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

‘We are the makers – Scénario d’apprentissage IoT : communication en fibre optique avec Adafruit CPX Express

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

Le document suivant a été développé et testé dans un environnement scolaire avec environ 16 élèves âgés de 13 à 14 ans en année scolaire 2019/2020. Il reflète l'expérience d'étudiants talentueux et curieux qui ont créé eux-mêmes leurs scripts de programmation après une unité d'enseignement de la technologie des réseaux. Ce document est censé être une recommandation comme point de départ.

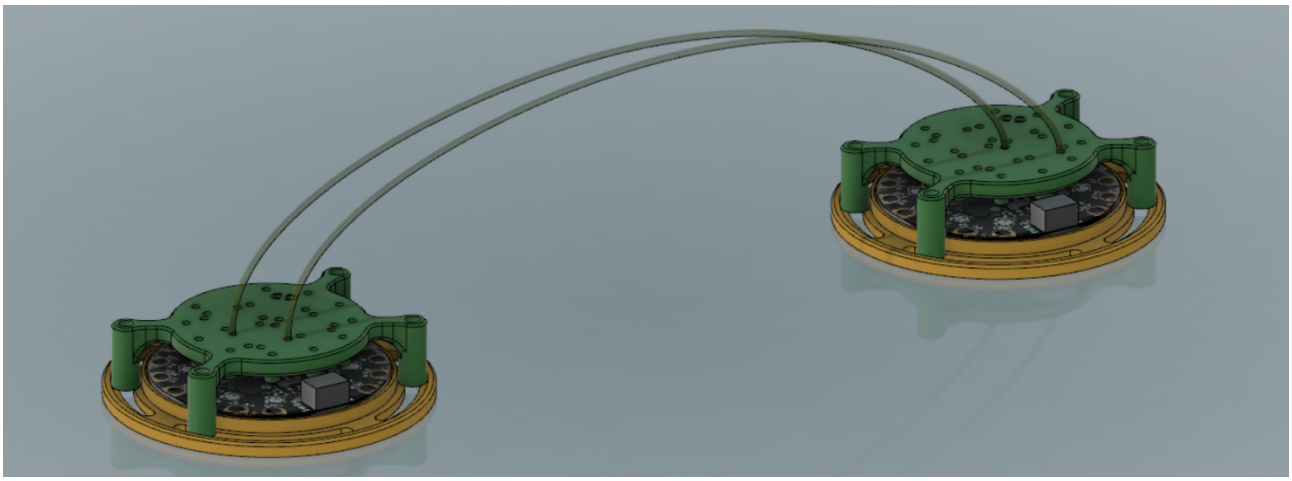
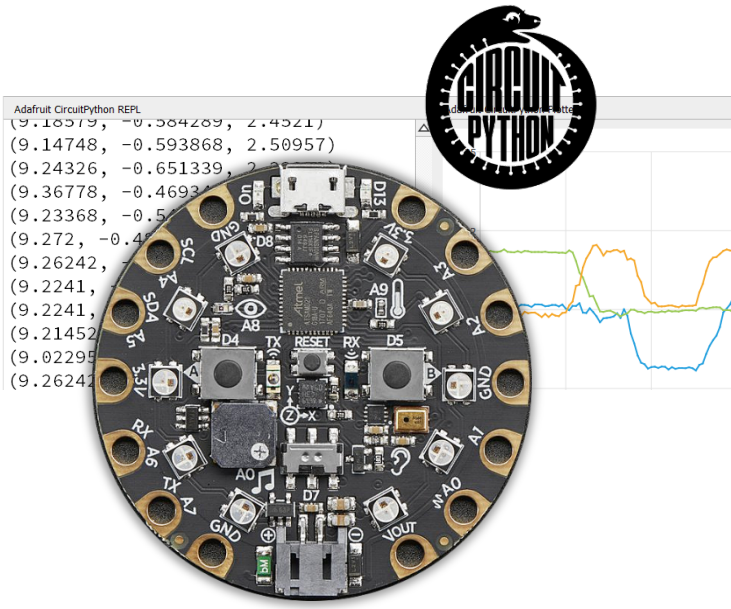


Figure 1: Prototype d'une configuration de communication en fibre optique entre deux microcontrôleurs



L'IoT signifie la mise en réseau d'appareils contrôlés par ordinateur. La communication réseau signifie souvent que les appareils échangent leurs informations sans fil, c'est-à-dire par radio. En fait, des supports classiques tels que la connexion par fibre optique sont également inclus.
L'avantage didactique de l'utilisation de la fibre optique pour l'introduction aux technologies de communication est la visibilité de l'échange d'informations: vous pouvez voir les impulsions lumineuses avec lesquelles les bits d'information sont échangés. Un principe physique simple, à savoir la réflexion totale, suffit pour comprendre la communication moderne par fibre optique.

