Co-funded by the Erasmus+ Programme of the European Union

‘We are the makers - IoT’ Learning Scenario: Glasfaser-Kommunikation mit dem Adafruit CPX Express

Autor: Thomas Jörg, Johannes-Kepler-Gymnasium Weil der Stadt

Das vorliegende Tutorial wurde entwickelt und getestet im Fach IMP (Informatik-Mathematik-Physik) eines Baden-Württemberger Gymnasiums mit Schülern aus der 8.Klasse. Es spiegelt die Erfahrung mit talentierten und engagierten Schülern wieder, die im Unterricht in Zweiergruppen die Skripten selbst entwickelt haben. Die Einheit wurde unterrichtet nach einer Unterrichtseinheit zur Netzwerktechnologie.

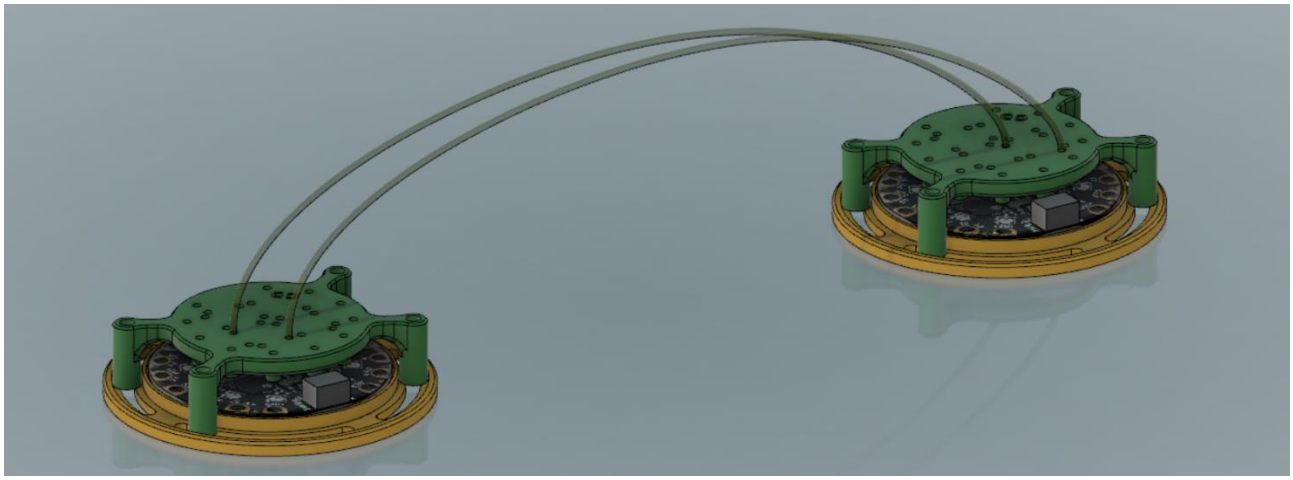
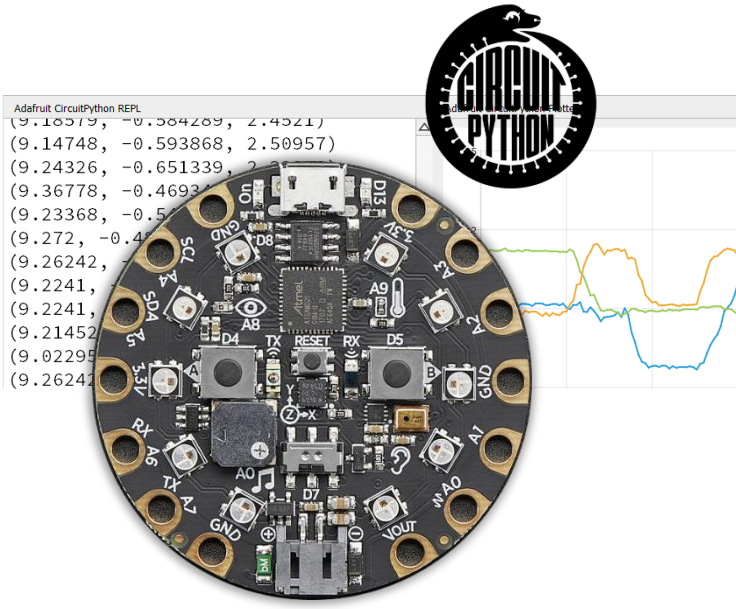


Figure 1: Prototype of a glass fibre communication setup of two microcontrollers



IoT bedeutet Vernetzung von computergesteuerten Geräten. Oftmals ist mit Netzwerk-Kommunikation gemeint, dass sich die Geräte drahtlos, also über Funk ihre Informationen austauschen. Tatsächlich zählen aber auch klassische Trägermedien wie die Glasfaser-Verbindung dazu.

Der didaktische Vorteil, Lichtwellenleiter für die Einführung in die Kommunikationstechnologie zu nutzen, ist die Sichtbarkeit des Informationsaustauschs: Man kann die Lichtimpulse sehen, mit denen Informations-Bits ausgetauscht werden. Ein simples physikalisches Prinzip, nämlich die Totalreflexion, reicht aus, um die moderne Glasfaserkommunikation zu verstehen.

