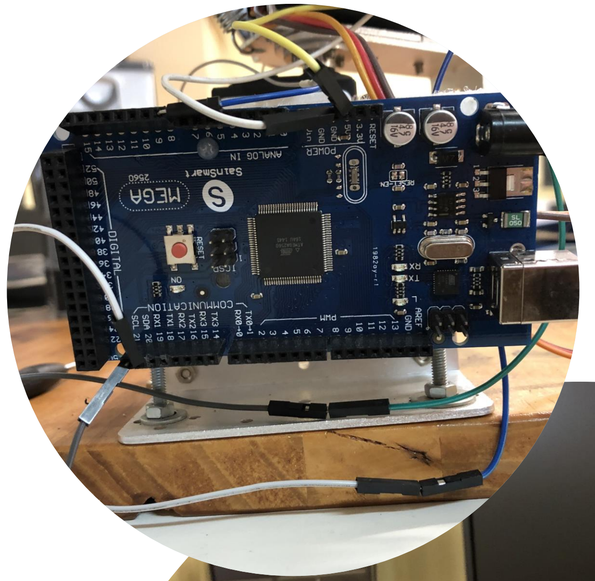
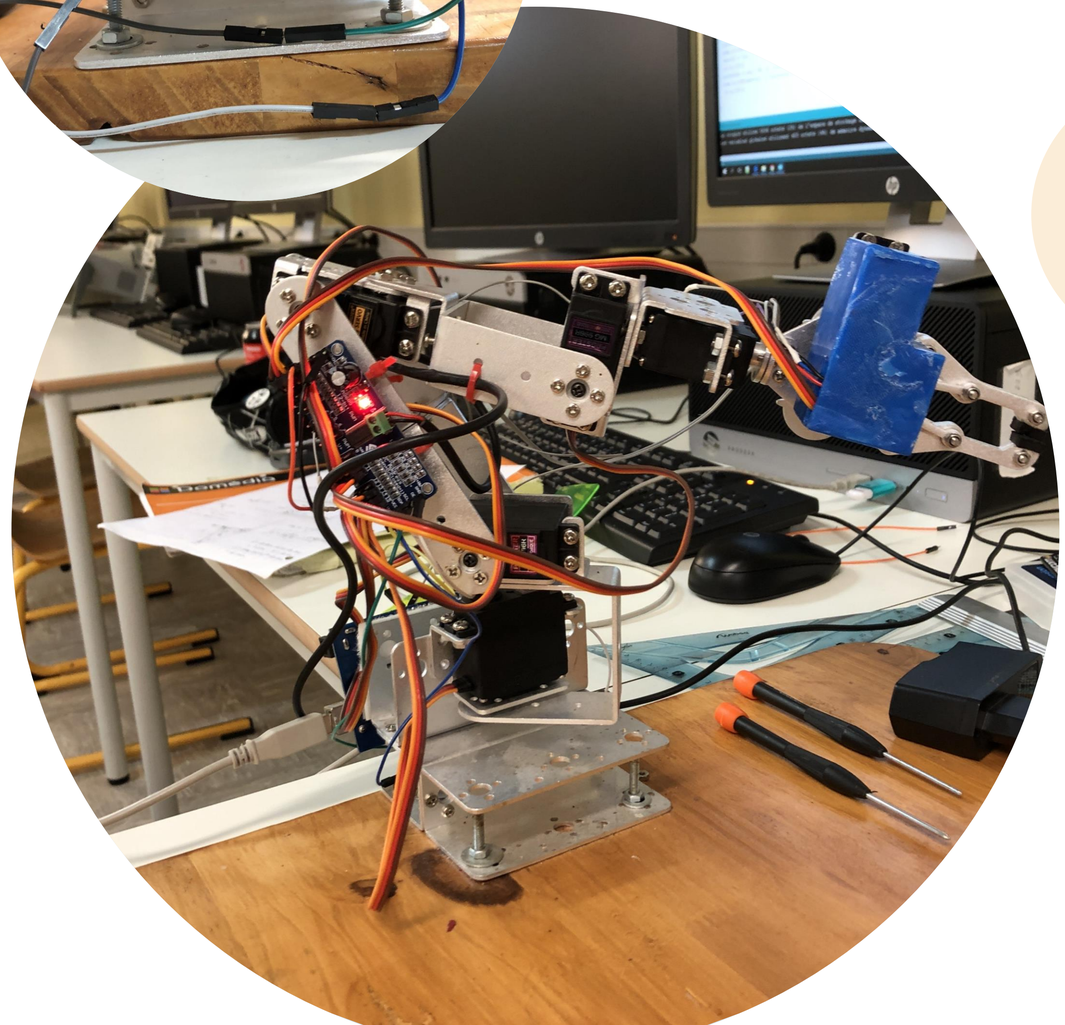
JustThink**IT**