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

1. Title du Scenario	**IoT Device Network Communication: Glass Fiber Programming**
2. Groupe cible	▪ 14 - 16 ans
3. Durée	• Au minimum 7 semaines avec 2-3 leçons par semaine
4. Besoins couverts par l'exercice	▪ Capteurs et actionneurs dans l'échange d'informations entre appareils numériques ▪ Principe de la communication réseau basée sur des protocoles et des paquets ▪ Application de la réflexion totale pour le transport des impulsions lumineuses. ▪ Programmation de microcontrôleurs basés sur python en petits groupes de deux étudiants chacun
5. Résultats attendus	▪ Comment fonctionne un système IoT? ▪ Comment structurer et mettre en œuvre la communication réseau? ▪ Pourquoi avez-vous besoin d'un protocole de communication? ▪ Comment fonctionne la communication numérique par paquets?
6. Méthodologies	▪ Dans ce scénario, les élèves construiront, construiront et programmeront eux-mêmes une communication série entre deux microcontrôleurs. Les étudiants utiliseront également le moniteur série et le traceur série pour visualiser et tracer des données.
7. Environnement	▪ Un ensemble de câbles à fibres optiques pour connecter les microcontrôleurs ▪ Un ensemble de microcontrôleurs Adafruit CPX, ▪ Ces contrôleurs CPX sont programmés avec Circuit python ▪ Chaque étudiant reçoit un ordinateur portable avec un éditeur Mu pré-installé pour son microcontrôleur CPX ▪ Chaque CPX est intégré dans un boîtier imprimé en 3D, ce qui garantit l'alignement précis des fibres de verre. ▪ Chaque étudiant rédige un journal de son travail de projet

8. Outils, matériaux et ressources

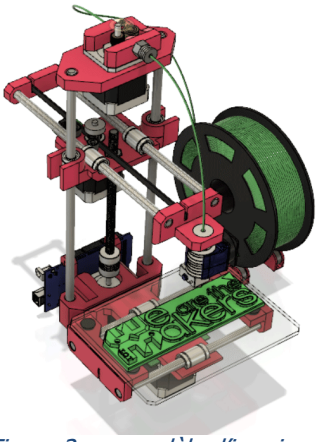


Figure 2: un modèle d'imprimante 3D

Imprimante 3D

Environ 2-3 imprimantes 3D sont nécessaires car les étudiants imprimeront leur Boîtiers CPX. Bien entendu, les élèves peuvent construire des pièces de machine par elles-mêmes

Éléments imprimés en 3D :

Comme point de départ, toutes les pièces nécessaires sont fournies au format .stl et en tant que fichiers Autodesk Fusion 360 :

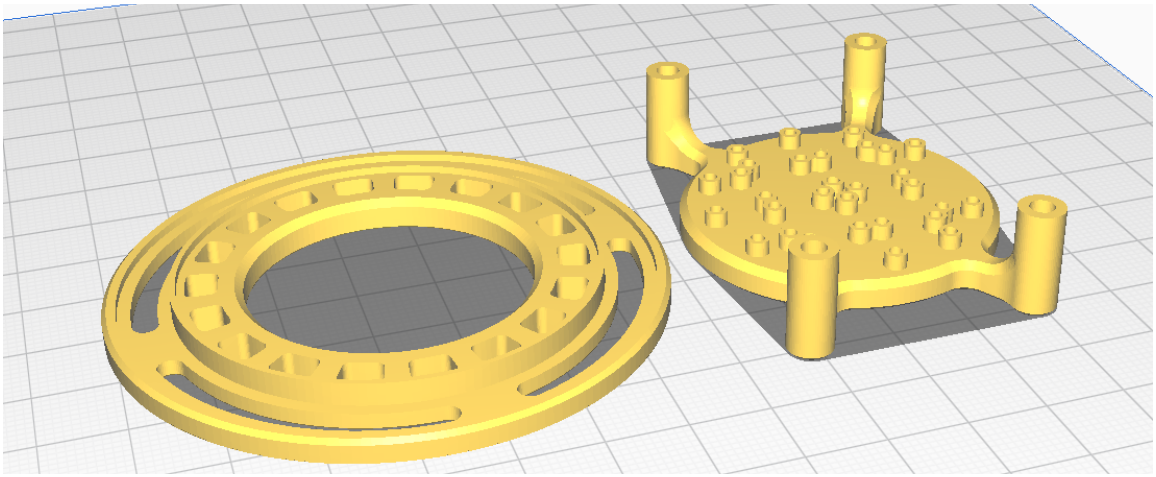
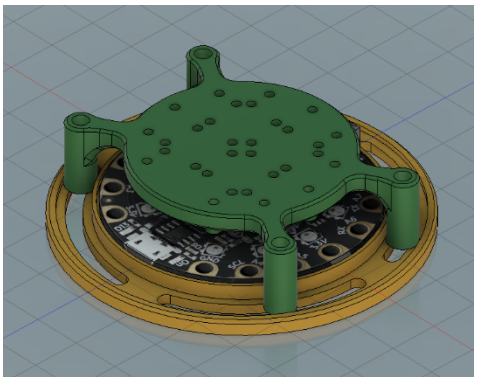


Figure 3: Le temps d'impression pour ces deux fichiers STL est d'environ 1h

Les deux parties sont dimensionnées de telle manière qu'un CPX avec un jeu d'environ 0,5 mm puisse être inséré dans la partie inférieure. Les parties supérieure et inférieure sont reliées par des vis M3: la longueur recommandée de la vis est de 25 mm. Les pièces peuvent être verrouillées avec des écrous papillon, de sorte que le microcontrôleur puisse être rapidement retiré et retiré.