Number of project: 2017-1-DE03-KA201-035615

1. Titel des Scenarios	**Netzwerkkommunikation von IoT-Geräten: Glasfaser-Programmierung**
2. Zielgruppe	14 - 16 Jahre
3. Dauer	Minimal 7 Wochen mit jeweils 2-3 Unterrichtsstunden pro Woche
4. Lerngegenstände, die während des Unterrichts besprochen werden	▪ Sensorik und Aktorik beim Informationsaustausch zwischen Digitalgeräten ▪ Prinzip der protokoll- und paketbasierten Neztwerk-Kommunikation ▪ Anwendung der Totalreflexion zum Transport von Lichtimpulsen. ▪ Programmierung von python-basierten Mikrocontrollern in Kleingruppen von je zwei Schülern
5. Erwartete Lernergebnisse	▪ Wie funktioniert ein IoT-System? ▪ Wie kann man Netzwerkkommunikation strukturieren und implementieren? ▪ Wozu benötigt man ein Kommunikations-Protokoll? ▪ Wie funktioniert paketbasierte digitale Kommunikation?
6. Methoden	In this scenario students will construct, build and program an interactive biosignal device from scratch by themselves. Students will also use the Arduino Serial Monitor and Serial plotter for visualizing and plotting biofeedback.
7. Setting	▪ Ein Klassensatz von Glasfaserkabeln um die Mikrocontroller zu verbinden ▪ Ein Klassensatz von Adafruit CPX-Microcontrollern, ▪ Diese CPX-Controller werden mit Circuitpython programmiert ▪ Jeder Schüler erhält für seinen CPX-Mikrocontroller einen Laptop mit vorinstalliertem Mu-Editor ▪ Jeder CPX wird in eine 3D-gedruckte Einhausung eingebaut, welche die genauer Ausrichtung der Glasfasern sicherstellt. ▪ Jeder Schüler schreibt ein Protokoll über seine Projektarbeit

8. Werkzeuge, Materialien und Resourcen

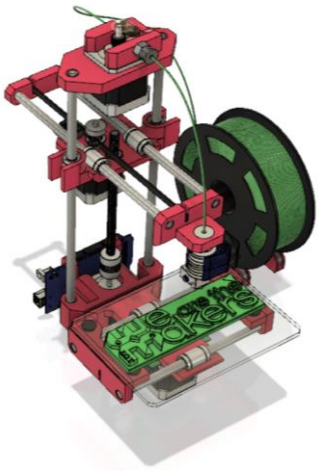


Figure 2:Symbol of 3D-Printer

3D-Drucker

Etwa 2-3 3D-Drucker sind nötig, damit die Schüler ihre eigenen CPX-Gehäuse drucken können. Selbstverständlich können die Schüler auch ihre eigenen Gehäuse selbst entwickeln und drucken.

3d gedruckte Komponenten:

Zum leichteren Einstieg werden zu dieser Unterrichtseinheit alle Teile im .stl-format und als Autodesk Fusion 360 Files mitgeliefert:

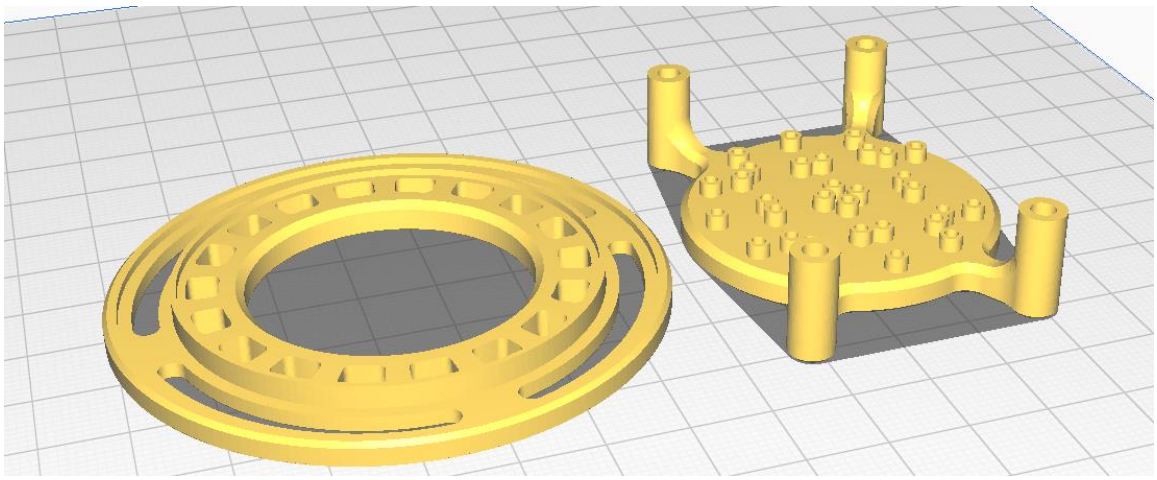
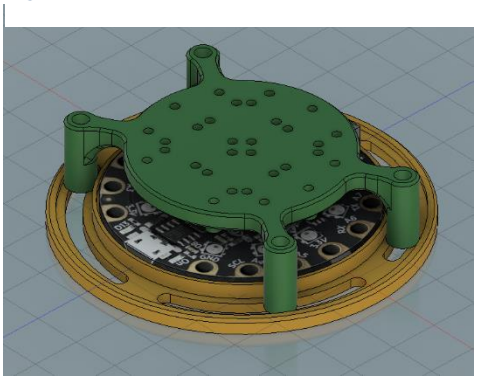


Figure 3: STL-File.Druckzeit für beide Teile etwa 1 Stunde

Die beiden Teile sind so dimensioniert, dass ein CPX mit etwa 0,5mm Spiel in den unteren Teil hineingelegt warden kann. Oberer und unterer Teil werden mit M3-Schrauben miteinander verbunden: Empfohlene Länge der Schraube ist 25mm. Die Teile können mit Flügelmuttern arretiert werden, sodass sich der Mikrocontroller schnell ein- und ausbauen lässt.

Figure 5: M3 Schraube und Flügelmutter

Figure 4: Preview der Fusion-Datei



Die Einhausung ist dafür gedacht, die Glasfaserkabel leicht einzustecken und sie optimal oberhalb der LEDs und des Helligkeitssensors zu positionieren.

