PREMIERE GENERALE

NUMERIQUE ET SCIENCES INFORMATIQUES

DECOUVERTE DE LA CARTE RASPBERRY PICO

- ▷ **Histoire de l'informatique**
- ▷ Représentation des données : types et valeurs de base
- ▷ Représentation des données : types construits
- ▷ Traitement de données en tables
- ▷ Interactions entre l'homme et la machine sur le Web
- ▷ **Architectures matérielles et systèmes d'exploitation**
- ▷ Langages et programmation
- ▷ Algorithmique

1 – MISE EN SITUATION

En l'an 2032, une planète habitée a été découverte aux confins de la galaxie, et baptisée Algoréa. Dix ans plus tard, vous êtes envoyé(e) en mission autour de cette planète dans le but d'établir un tout premier contact avec ses habitants. Malheureusement, la comète Hal a percuté votre vaisseau alors que vous vous approchiez d'Algoréa. Vous avez été contraint(e) de vous éjecter en urgence dans une capsule de sauvetage. Vous atterrissez sur la planète avec pour seul bagage un robot de maintenance.

Isolé(e) sur cette planète lointaine, vous allez donc rencontrer ses habitants, découvrir leurs coutumes et créer des liens afin d'entamer une relation pacifique en attendant que la prochaine mission vienne vous récupérer. Vous allez pour cela profiter des capacités de votre robot ! Celui-ci dispose d'une interface homme/machine (IHM), qui vous permettra de consulter sur un écran la notice de fonctionnement du robot. Vous pourrez également programmer le robot à partir de cette IHM.

Vous remarquez que le manuel propose de commander le robot à l'aide d'une **carte Raspberry Pico** en écrivant du code sur un clavier. Vous avez également à votre disposition l'ordinateur de bord de votre capsule de sauvetage.

2 – DECOUVERTE DU ROBOT ET DE LA CARTE RASPBERRY PICO

En lisant le manuel, vous apprenez que le robot est équipé d'une **carte Raspberry PICO**. Il s'agit d'une plateforme basée sur une carte à microcontrôleur. Cette carte intègre le SoC **RP2040** crée par la fondation Raspberry. Cette carte est dédiée à l'internet des objets (IoT) . Il existe plusieurs solutions permettant de programmer cette carte : **MicroPython** (version du langage Python adaptée à la programmation des µC), en **langage C++**, …

La version de la carte Arduino installée sur le robot est la version « **Raspberry PICO W**» qui intègre une interface Wi-Fi.

2.1 – Caractéristiques techniques

Avant de pouvoir commencer à programmer ce robot, vous devez connaitre les différentes caractéristiques techniques de cette carte Raspberry PICO. Pour cela vous pouvez consultez la notice « Carte Raspberry PICO ».

Mission 1

A partir document de présentation de la carte, **donner** les valeurs des caractéristiques suivantes.