Figure 4: Vis M3 et écrou papillon

Figure 5: Preview of Fusion-file



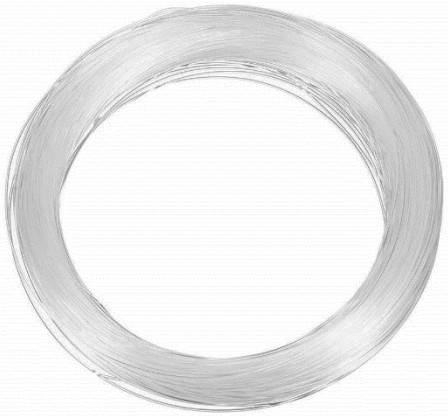


Figure 6: 100 mètres de fibre de verre

Le boîtier est conçu pour brancher facilement les câbles à fibres optiques et les positionner de manière optimale au-dessus des LED et du capteur de luminosité.
Un câble PMMA de 1,5 mm est utilisé comme fibre de verre. Vous obtenez ces fibres en rouleaux d'environ 100 mètres de long pour environ 20 euros. Chaque groupe d'étudiants souhaitant connecter deux CPX a besoin de 2 mètres sur 1 de câble. Un câble comme ligne de transmission et un câble comme ligne de réception. Avec une de ces bobines, plusieurs classes peuvent être alimentées en fibre optique à moindre coût.
Le diamètre de 1,5 mm garantit que les étudiants ne peuvent pas se blesser si facilement. De plus, les fibres de cette épaisseur sont résistantes à la flexion.

La partie supérieure du boîtier imprimé en 3D comporte de nombreux trous, destinés à guider ces câbles. L'alésage pré-construit a un diamètre de 1,8 mm.

Figure 7: Perceuse à main

Si, en raison d'une impression rapide et de mauvaise qualité, le câble ne peut pas être poussé à travers les ouvertures du boîtier, il peut être retravaillé avec une petite perceuse à main. Ces perceuses à sont guidées à la main et donc inoffensives; ils sont généralement livrés avec un bon équipement de base pour divers exercices.

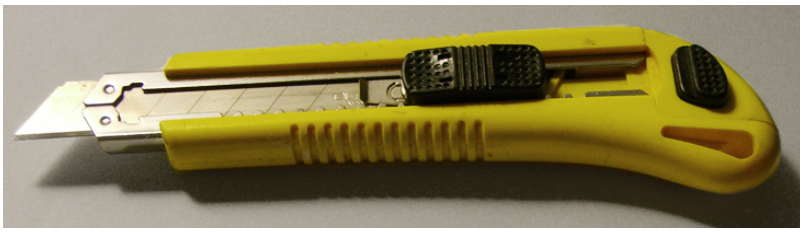


Figure 8: Cutter

Pour couper les fibres de verre, vous ne devez pas utiliser de pinces ou de ciseaux car ils pressent la fibre de verre. Une coupe lisse est nécessaire à partir de laquelle la lumière peut sortir. Par conséquent, il est recommandé d'utiliser un couteau à moquette.

Le CPX Adafruit

Pour ce projet, nous utilisons le microcontrôleur basé sur Python "Circuit Python Express" de la société "Adafruit Industries". Cette carte est un moyen pratique d'utiliser un microcontrôleur avec de nombreux actionneurs et capteurs préfabriqués et intégrés dans la salle de classe.

Le CPX s'est avéré robuste et fiable dans de multiples applications d'enseignement. De plus, Adafruit propose également de nombreuses informations sur le microcontrôleur. En plus de nombreux exemples de programmes, il existe également une bonne documentation.