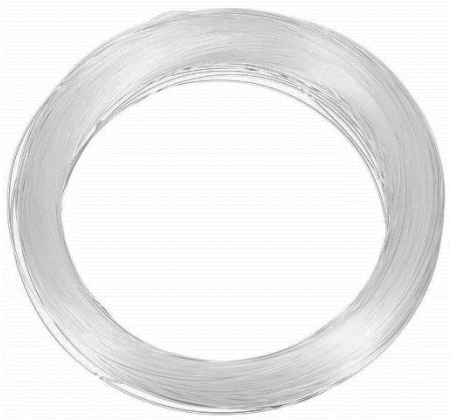


Figure 6:Glasfaser PMMA (Acrylglas)

Als Glasfaser wird ein 1,5mm PMMA-Kabel verwendet. Man erhält diese Fasern in Rollen von etwa 100Metern Länge für etwa 20 Euro. Jede Schülergruppe, die zwei CPX miteinander vernetzen will, benötigt 2 mal 1 Meter Kabel. Ein Kabel als Sendeleitung und ein Kabel als Empfangsleitung. Mit einer solchen Rolle können also mehrere Klassen mit Glasfasern günstig versorgt warden.

Die 1,5mm Durchmesser stellen sicher, dass die Schüler sich nicht so leicht verletzen können. Außerdem sind Fasern in dieser Dicke robust gegen Biegen oder Knicken.

Der obere Teil der 3D-gedruckten Einhausung besitzt viele Löcher, welche als Durchführung für diese Kabel gedacht sind. Die vorkonstruierte Bohrung besitzt einen Durchmesser von 1,8mm.

Figure 7: Handbohrer

Wenn es durch schnellen und qualitativ schlechten Druck vorkommen sollte, dass sich das Kabel nicht durch die Öffnungen der Einhausung schieben lässt, dann kann mit einem kleinen Handbohrer nachgearbeitet werden. Diese Handbohrer sind handgeführt und daher ungefährlich; üblicherweise werden sie mit einer guten Grundausstattung an verschiedenen Bohrern ausgeliefert.

Um die Glasfasern zu schneiden, sollte man keine Zange oder Schere verwenden, weil sie die Glasfaser quetschen. Es wird ein glatter Schnitt benötigt, aus dem das Licht heraustreten kann. Deshalb empfiehlt es sich, ein Teppichmesser zu benutzen.

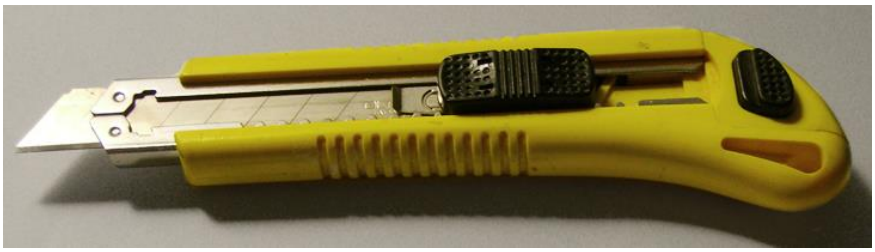


Figure 8: Cutter

Der CPX

In dieser Arbeit verwenden wir den Python-basierten Mikrocontroller "Circuit Python Express" der Firma "Adafruit Industries". Dieses Board stellt eine günstige Möglichkeit dar, einen Mikrocontroller zusammen mit vielen vorgefertigten und eingebauten Aktoren und Sensoren im Unterricht einzusetzen.

Der CPX hat sich im mehrfachen Unterrichtseinsatz als robust und zuverlässig erwiesen. Zudem bietet die Firma Adafruit auch sehr viel Information rund um den Mikrocontroller an. Neben zahlreichen Beispielprogrammen existiert auch eine gute Dokumentation.