Projet SI 2020 – Quentin PETIT





Introduction

Présentation globale du projet, Problématique,
étude du besoin…

Introduction

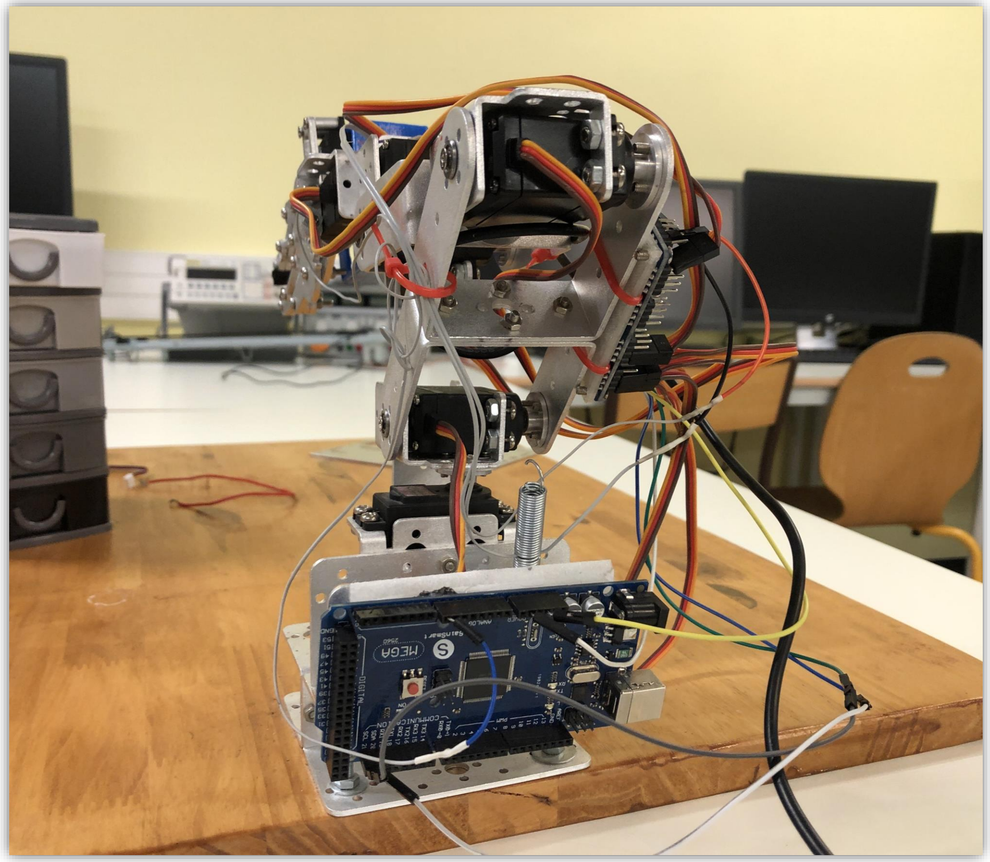
Présentation globale du projet

Objectif:

Piloter **un bras robotisé** à l'aide d'un casque NeuroSky mesurant **les ondes cérébrales**

Problématique

Comment aider **une personne** souffrant d'un handicap moteur à acquérir **plus d'autonomie** ?