LED verte "ON"

10 x NeoPixel

Capteur de lumière

Capteur de température

Accéléromètre

7 x entrées tactiles
(A1, A2, A3, A4, A5, A6, A7)

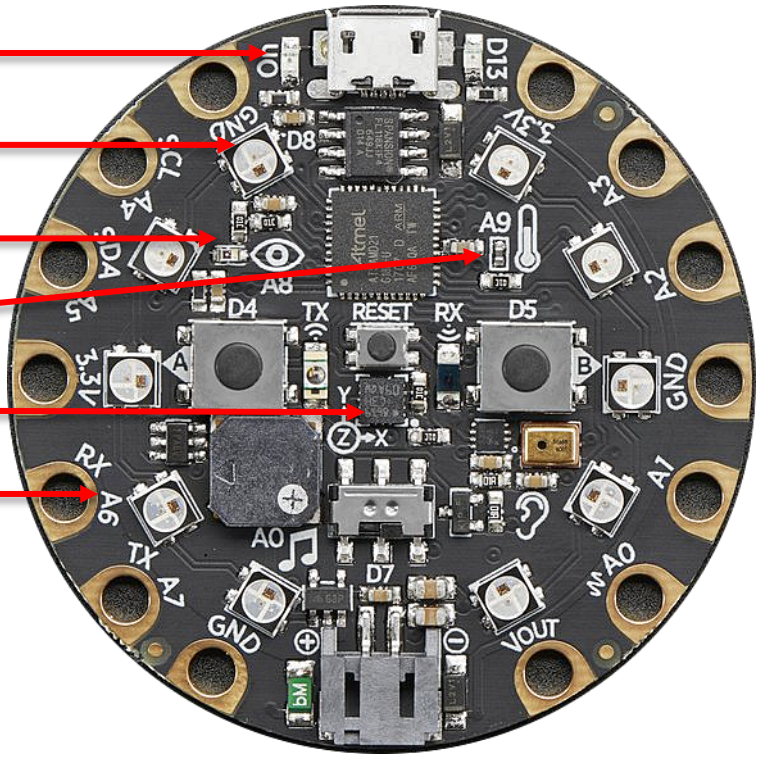
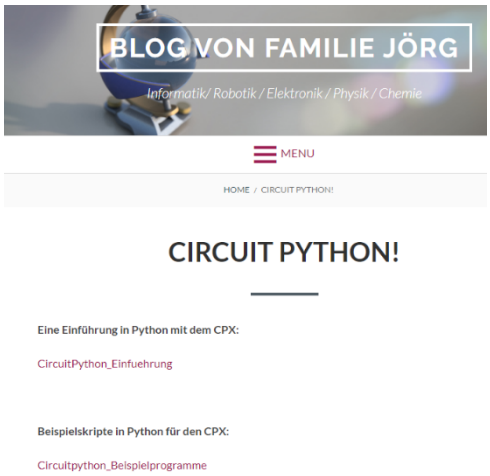


Figure 9: some features of the CPX

Un tutoriel sur la façon d'introduire la programmation du CPX avec de nombreux exemples de programmes peut être trouvé ici :
https://iludis.de/?page_id=291



Équipement necessaire

Les ordinateurs avec lesquels les étudiants travaillent doivent avoir les logiciels suivants installés:

- Autodesk Fusion 360 (ou tout autre logiciel de modélisation 3D, par exemple Wings3D)
- Logiciel de tranchage CURA,
- Une connexion Internet pour télécharger des bibliothèques
- Editeur Mu (https://codewith.mu/)