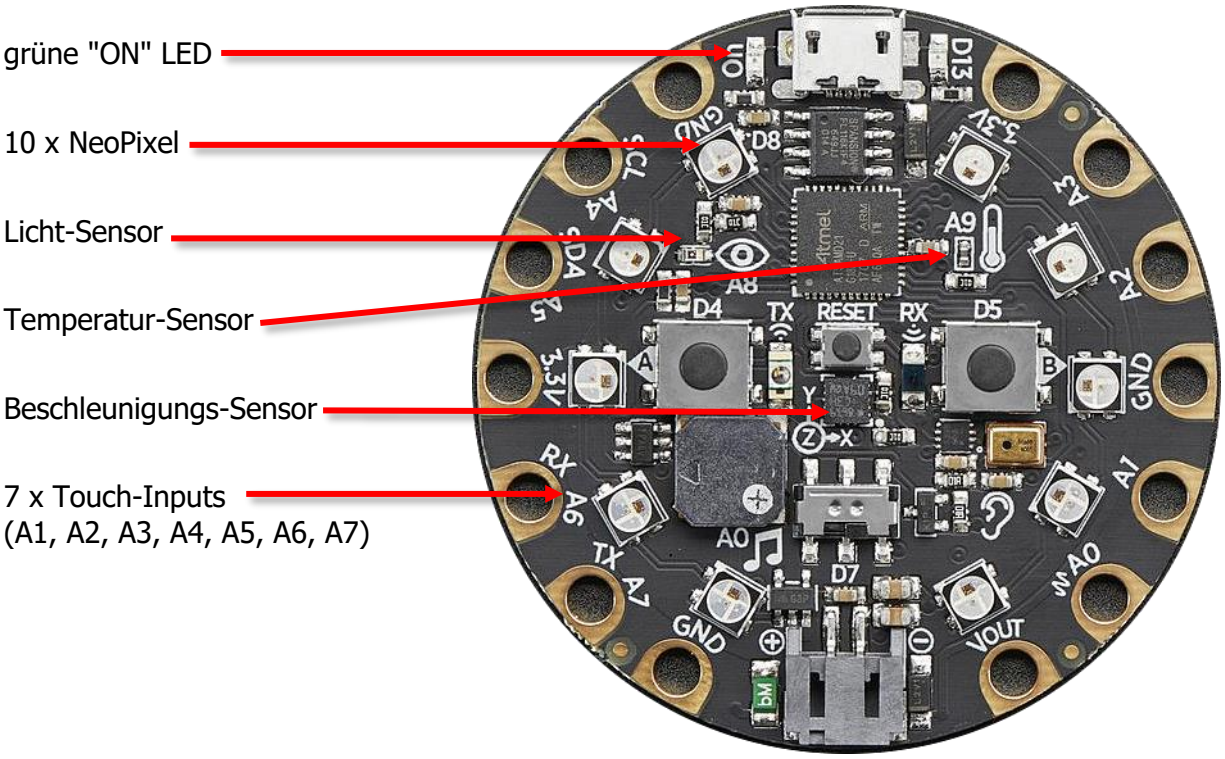


Figure 9: Einige Funktionen des CPX

Ein Tutorial zur Einführung in die Programmierung des CPX zusammen mit vielen Beispielprogrammen findet sich hier:

https://iludis.de/?page_id=291

Die Computer, mit denen die Schüler arbeiten, sollte folgende Software installiert haben:

- Autodesk Fusion 360 (or any other 3D-modeling-Software, e.g. Wings3D)
- CURA slicing software,
- An internet connection for downloading libraries
- **Mu Editor (https://codewith.mu/)**

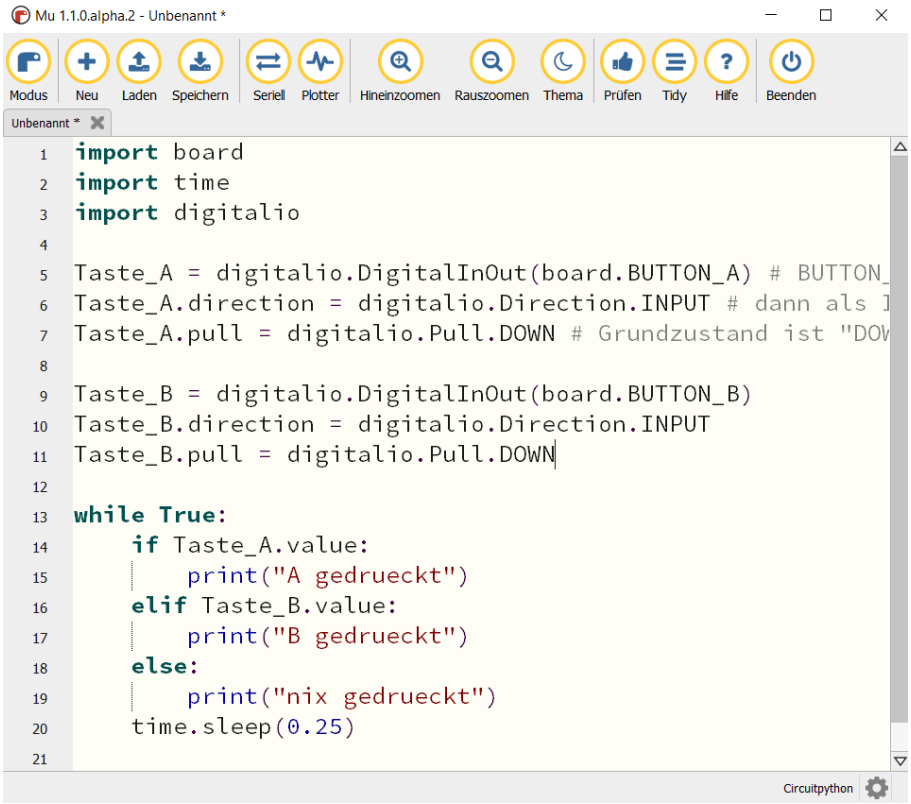


Figure 10: Screenshot Mu Editor

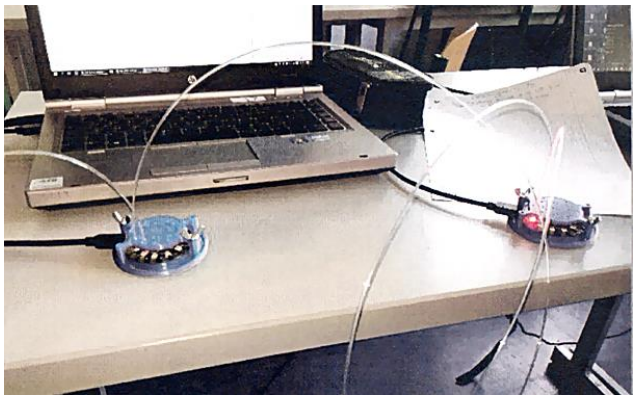


Figure 11: Foto des Aufbaus von Schülern

9. Beschreibung der Unterrichtseinheit: Alle Aktivitäten und Inhalte

Lesson 1 & 2 (90min): Einführung in IoT

Die Schüler lernen IoT durch Beispiele kennen: Staubsauger-Roboter mit App-Fernsteuerung, internet-gestützte Wetterstationen, Smart-Farming und nicht zu guter letzt Gesundheits-Apps. Die Schüler erhalten Informationen darüber, wie diese Geräter funktionieren, wie die Steuerung von Mikrocontroller-basierten Systemen funktioniert, welche mit Sensoren und Aktoren verbunden sind. Weiterhin ist die Art und Weise der Kommunikation der zahlreichen Geräte untereinander für die weitere Unterrichtseinheit relevant. Außerdem sollen Möglichkeiten und Gefahren der Technologie angesprochen werden – zusammen mit der Frage, wo IoT sinnvoll ist und wo nicht.