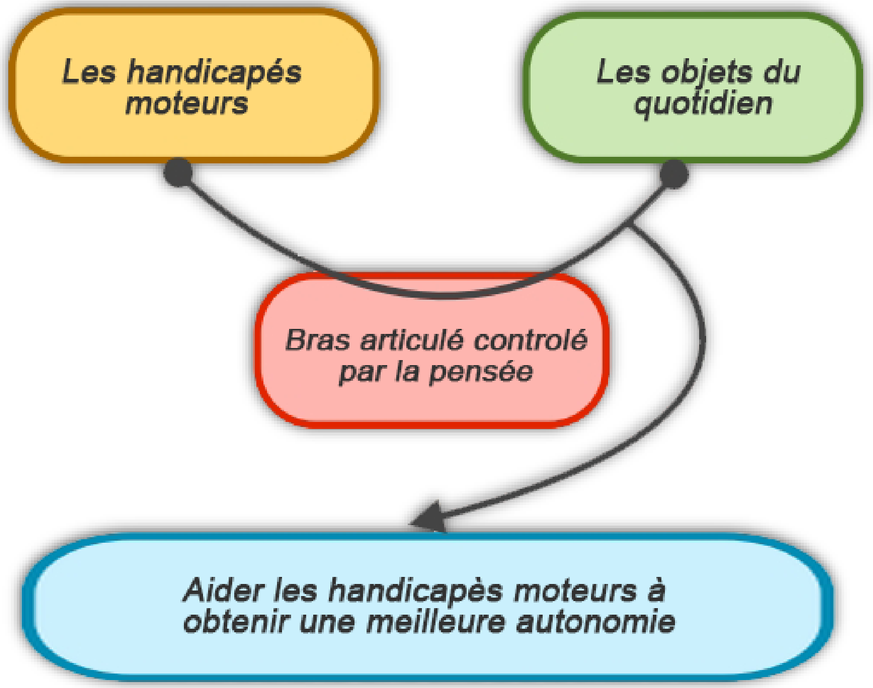


Introduction

Etude du besoin

Equipements du prototype:

- 6 Servomoteurs pour le bras articulé
- Carte en I2C PCA9685 pour commander les Servo
- Cartes Programmables (Arduino, Pycom)
- Piles pour l'alimentation du casque
- Batterie pour l'alimentation du bras
- Librairie PWM, Sklearn..



Introduction

Problématique individuelle

Abel

Comment avoir un déplacer du bras le plus fidèle possible à la demande numérique ?

Quentin

Comment traduire les informations transmises par le casque pour ordonner les déplacements ?

Elodie

Comment déplacer le bras sur les axes X,Y et Z de façon uniforme ?

Partie 1: Pilotage du bras

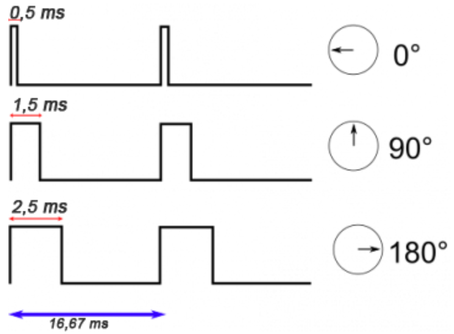
Comment avoir un déplacer du bras le plus fidèle possible à la demande numérique ?

Partie 1: Pilotage du bras

Protocole d'expérimentation pour déterminer les impulsions minimales et maximales

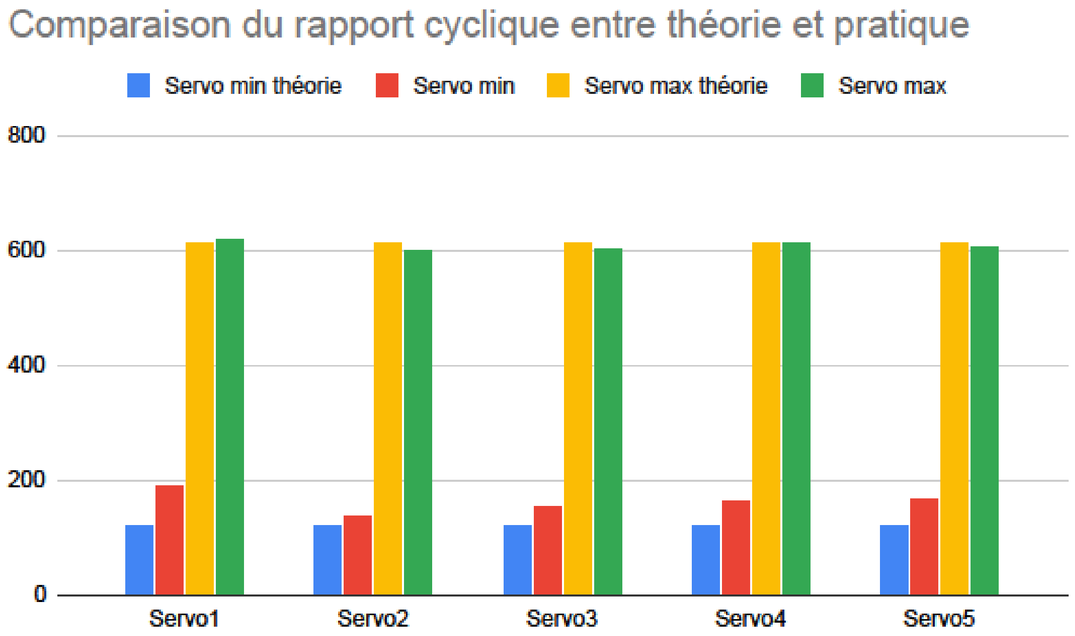
Comment fonctionne le Servomoteur ?

- Rapport cyclique: Toff = 4096 * a (avec a = ms/16,67)
- Toffmin=0.5/16.67*4096=123
- Toff max=2.5/16.67*4096=614



Etapes de la détermination?