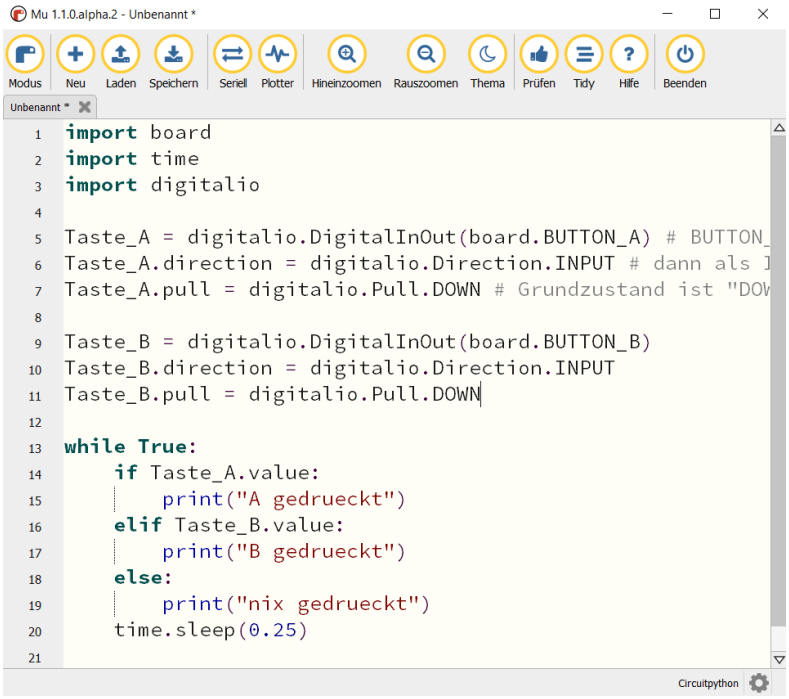


Figure 10: Ecran de l'éditeur Mu

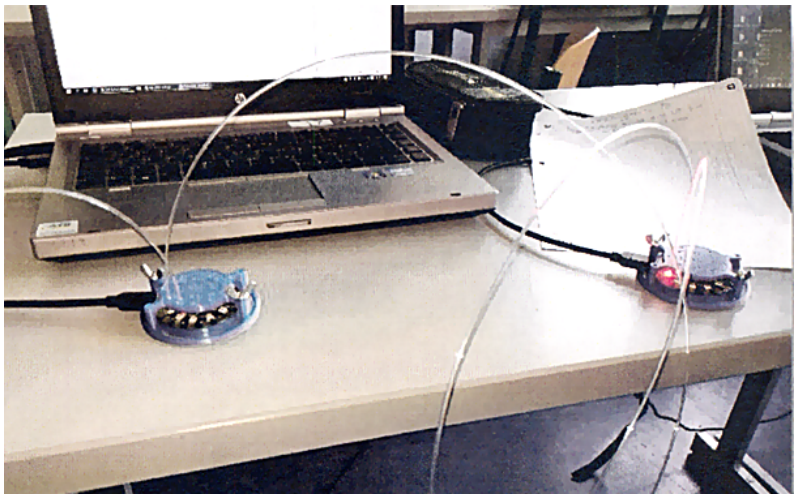


Figure 11: Photographie de l'installation d'un étudiant

9. Plan de leçon: description étape par étape de l'activité / du contenu

Leçons 1 & 2 (90min): Introduction à l'IoT

Les étudiants découvriront l'IoT par des exemples: des robots aspirateurs avec télécommandes d'applications, des stations météorologiques basées sur Internet, une agriculture intelligente et, enfin, des applications de santé. Les étudiants doivent examiner le fonctionnement de ces appareils et les composants nécessaires : un système basé sur un microcontrôleur contrôle et coordonne les capteurs et les acteurs connectés. De plus, il communique et se coordonne avec d'autres systèmes de type similaire souvent via des réseaux de communication sans fil. Pièces nécessaires: capteurs, acteurs, appareils de communication. Les possibilités et les menaces doivent être discutées ainsi que les limites: où l'IdO a-t-il un sens et où pas ?

Leçons 2 & 3 (90min): Introduction à la théorie des protocoles de communication

Les élèves de la classe jouent à un jeu de rôle : la classe regarde et développe des idées. Les règles suivantes s'appliquent :

- Deux élèves «émetteur» et «récepteur» s'assoient ensemble sur des chaises, qui se tiennent dos à dos de façon à ce que les deux ne puissent pas se voir. Entre les deux se trouve une troisième chaise vide.
- L'émetteur est censé envoyer au récepteur deux mots complètement différents, qui sont prononcés à l'envers avec un sens complètement différent, comme les paires de mots:
- (4 lettres: «ADOS» / SODA »et« BONS »/« SNOB »| 3 lettres:« RIT »/« TIR »et« LES / SEL »)(ces mots sont dits anacycliques NdT)
- Comme la langue parlée, la transmission de mots ne permet qu'une seule séquence de caractères individuels à la suite, de sorte que toutes les lettres doivent être transmises individuellement. Par conséquent, les lettres des deux mots sont écrites sur des notes individuelles.
- Ces notes ne peuvent être transférées une par une de l'expéditeur au destinataire qu'en plaçant une note sur le fauteuil et le destinataire y ramassant la note - sans que les deux parties puissent se voir.
- La seule communication directe autorisée entre les deux parties est une tonalité unique (par exemple "bip") que tout le monde est autorisé à donner.
- Si une communication se passe mal, la transmission du message est interrompue, une nouvelle règle est établie et les étudiants recommencent.

Les étudiants doivent se rendent compte que la communication doit se faire selon des règles convenues d'un commun accord, à savoir un protocole :

Les étudiants doivent s'entendre sur un signal de départ. Il doit y avoir une pause dans le temps après chaque lettre pour que les caractères arrivent dans le bon ordre; cela peut être convenu par pointage ou par "bips". En outre, l'ordre dans lequel les lettres sont échangées doit être convenu. Sinon, les mots auront une signification différente et involontaire. Et enfin, la communication doit être arrêtée.

Leçons 4 & 5 & 6 (120min) : Principes de la communication série

Principes de multiplexage et de démultiplexage: Les bits de données sont transmis un par un, à partir du MSB («bit le plus significatif») vers le LSB («bit le moins significatif») dans un ordre spécifié via un seul câble de données.

Une distinction est faite entre la transmission de données synchrone et asynchrone: dans le cas synchrone plus simple, le récepteur - qui fonctionne en tant que maître - spécifie la fréquence d'horloge commune avec laquelle il déclenche la transmission de bits individuels au niveau de l'émetteur (esclave). Après un nombre prédéterminé de bits transmis, le transfert de données est terminé.

Dans le cas asynchrone, un temps de cycle doit être convenu au préalable pour les deux appareils impliqués dans la communication, bien que ces temps puissent différer très peu l'un de l'autre..