Lesson 2 & 3 (90min): Einführung in die Theorie von Kommunikations-Protokollen

Die Schüler der Klasse spielen ein Rollenspiel: Die Klasse schaut zu und entwickelt Ideen mit. Folgende Regeln gelten:

- Zwei Schüler "Sender" und "Empfänger" setzen sich zusammen nach vorne auf Stühle, die Rücken an Rücken stehen, damit sich beide nicht sehen können zwischen den beiden steht ein dritter, leerer Stuhl.
- Der Sender soll dem Empfänger zwei völlig unterschiedliche Worte senden, die rückwärts gesprochen eine völlig andere Bedeutung erhalten, wie zum Beipiel die Wortpaare: "NEBEL"/ "LEBEN" und "LAGER"/ "REGAL" | "SARG"/ "GRAS" und "EBER"/ "REBEN". (4 letters: "LIVE"/EVIL" and "STAR"/ "RATS" | 3 letters: "GOD"/ "DOG" and "RAW/WAR")
- Die Wortübertragung lässt – genauso wie gesprochene Sprache – nur eine Sequenz von einzelnen Zeichen nacheinander zu, es muss also alle Buchstaben einzel übertragen werden. Deshalb werden die Buchstaben der beiden Worte auf einzelne Zettel geschrieben.
- Diese Zettel können nur einer nach dem anderen vom Sender auf den Empfänger übergeben werden, indem der Sender einen Zettel auf den Stuhl legt und der Empfänger den Zettel dort abholt – ohne dass sich die beiden Parteien sehen können.
- Die einzige direkte Kommunikation, die zwischen beiden Parteien erlaubt ist, ist ein einzelner Ton (z.B. "piep"), den jeder abgeben darf.
- WEnn eine Kommunikation schiefläuft, dann wird die Nachrichtenübertragung unterbrochen, eine neue Regel wird aufgestellt und man beginnt erneut von vorne.

Sinn ist es, dass die Schüler erkennen, dass Kommunikation über gemeinsam verabredete Regeln erfolgen muss, nämlich über ein Protokoll:

Die Schüler müssen ein Startsignal vereinbaren. Es muss nach jedem Buchstaben eine zeitliche Pause erfolgen, damit die Zeichen in der richtigen Reihenfolge ankommen; dies kann durch Taktung oder durch "piepsen" vereinbart werden. Weiterhin muss die Reihenfolge verabredet werden, in der die Buchstaben ausgetauscht werden. Denn sonst erhalten die Wörter eine andere, nicht gewollte Bedeutung. Und zuletzt muss die Kommunikation beendet werden.

Lesson 4 & 5 & 6 (120min): Principles of serial communication

Prinzipien des Multiplexings und Demultiplexings: Datenbits werden eins nach dem anderen, beginnend vom MSB ('most significant bit') hin zum LSB ('least significant bit') in einer festgelegten Reihenfolge über ein einzelnes Datenkabel übertragen.

Dabei wird unterschieden zwischen synchroner und asynchroner Datenübertragung: Im einfacheren synchronen Fall gibt der Empfänger – der als Master funktioniert – die gemeinsame Taktfrequenz vor, mit der er einzelne Bitübertragung beim Sender (Slave) anstößt. Nach einer vorher zu verabredenden Anzahl an übertragenen Bits ist die Datenübertragung beendet.

Im asynchronen Fall muss vorher bei beiden an der Kommunikation beteiligten Geräten eine Taktzeit verabredet werden, wobei diese Zeiten nur ganz wenig voneinander abweichen dürfen.

Lesson 7 (45min):

Geschichte der Datenübertragung: Émile Baudot

Anhand des 5-Bit-Baudot-Codes werden die Grundlagen der Datenübertragung am praktischen Beispiel erarbeitet. Wenn man lediglich Großbuchstaben übertragen will, so benötigt man 26 verschiedene Buchstabenzeichen und eventuell 2-3 Satzzeichen, wie z.B. das Leerzeichen oder das Fragezeichen. Um diese weniger als 32 verschiedenen Symbole mit Bits zu codieren, benötigt man also 5 Bit; eine mögliche Codierung wäre zum Beispiel:

Symbol	A	B	C	D	E	F	G	H	I	J	K
Bitcode	00001	00010	00011	00101	00110	00111	01000	01001	01010	01011	01100

Symbol	L	M	N	O	P	Q	R	S	T	U	V
Bitcode	01101	01110	01111	10000	10001	10010	10011	10100	10101	10110	10111

Symbol	W	X	Y	Z	LEER	START	STOP	.	
Bitcode	11000	11001	11010	11011	11100	11101	11110	11111	00000

Die Schüler sollen sich ein System überlegen, mit dem die bitweise Codierung von Buchstaben erfolgen kann – idealerweise gelangen sie zu ähnlichen Ideen wie in obiger Tabelle.

Da Bits in einer bestimmten Geschwindigkeit übertragen werden, spricht man von einer sogenannten Baudrate, die nach dem französischen Ingenieur Emile Baudot benannt wurde. Hier kann auf die Biografie von Baudot eingegangen werden.