Tension de fonctionnement : 3.3V

Intensité maximale consommée en fonctionnement : 3W

Tension d'alimentation maximale : 5V

Intensité maximale délivrée sur une broche : 5V

Intensité maximale délivrée sur l'ensemble des broches : 3.3V

Fréquence de fonctionnement : ……

Taille mémoire SRAM : 512Mb

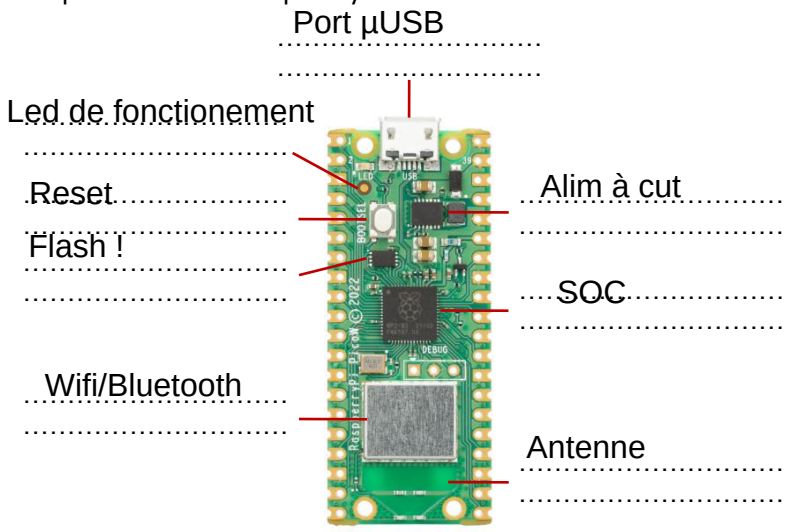
Taille mémoire Flash : 512mb

2.2 – Différents éléments de la carte Raspberry PICO

Afin de ne pas faire d'erreurs lorsque vous aurez à manipuler cette carte, vous devez connaître les différents composants qui la constituent.

Mission 2

Compléter le schéma descriptif de la carte Raspberry PICO.



2.3 – Broches numériques de la carte Raspberry PICO

La carte carte Raspberry PICO possède 26 E/S numériques (numérotées **`GPIO0`** à **`GPIO22`** et **`GPIO26`** à **`GPIO28`**). Chaque broche peut être utilisée comme une :

- **Entrée numérique** qui permet de lire une tension de **`0V`** (**`False`**) ou une tension de **`3.3V`** (**True**).
- **Sortie numérique** qui permet de fournir une tension de **`0V`** (**`False`**) ou une tension de **`3.3V`** (**True**).

Certaines de ses broches peuvent être utilisées pour d'autres fonctions spécialisées telles que la **communication série** (UART, SPI, I2C), **PWM** (Pulse With Modulation), **interruptions**.

Mission 3

Indiquer, les broches numériques réalisant les fonctions suivantes :

Communication série UART : SOC

Liaison SPI : GPIO

Liaison I2C : GPIO

500MHz

2.4 – Broches analogiques de la carte Raspberry PICO

La carte Arduino Uno possède 3 entrées analogiques **`ADC0`**, **`ADC1`** et **`ADC0`** (**`GP26`** à **`GP27`**). Il est possible de fournir une tension analogique comprise entre **`0V`** et **`3.3V`** sur chacune de ses broches.

Ces 3 broches sont connectées à un Convertisseur Analogique Numérique (CAN ou ADC en anglais de **`12 bits`**) qui permettra à la carte Raspberry PICO de récupérer un nombre numérique proportionnel à la tension analogique appliquée sur une de 3 entrées.

Mission 4

Donner la valeur numérique (binaire et décimale) correspondant aux valeurs extrêmes de la tension analogique appliquée sur une des entrées analogiques.

`0V` ⇒ **`0b`**000000000000 = 0

`3.3V` ⇒ **`0b`**111111111111 = 4095

3 – ENVIRONNEMENT DE PROGRAMMATION DE LA CARTE RASPBERRY PICO

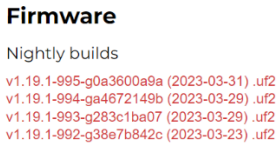
Maintenant que vous avez découvert les aspects techniques de la carte Raspberry PICO, il faut vous intéresser à son environnement de programmation.

3.1 – Installation de MicroPython

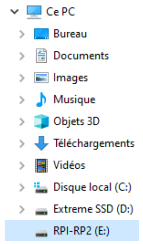
La carte Raspberry PICO sera programmée en MicroPython qui est un interpréteur Python optimisé pour les microcontrôleurs. Les scripts Python sont directement exécutés sur les cartes Raspberry PICO. Pour cela, il suffit de flasher la carte Raspberry PICO avec MicroPython et d'utiliser un IDE Python (par exemple Thonny IDE) pour éditer des scripts Python et les transférer sur la Raspberry PICO.