Figure 12: Emile Baudot

Leçon 7 (45min):

Historique de la transmission de données: Émile Baudot

En utilisant le code Baudot à 5 bits, les bases de la transmission de données sont développées à l'aide de l'exemple pratique. Si vous souhaitez uniquement transférer des lettres majuscules, vous avez besoin de 26 lettres différentes et éventuellement de 2-3 signes de ponctuation, tels que l'espace ou le point d'interrogation. Pour encoder jusqu'à 32 symboles différents avec des bits, vous avez besoin de 5 bits ; le codage possible serait, par exemple :

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

Les élèves devraient réfléchir à un système qui peut être utilisé pour encoder des lettres au niveau du bit - idéalement, ils proposeront des idées similaires à celles du tableau ci-dessus.

Étant donné que les bits sont transmis à une certaine vitesse, on l'appelle le débit en bauds, qui a été nommé d'après l'ingénieur français Emile Baudot. Ici vous pouvez présenter la biographie de Baudot.

Leçons 8 & 9 (90min) : Répétition Réflexion totale

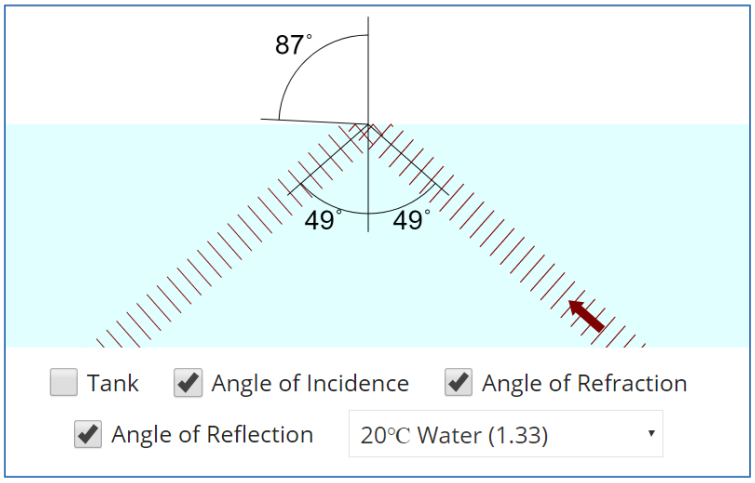
Réflexion totale

Nous utilisons les simulations «Réfraction de la lumière» : https://javalab.org/en/light_refraction_en/
Et «Réflexion interne totale» : https://javalab.org/en/total_internal_reflection_en/

Lors de la transition de l'eau à l'air,
"Le faisceau lumineux est réfracté à la transition du milieu optiquement plus dense (eau) au milieu optiquement moins dense (air)"

Si l'angle d'incidence du faisceau lumineux dans l'eau dépasse un soi-disant "**angle critique**", alors l'angle de réfraction dans l'air du milieu devrait être supérieur à 90 ° - et c'est impossible ! Par conséquent, le faisceau de lumière n'a d'autre choix que de rester dans l'eau.

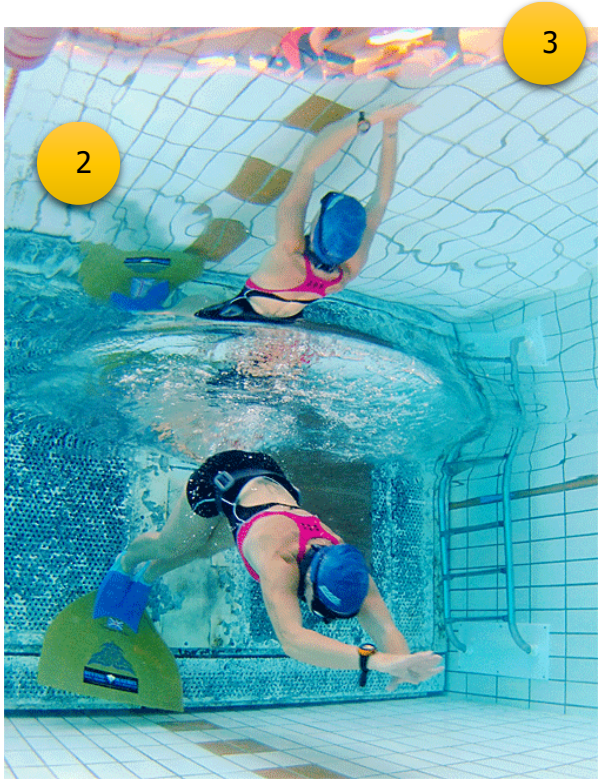
1) angles critiques lors du passage d'un média différent à l'air
https://javalab.org/en/light_refraction_en/



Déterminer les angles critiques - c'est-à-dire les angles d'incursion auxquels le faisceau lumineux ne sort PAS du milieu :