Lesson 8 & 9 (90min): Wiederholung Totalreflexion

Die Totalreflexion

Wir nutzen die Simulationen „Light Refraction“

https://javalab.org/en/light_refraction_en/

und „Total Internal Reflection“

https://javalab.org/en/total_internal_reflection_en/

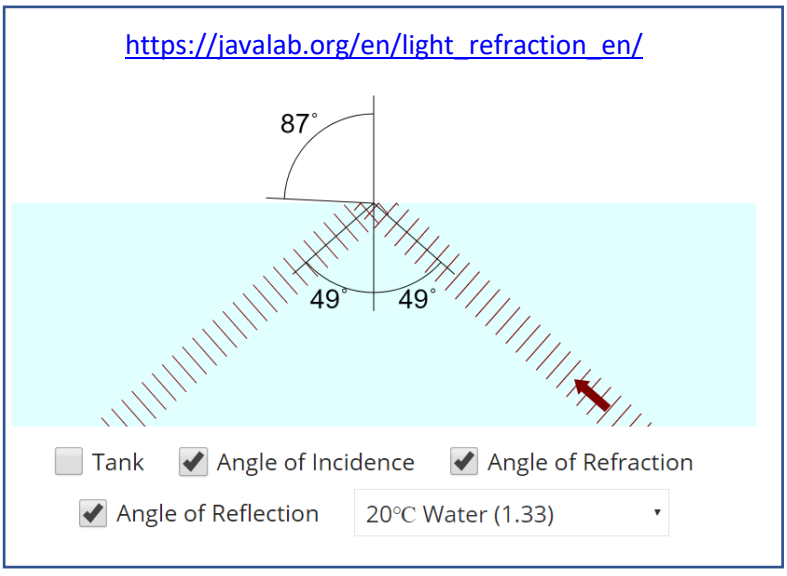
Für den Übergang von Wasser in Luft gilt

„Der Lichtstrahl wird beim Übergang vom optisch dichteren Medium (Wasser) ins optisch weniger dichte Medium (Luft) vom Lot weggebrochen“

Überschreitet der Einfallswinkel des Lichtstrahls im Wasser einen sogenannten „**kritischen Winkel**“, dann müsste der Brechungswinkel im Medium Luft größer als 90° sein – und das ist unmöglich!
Deshalb bleibt dem Lichtstrahl nichts anderes übrig, als im Wasser zu bleiben.

1) kritische Winkel beim Übergang von verschiedenen Medien in Luft

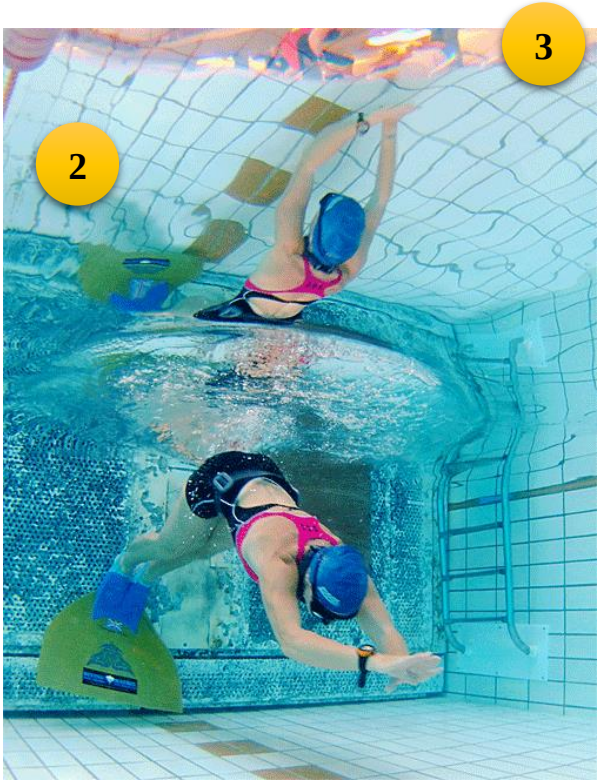
https://javalab.org/en/light_refraction_en/



Bestimme die kritischen Winkel – also diejenigen Einfalls-Winkel, bei denen der Lichtstrahl es gerade NICHT mehr aus dem Medium schafft:

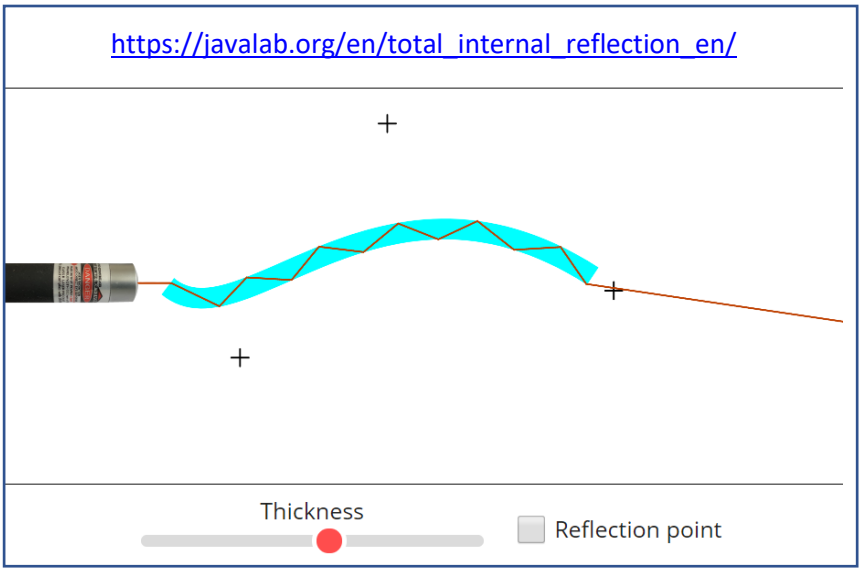
Medium	Kritischer Winkel, oberhalb erfolgt Totalreflexion)
Wasser	49°
Diamant	
Saphir	
Glas	

2) Interpretiere die folgenden beiden Bilder:



Interpretiere die markierten Bereiche in den beiden Bildern mit eigenen Worten. Wie kommt es zu diesen Spiegelungen? Und was genau sieht man in Bereich 3?

3) Anwendungen der Totalreflexion



Auf der linken Seite befindet sich ein Laser, der seinen Lichtstrahl in eine sogenannte Glasfaser einspeist. Glasfasern werden verwendet, um Informationen mit Licht zu transportieren. Dabei wird auf der einen Seite der Glasfaser Computer-Signale mittels Laserstrahl eingestrahlt und auf der anderen Seite mittels eines Lichtsensors die Lichtsignale empfangen und an den Zielcomputer weitergegeben.