Pour flasher MicroPython sur une carte Raspberry PICO avec l'IDE Thonny il faut suivre les étapes suivantes :

1 Télécharger la dernière version stable du firmware MicroPython (fichier **`.uf2`**) sur le site : https://micropython.org/download/rp2-pico-w/

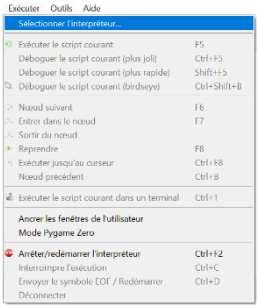


2 Maintenir le bouton **`BOOSEL`** appuyé et connecter la carte Raspberry PICO W à l'ordinateur. La carte Raspberry PICO W apparaît alors dans l'explorateur de fichiers comme une clé USB classique qui s'appelle **`RPI-RP2`**. Copier/coller le firmware MicroPython (fichier **`.uf2`**) téléchargé dans le dossier **`RPI-RP2`**. Le dossier se ferme et la carte Raspberry PICO W redémarre sur MicroPython.

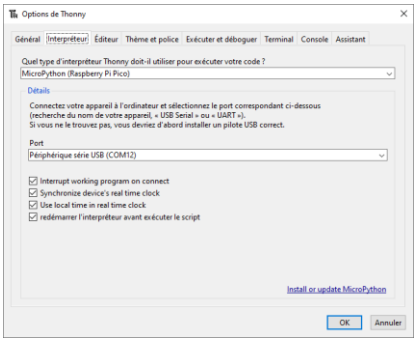


3 Lancer l'IDE Thonny.

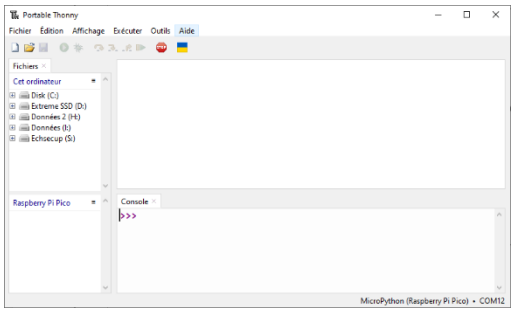
4 Cliquer sur « **`Sélectionner l'interpréteur…`** » dans le menu « **`Exécuter`** ».



5 Choisir l'interpréteur « **`MicroPython (Raspberry Pi Pico)`** » et indiquer le port COM utilisé par la carte Raspberry PICO puis cliquer sur « **`OK`** ».



6 La carte Raspberry PICO apparait alors dans l'onglet « **`Fichiers`** » de Thonny.



Mission 5

Flasher MicroPython sur la carte Raspberry PICO. Vérifier que La carte Raspberry PICO apparait alors dans l'onglet « **`Fichiers`** » de Thonny.

Vous avez maintenant toutes les informations techniques nécessaires à l'utilisation et la manipulation de la carte Raspberry PICO aini qu'une carte prête à être programmer. Pour poursuivre votre mission vous devez **maintenant apprendre à programmer cette carte Raspberry PICO**.

3.2 – Premier programme python sur la carte Raspberry PICO

Maintenant que tous les outils sont installés, on va pouvoir créer un premier programme et l'envoyer sur la carte Raspberry PICO.

Mission 6

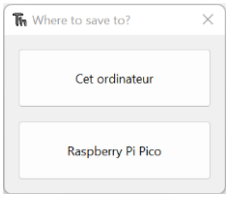
1. **Editer**, sous Thonny, le programme suivant :

```
from machine import Pin
import time

pin_led = Pin('LED', mode = Pin.OUT) # Broche de la LED built-in placée en sortie

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

2. **Enregistrer** le script dans « **`cet ordinateur`** » sous le nom **`mission6.py`**.



3. **Exécuter** le programme en cliquant sur « **`Exécuter le script courant`** » dans le menu « **`Exécuter`** » ou bien en cliquant sur l'icône . **Vérifier** que la LED rouge clignote

4 – LECTURE D'UNE ENTREE NUMERIQUE

Vous partez à la découverte de la planète. Après un certain temps de marche vous vous trouvez nez à nez avec un groupe d'habitants. Ces habitants regardent votre compagnon bizarrement. Pour les rassurer, vous vous dites qu'il serait de bon ton que votre robot affiche quelques mots sur son écran.

Le robot possède un certain nombres de boutons permettant de le commander. Vous décidez d'utiliser ces boutons pour gérer l'affichage. Parmi ces boutons, un **bouton rouge** est connecté sur la **broche numérique GPIO20** et un **bouton bleu** connecté **broche numérique GP2IO1**.

Pour continuer votre travail, vous devez comprendre comment connaître l'état des boutons.

Lorsque un **bouton est appuyé**, une **tension `0V`** est appliquée sur la broche numérique. Cette broche reçoit donc l'état logique **`False`**. Lorsque un **bouton est relaché**, la tension **`5V`** est appliquée sur la broche numérique. Cette broche reçoit donc un l'état logique **`True`**.

Pour connaitre l'état du bouton, il suffit donc de connaitre le niveau logique appliqué sur la broche sur laquelle est connecté le bouton.

Pour lire l'état logique d'une broche numérique, il faut au préalable configurer la broche numérique en mode « entrée » par l'intermédiaire de l'instruction **`nom_broche = Pin(num_pin, PIN.IN)`** en indiquant le numéro de la broche numérique. L'instruction **`etat = nom_broche.value()`** permettra ensuite de récupérer l'état logique de cette entrée.

Le programme suivant permet d'afficher l'état du bouton connecté sur la broche **GPIO20**.