Milieu	Angle critique, au-delà duquel la réflexion est totale
Eau	49°
Diamant	
Saphire	
Verre	

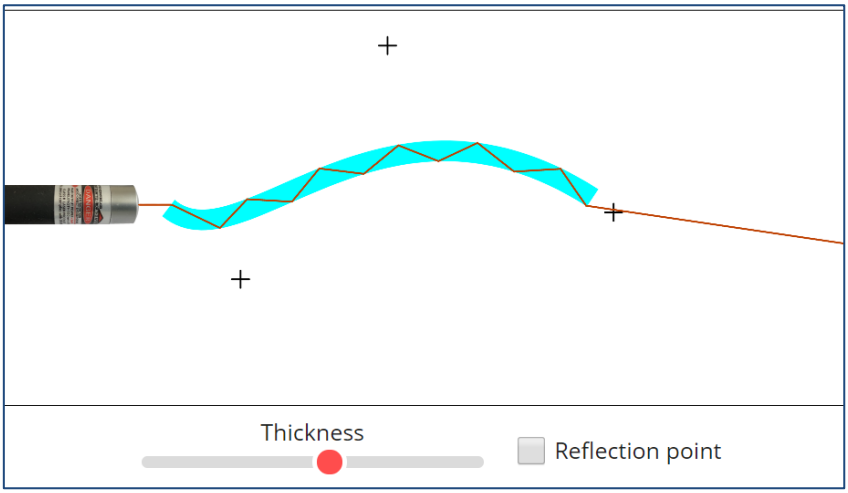
2) Interprétez les deux images suivantes :



Interprétez les zones indiquées dans les deux images. Comment naissent ces réflexions ? Et que voyez-vous exactement dans la zone 3?

3) Applications de la réflexion totale

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



Sur le côté gauche se trouve un laser qui envoie son faisceau de lumière dans une fibre de verre. Les fibres optiques sont utilisées pour transporter des informations avec la lumière. D'un côté, les signaux d'ordinateur à fibre optique sont irradiés au moyen d'un faisceau laser et, d'autre part, au moyen d'un capteur de lumière, les signaux lumineux sont reçus et transmis à l'ordinateur cible. Expliquer le principe de fonctionnement de la fibre de verre, en répondant aux questions suivantes:

1. 1. Pourquoi le faisceau lumineux reste-t-il dans la fibre ?
2. 2. Y a-t-il des situations où le faisceau lumineux s'échappe trop tôt de la fibre ?
3. 3. Comment la ligne lumineuse dans la fibre dépend-elle de l'épaisseur de la fibre ?

Leçons 10 & 11 (90min), Introduction à la programmation avec Python :

Le CPX est connecté à l'ordinateur via un câble USB et programmé avec l'environnement de programmation "mu-Editor". Voici quelques scripts pour vous habituer au CPX :
https://iludis.de/?page_id=291

Script 1, "Blink", Interrupteur LED marche et arrêt :

```
import board
import digitalio
import time

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

while True:
      meinPin.value = True
      time.sleep(1)
      meinPin.value = False
      time.sleep(1)
```

Script 2, "beep", jouer un son:

```
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)
```

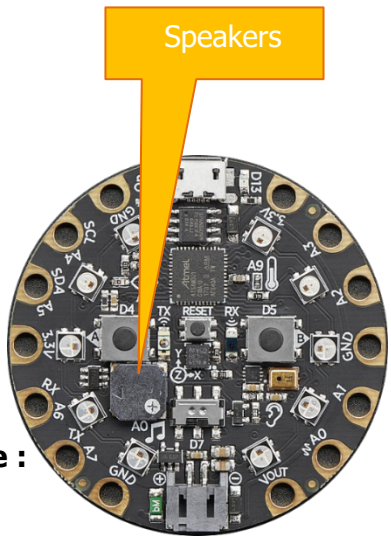


Figure 13: Location of the loudspeaker

Script 3, "Touch", Jouer un son lorsqu'une touche est pressée :

```
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)
```

Script 4, Commutation Neopixel

```
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)
```