Erläutere das Funktionsprinzip der Glasfaser, beantworte dabei auch die folgenden Fragen:

- Wieso bleibt der Lichtstrahl in der Faser?
- Gibt es Situationen, in denen der Lichtstrahl zu früh aus der Faser austritt?
- Wie hängt die Lichtleitung innerhalb der Faser von der Dicke der Faser ab?

Lesson 10 & 11 (90min):

Einführung in die Programmierung mit Python: der CPX wird mit dem Rechner über USB-Kabel verbunden und mit der Programmierumgebung "mu-Editor" programmiert. Hier einige Skripten, um sich in den CPX einzuarbeiten. Quelle: https://iludis.de/?page_id=291

Skript 1, "Blink", LED an- und ausschalten:

```
import board            # Objekt board: Befehle für den Board-Zugriff laden
import digitalio        # Objekt „Pin-Daten" des Boards dazuladen
import time             # Objekt time: Zeit-Funktionen dazuladen

meinPin = digitalio.DigitalInOut(board.D13)
meinPin.direction = digitalio.Direction.OUTPUT

while True:
      meinPin.value = True    # Der Pin wird angeschaltet.
      time.sleep(1)           # Es wird eine Sekunde gewartet.
      meinPin.value = False   # Der Pin wird angeschaltet.
      time.sleep(1)           # Es wird eine Sekunde gewartet.
```

Skript 2, "Piep", Ton abspielen:

```
import time
from adafruit_circuitplayground.express import cpx

for i in range (1, 5, 1):
    cpx.start_tone(262)
    time.sleep(0.2)
    cpx.stop_tone()
    time.sleep(0.2)
    print(i)
```

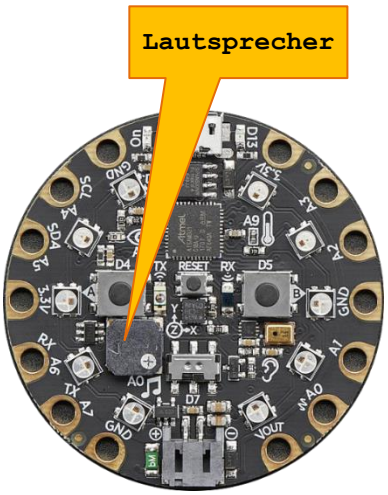


Figure 12: Ort des Lautsprechers

Skript 3, "Touch", Ton abspielen, wenn Touch gedrückt:

```
import board
import time
import touchio
from adafruit_circuitplayground.express import cpx

B_A1 = touchio.TouchIn(board.A1)
B_A2 = touchio.TouchIn(board.A2)

while True:
    if B_A1.value == True:
        cpx.start_tone(262)
    else:
        cpx.stop_tone()

    if B_A2.value == True:
        cpx.red_led = True
    else:
        cpx.red_led = False

    print(B_A1.value, B_A2.value)
    time.sleep(0.1)
```

Skript 4, Neopixel anschalten

```
import board, time, neopixel
NeopixelListe = neopixel.NeoPixel(board.NEOPIXEL, 10)

for i in range (10):
    NeopixelListe[i] = (0,64,0)
    print(NeopixelListe[i])
    time.sleep(0.1)
print(NeopixelListe)
```

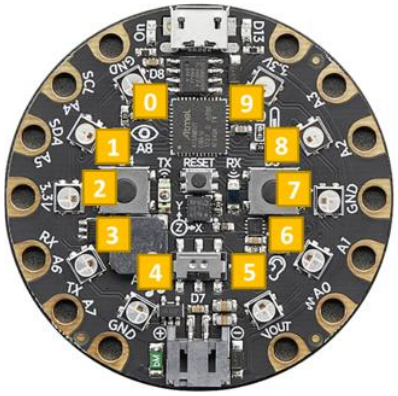


Figure 13: Die Nummerierung der Neopixel

Skript 5, Helligkeitssensor auslesen

```
import board
import time
import analogio

licht = analogio.AnalogIn(board.LIGHT)

while True:
    print((licht.value,))
    time.sleep(0.1)
```