```
from machine import Pin
import time

pin_bp = Pin(20, mode=Pin.IN)

while True:
    etat_BP = pin_bp.value()
    if (etat_BP == False):
        print("Bouton appuyé")
    else:
        print("Bouton relaché")
    time.sleep(1)
```

Mission 7

En vous aidant de l'exemple ci-dessus trouvé dans la notice, **éditer** un programme permettant d'afficher l'état du bouton rouge et l'état du bouton bleu. **Tester** le.

Mission 8

Editer un programme permettant de lire l'état des boutons bleu et rouge et permettant d'afficher en fonction de l'état de ces boutons, un message différent :

- Si le bouton bleu est le seul appuyé : afficher « Bonjour habitants de la planète Algoréa ».
- Si le bouton rouge est le seul appuyé : afficher « Nous venons de la planète Terre ».
- Si les deux boutons sont appuyés : afficher « Nous avons besoin de votre aide ».

Tester le programme.

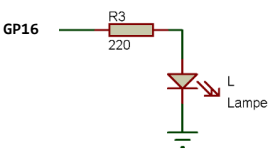
5 – PLACER UNE BROCHE NUMERIQUE A UN NIVEAU LOGIQUE

Les habitants semblent à présent rassurés, ils vous indiquent un chemin à travers une forêt épaisse qui vous permettra de rejoindre un village dans lequel vous pourrez trouver de l'aide. Vous prenez la direction de la forêt. Plus vous vous en approchez, plus vous vous rendez compte qu'elle est épaisse et sombre. Avant de vous y engager, vous cherchez une solution qui vous permettra de vous éclairer.Le robot possède une lampe sur sa face avant. Cette lampe est commandée par la broche numérique **GPIO16** de la carte Raspberry.

Remarque

Pour simuler le fonctionnement de la lampe, vous allez utiliser une DEL qui sera directement commandée par la broche **GPIO16** de la carte Raspberry. Dans le cas de l'utilisation d'une lampe, il aurait fallu utiliser un composant de puissance qui vous aurait permis de fournir une tension et un courant suffisant pour alimenter la lampe.

La DEL est connectée à la broche **GPIO16** via une résistance R3 qui permet d'ajuster son courant. Si la broche **GPIO16** est placée au **niveau logique True**, une tension de **5V** est appliquée au circuit, la **DEL s'allume**. Si la broche **GPIO16** est placée au **niveau logique False**, la tension aux bornes du circuit est de **0V**, la **DEL est éteinte**.



Pour allumer ou éteindre la DEL, il suffit de placer la broche correspondante à l'état **`True`** ou à l'état **`False`**. Pour pouvoir placer une broche numérique dans un niveau logique **`True`** ou **`False`**, il faut au préalable configurer la broche numérique en mode « sortie » par l'intermédiaire de l'instruction `nom_broche = Pin(num_pin, PIN.OUT)`. L'instruction `nom_broche.value(etat)` permettra de placer cette sortie dans l'état logique désiré.
Le programme suivant permet de faire clignoter une LED connectée sur la sortie **`GPIO16`** :

```
from machine import Pin
import time

pin_led = Pin(16, mode=Pin.OUT)

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

Mission 9

Editer le programme permettant d'obtenir d'allumer la lampe lorsque bouton rouge est appuyé et de l'éteindre lorsqu'il est relâché. **Tester** le programme.