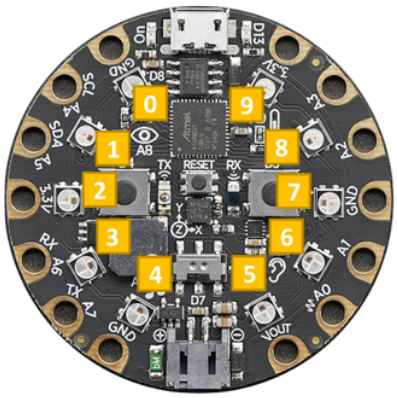


Figure 14: The numbering of the neopixels

Skript 5, Lecture du capteur de lumière

```
import board
import time
import analogio

licht = analogio.AnalogIn(board.LIGHT)

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

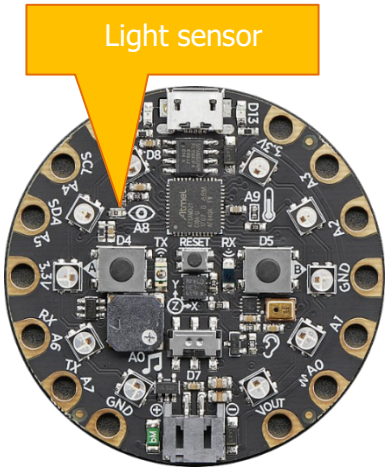


Figure 15: Location of the light sensor

Leçons 12 & 13 (90 min): Exemple de communication synchrone

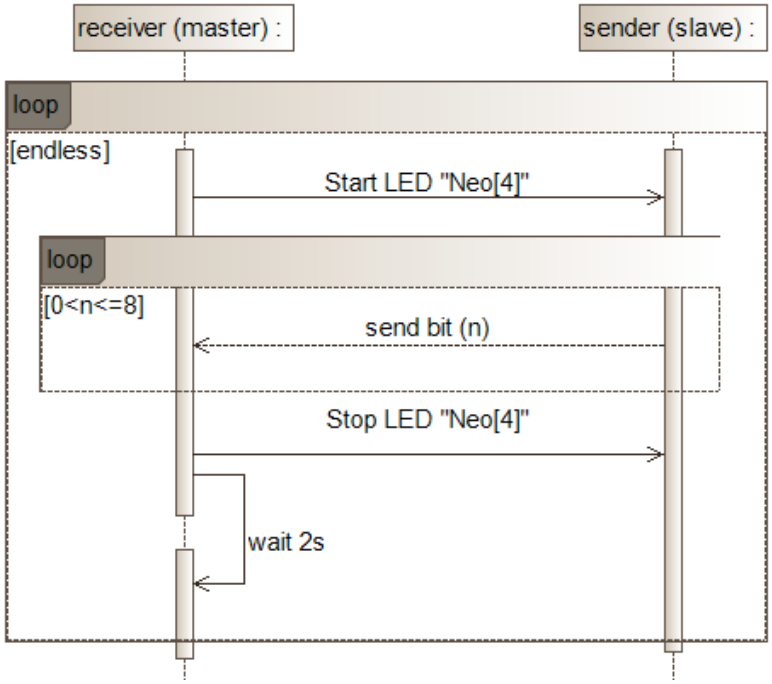


Figure 16: Diagramme de séquence UML de la communication synchrone

Code source du récepteur	Code source de l'émetteur

`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)`
	`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)`

Leçons 14 & 15 (90 min): Exemple de communication asynchrone

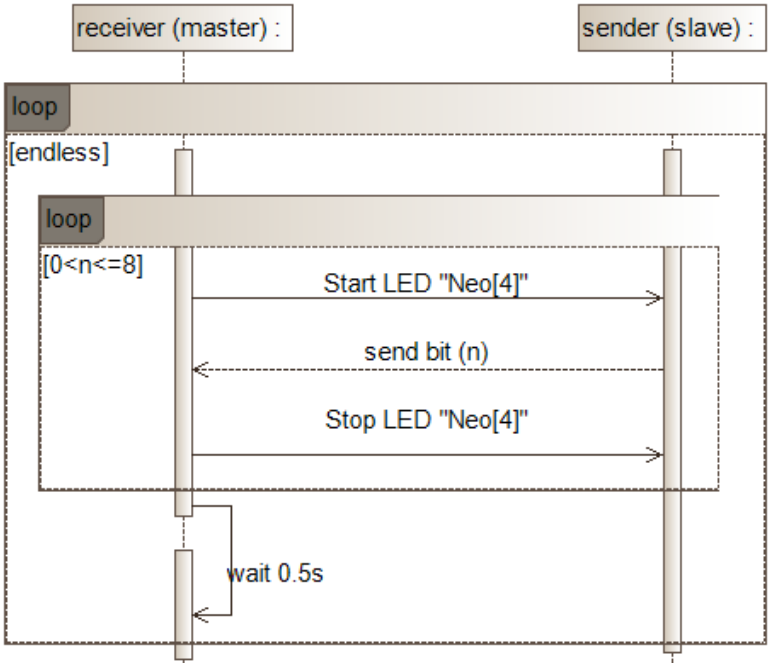


Figure 17: Diagramme de séquence UML de la communication asynchrone

Code source du récepteur	Code source de l'émetteur

`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. Retour d'information	À la fin du module, les étudiants devraient avoir développé une compréhension plus approfondie du fonctionnement de la communication série et des principes informatiques nécessaires à la mise en œuvre de cette technologie. Pendant les leçons, des aspects importants de l'électronique, de l'optique et de la construction de protocoles sont discutés.
11. Evaluation	Les élèves tiennent leur journal du travail, qui peut être révisé par l'enseignant. Les élèves peuvent également présenter les résultats de leurs expériences. De plus, un test standard en classe doit être effectué à la fin des cours.