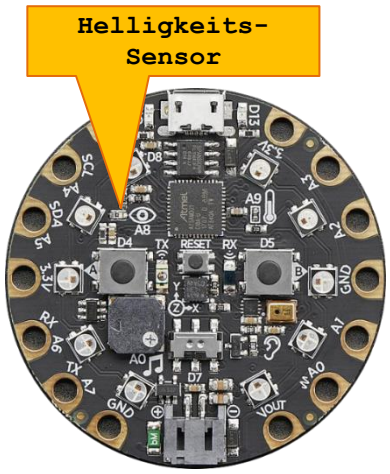


Figure 14: Ort des Helligkeitssensors

Lesson 12 & 13 (90 min): Beispielskript der synchronen Kommunikation

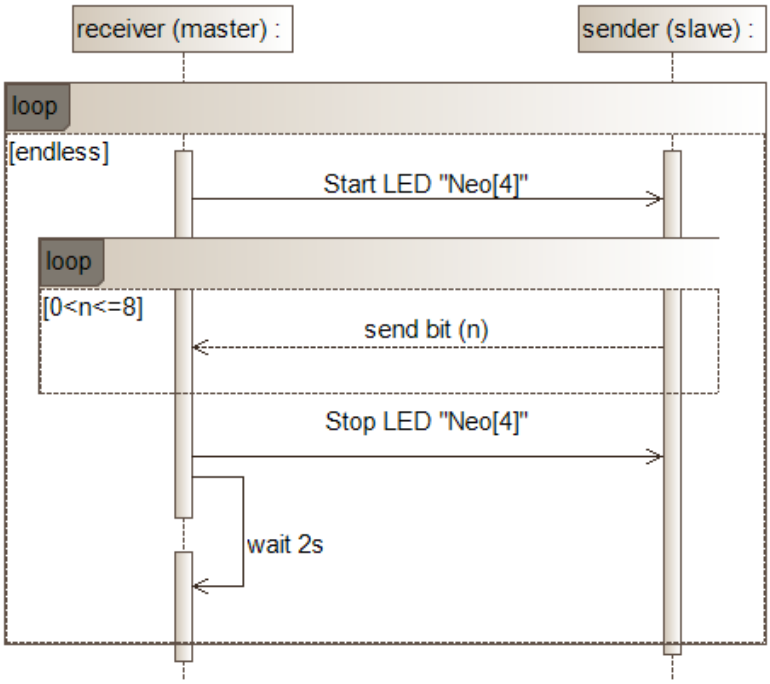


Figure 15: UML-Sequenzdiagramm der synchronen Kommunikation

Empfänger Sourcecode

```
import board
import time
import neopixel
import analogio

light = analogio.AnalogIn(board.LIGHT)
Neo=neopixel.NeoPixel(board.NEOPIXEL,10)
pulseDuration = 0.05

while True:
    listenBits = [2, 2, 2, 2, 2, 2, 2, 2]
    Neo[4] = (0, 255, 0)
    for i in range(len(listenBits)):
        time.sleep(pulseDuration)
        if light.value < 40000:
            listenBits [i] = 0
        if light.value > 40000:
            listenBits [i] = 1
    Neo[4] = (0, 0, 0)
    print(listenBits)
    time.sleep(2)
```

Sender Sourcecode

```
import board
import time
import neopixel
import analogio

light = analogio.AnalogIn(board.LIGHT)
Neo=neopixel.NeoPixel(board.NEOPIXEL,10)
pulseDuration = 0.05

while True:
    sendBits = [1, 0, 0, 1, 1, 0, 1, 0]
    if light.value > 40000:
        for i in range(len(sendBits)):
            if sendBits[i] == 1:
                Neo[6] = (255, 0, 0)
            if sendBits[i] == 0:
                Neo[6] = (0, 0, 0)
            time.sleep(pulseDuration)
```

Lessons 14 & 15 (90 min): Beispielskript der asynchronen Kommunikation

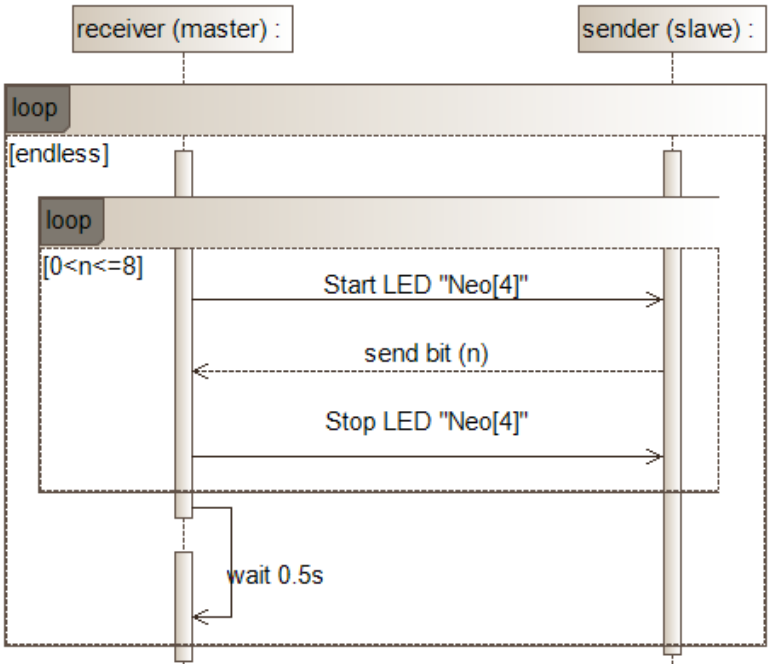


Figure 16: UML-Sequenzdiagramm der asynchronen Kommunikation

```
import board
import time
import neopixel
import analogio

light = analogio.AnalogIn(board.LIGHT)
Neo=neopixel.NeoPixel(board.NEOPIXEL, 10)
pulseDuration = 0.02



while True:
    listenBits = [2, 2, 2, 2, 2, 2, 2, 2]
    for i in range(len(listenBits)):
        Neo[4] = (0, 255, 0)
        time.sleep(pulseDuration)
        if light.value < 40000:
            listenBits[i] = 0
        if light.value > 40000:
            listenBits[i] = 1
        time.sleep(pulseDuration)
        Neo[4] = (0, 0, 0)
        time.sleep(pulseDuration)
    print(listenBits)
    time.sleep(0.5)
```

```
import board

import neopixel
import analogio

light = analogio.AnalogIn(board.LIGHT)
Neo=neopixel.NeoPixel(board.NEOPIXEL, 10)
sendBits = [1, 0, 0, 1, 1, 0, 1, 0]
i = 0
transmit = True

while True:
    print(i)
    if light.value > 40000 and transmit:
        if sendBits[i] == 1:
            Neo[6] = (255, 0, 0)
        if sendBits[i] == 0:
            Neo[6] = (0, 0, 0)
        transmit = False
    if light.value < 40000 and not transmit:
        i = i+1
        i = I % 8
        transmit = True
```

10. Feedback	Am Ende der Unterrichtseinheit sollten die Schüler ein grundlegendes tieferes Verständnis entwickelt haben, wie serielle Kommunikation funktioniert und welche informatischen Prinzipien dazu notwendig sind. Während der Lektionen werden wichtige Aspekte der Elektronik, Optik und dem Aufbau von Protokollen angeschnitten.
11. Assessment & Evaluation	Die Schüler führen ein Praktikums-Journal, welches vom Lehrer kontrolliert werden kann. Die Schüler können ebenfalls ihre Arbeitsergebnisse vor der Gruppe vorstellen. Zusätzlich sollte eine Klassenarbeit über den Stoff am Ende der Unterrichtseinheit geschrieben werden.