Le fonctionnement n'est pas optimal car vous devez maintenir le bouton appuyé pour allumer la lampe. Dès que vous relachez le bouton, elle s'éteint.

Après réflexion, vous avez une idée pour régler ce problème. Pour cela il faut utiliser une variable qui contiendra l'état de la lampe. Ainsi un appui sur le bouton allumera la lampe, l'appui suivant éteindra la lampe et ainsi de suite.

Mission 10

Modifier le programme précédant afin d'obtenir le fonctionnement ci contre. **Tester** le programme.

Pour faire peur aux bêtes monstrueuses que vous aurez peut être à rencontrer dans cette forêt, pourquoi ne pas faire clignoter la lampe plusieurs fois. Pour cela il faudrait placer la sortie numérique au niveau logique `True` puis au niveau logique `False`, et répéter ceci plusieurs fois. Si on veut faire clignoter la DEL 10 fois par exemple, il faudrait appeler 10 fois les fonctions `pin_led.value(True)` et `pin_led.value(False)`. Le programme risque d'être un peu long. En lisant le manuel, vous apprenez que pour simplifier la programme il est possible d'utiliser des boucles notamment la boucle `for`.

Mission 11

Editer un programme permettant d'obtenir le fonctionnement suivant :

- Un appui sur le bouton rouge permet d'allumer la lampe.
- Un nouvel appui sur le bouton rouge permet d'éteindre la lampe.
- Un appui sur le bouton bleu permet de faire clignoter la lampe 10 fois.

Tester votre programme.

6 – SORTIES PWM

Vous voilà mieux équipé pour traverser cette forêt. Cependant le fonctionnement n'est pas encore optimal. Quelque soit la luminosité, la lampe éclaire toujours avec la même intensité lumineuse. Vous vous replongez à

nouveau dans le manuel du robot et de la carte Raspberry PICO. Vous découvrez l'existence de la méthode `duty_u16(rapport_cyclique)`.

Cette fonction permet de fournir un signal rectangulaire dont il est possible de faire varier la durée à l'état haut, c'est-à-dire son **rapport cyclique**. On parle de **PWM** (**Pulsed Width Modulation**) ou **MLI** (**Modulation par Largeur d'impulsion**).

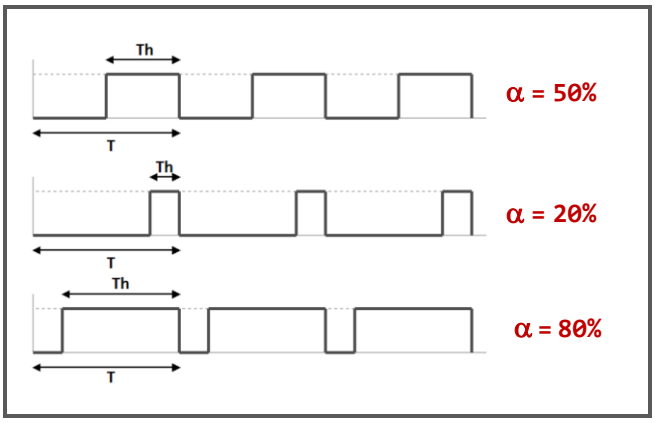
Le rapport cyclique est donné par la relation suivante :

$$\alpha = \frac{Th}{T}$$

La tension moyenne obtenue pour ce signal rectangulaire est donnée par :

$$V_{moy} = \alpha \times V_{CC}$$

Donc en modifiant ce rapport cyclique, on peut faire varier la tension moyenne du signal rectangulaire généré sur la broche PWM et donc la luminosité de la lampe.



