (Einführung in die Informatik).

- *Entwickeln Sie ein Programm, das ...!*
- *Beschreiben Sie die Funktionsweise des folgenden Programms ...!*
- *Welchen Wert gibt das folgende Programm in Zeile x aus?*
- *Finden Sie alle syntaktischen und logischen Fehler im nachfolgenden Code*
- *Bewerten Sie folgenden Aussagen: ...*

Studierenden, die an der Einführung in die Informatik (7 LP), realisieren neben der Klausur eine praktische Programmieraufgabe. Diese wird mit den Übungsbetreuern abgestimmt.

Wie können Sie zum Gelingen der Veranstaltung beitragen?

- Stellen Sie Fragen, seinen Sie kommunikativ!

Hinweis auf OPAL Forum!

- Organisieren Sie sich in Arbeitsgruppen!

Hinweis auf Repl.it

- Machen Sie Verbesserungsvorschläge für die Vorlesungsfolien!
- Definieren Sie mit uns Anwendungsfälle, die die Übertragbarkeit des erworbenen Wissens auf Ihre Fachdisziplinen deutlich machen.
- Sprechen Sie uns gern wegen "Bastelbedarf" für ein eigenes Projekt an!
- Kommentieren Sie die Vorlesungsunterlagen unter https://app.shrimpp.de/pod/90o.1.1.XxQ9

Beispiel der Woche

Aufgabe: Wann erscheinen Studierende in der Vorlesung? Gibt es da Muster - *Der frühe Vogel*, *rechtzeitig aber knapp*, *In den ersten 10 Minuten passiert sowieso nichts!*

Wir starten unsere Demo mit einem Ultraschallsensor, der die Entfernung von Personen misst. Der folgende Code misst die Entfernung und gibt diese über die serielle Schnittstelle aus.

```
#define PIN_TRIGGER 12
#define PIN_ECHO    13

unsigned long duration;
unsigned int distance;

void setup()
{
  Serial.begin(9600);
  pinMode(PIN_TRIGGER, OUTPUT);
  pinMode(PIN_ECHO, INPUT);
}

void loop()
{
  digitalWrite(PIN_TRIGGER, LOW);
  delayMicroseconds(2);

  digitalWrite(PIN_TRIGGER, HIGH);
  delayMicroseconds(10);

  duration = pulseIn(PIN_ECHO, HIGH);
  distance = duration/58;
  Serial.println(distance);
```

```
  delay(100);
}
```

Der Mikrocontroller misst die Zeit \(t \), die ein Schallsignal benötigt, um vom Sender zur reflektierenden Oberfläche und wieder zurück zu gelangen.

$$s_{\text{hin+zurück}} = v \cdot t$$

wobei

- $s_{\text{hin+zurück}}$ die gesamte Laufstrecke (Hin- und Rückweg) ist,
- v die Schallgeschwindigkeit in Luft (in cm/µs),
- t die gemessene Zeit (in µs).

Da uns nur die **einfache Entfernung** (Hinweg) interessiert, gilt:

$$s = \frac{s_{\text{hin+zurück}}}{2} = \frac{v \cdot t}{2}$$

Die Schallgeschwindigkeit in Luft beträgt bei 20°C:

$$v = 343\,\text{m/s} = 0.0343\,\text{cm/µs}$$

Damit folgt:

$$s = \frac{0.0343 \cdot t}{2} = 0.01715 \cdot t$$

Die Entfernung in Zentimetern ergibt sich somit zu:

$$s_{\text{cm}} = \frac{t}{58.3}$$

Wie geht es weiter ... im Projektordner wartet die Lösung auf Sie!