Le Raspberry Pi Pico possède 8 paires de blocs PWM (**`PWM0`** à **`PWM7`**), soit **16 sorties PWM** au total. Ces sorties PWM peuvent, théoriquement, être rattachées à n'importe quelle broche de la carte. Chaque bloc PWM peut avoir une fréquence indépendante. Sur la base d'un µC RP2040 fonctionnant à **`125 MHz`**, les fréquences PWM possibles sur les sorties d'un Raspberry Pico sont comprises entre **`7.5 Hz`** à plusieurs Mégahertz.

MicroPython fait abstraction du hardware et sélectionne automatiquement un bloc PWM disponible. La configuration consiste à associer un objet **`PWM`** à une broche et de choisir la fréquence PWM. Par exemple le script ci-dessous permet de définir une sortie PWM sur la broche `GP16` avec une fréquence de `50 Hz`.

```
from machine import PWM, Pin
led_pwm = PWM(Pin(16, mode=Pin.OUT))
led_pwm.freq(50)
```

Pour générer le signal PWM, on dispose de la méthode `duty_u16(rapport_cyclique)`. Le paramètre `rapport_cyclique` doit être égal à **`0`** pour un rapport cyclique de `0%` et `65535` pour un rapport cyclique de `100%`. Par exemple pour fixer un signal PWM avec un rapport cyclique de `50 %`.

```
led_pwm.duty_u16(32768)
```

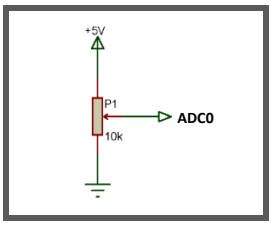
Mission 12

1. **Modifier** le programme précédent que lors de l'allumage normal de lampe (hors clignotement), elle soit commandée par un signal PWM avec un rapport cyclique de `60%`.
2. **Tester** ensuite votre programme avec un rapport cyclique de `20%` puis de `80%`.

7 – LIRE UNE TENSION SUR LES ENTREES ANALOGIQUES

Il faut trouver une solution qui permette de régler la luminosité sans avoir à reprogrammer la carte Arduino. Pour cela vous allez utiliser un potientiomètre connecté sur la broche `ADC0`. Il s'agit d'une des 3 broches analogiques qui sont connectées à un Convertisseur Analogique Numérique, 12 bits, interne à la carte Raspberry PICO.

Le potentiomètre P1 est connecté sur la broche ADC0. En tournant le potentiomètre, on fait varier la tension appliquée sur la broche ADC0 **entre 0V et 3.3V**.



Pour utiliser les entrées analogiques, il faut utiliser le sous-module **ADC** de la librairie **machine**. Il faut ensuite créer en objet **ADC** en spécifiant le numéro de la broche analogique ou le numéro de la voie.

```
from machine import ADC
adc_pin = Pin(26, mode=Pin.IN)
adc = ADC(adc_pin)  # Utilisation de la broche GPIO26 (ADC0)
```

```
from machine import ADC
adc = ADC(0)   # Utilisation de la voie ADC0 (GPIO26)
```

Chaque ADC renvoie un nombre numérique **N** de **16 bits** proportionnel à la tension appliquée **Ve** sur l'entrée du convertisseur.

$$N = Ve \times \frac{65535}{3.3}$$

Pour récupérer ce nombre numérique N, il faut utiliser l'instruction suivante :

```
N = adc.read_u16();
```

Mission 13

Donner la valeur numérique (binaire et décimale) obtenue pour une tension de 0 V puis celle obtenue pour une tension de 3.3 V.

Editer un programme permettant d'afficher la valeur numérique correspondant à la tension appliquée sur la broche ADC0. **Tester** le programme et **vérifier** que les valeurs obtenues pour les positions extrêmes du potentiomètre correspondent aux valeurs précédentes.

Vous voulez utiliser cette valeur pour le signal PWM qui commande la lampe.

Mission 14

Justifier que les valeurs fournies par la méthode adc.read_u16() peuvent être utilisées comme valeurs de la commande PWM.

Modifier le programme de la mission 12 afin que la luminosité de la Lampe soit proportionnelle à la position du potentiomètre. **Tester** le programme.