1) Débuter avec les valeurs théoriques calculées (123,614)
2) Constater les résultats et ajuster les valeurs si nécessaire
3) Faire de même avec les servomoteurs restants
4) Comparer avec les valeurs théoriques



Partie 1: Pilotage du bras

Faire bouger le bras à l'aide de Servomoteurs

Code Arduino pour le déplacement d'un servo moteur

Déclaration des variables et initialisation de la fréquence

Deux fonctions qui permettent le contrôle du bras

- ***positionInitiale***: mettre le bras en position de départ
- ***defPosition***: déterminer une longueur d'impulsions selon l'angle prédéfini (PWM)

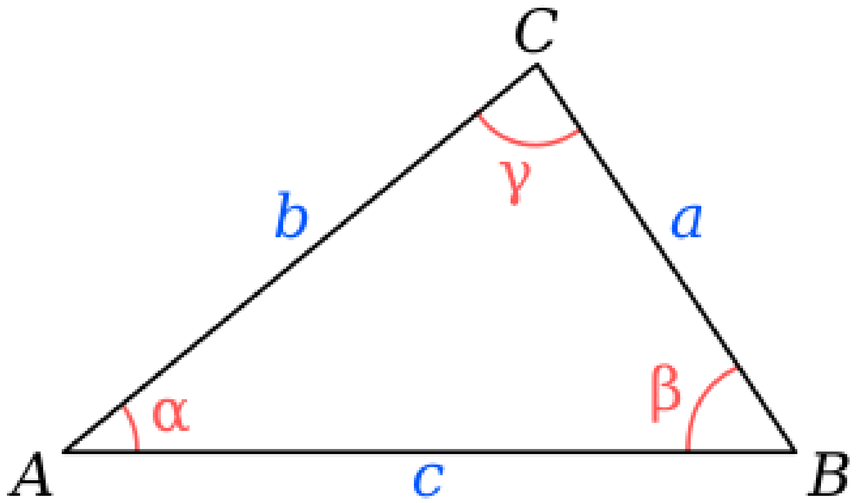
pulseLen qui permet de convertir l'angle en impulsions

pwm.setPWM qui permet d'envoyer l'information

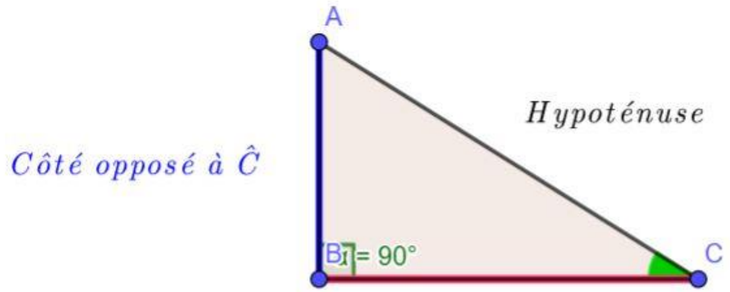
```
int servo;
int pulseLen;

void setup() {
  Serial.begin(9600);
  pwm.begin();
  pwm.setPWMFreq(60);
}

void loop() {
  positionInitiale();
}

void positionInitiale() {
  defPosition(90, 15, SERVOMIN15, SERVOMAX15);
  defPosition(125, 14, SERVOMIN14, SERVOMAX14);
  defPosition(0, 13, SERVOMIN13, SERVOMAX13);
  defPosition(90, 11, SERVOMIN11, SERVOMAX11);
}

void defPosition(int degree, int servo, int servomin, int servomax) {
  pulseLen= map( degree, 0, 180, servomin, servomax );
  pwm.setPWM(servo, 0, pulseLen);
  delay(15);
}
```

Partie 1: Pilotage du bras

Rappel trigonométrie & AlKashi



$$\begin{cases} a^2 = c^2 + b^2 - 2 \cdot c \cdot b \cdot cos(\alpha) \\ b^2 = c^2 + a^2 - 2 \cdot c \cdot a \cdot cos(\beta) \\ c^2 = b^2 + a^2 - 2 \cdot b \cdot a \cdot cos(\gamma) \end{cases}$$



$$Sinus\ \hat{C} = \frac{longueur\ du\ côté\ opposé\ à\ \hat{C}}{longueur\ de\ l'hypoténuse}$$

$$Cosinus\ \hat{C} = \frac{longueur\ du\ côté\ adjacent\ à\ \hat{C}}{longueur\ de\ l'hypoténuse}$$

$$Tangente\ \hat{C} = \frac{longueur\ du\ côté\ opposé\ à\ \hat{C}}{longueur\ du\ côté\ adjacent\ à\ \hat{C}}$$

Partie 1: Pilotage du bras

Calcul d'angle à fournir pour les Servomoteurs

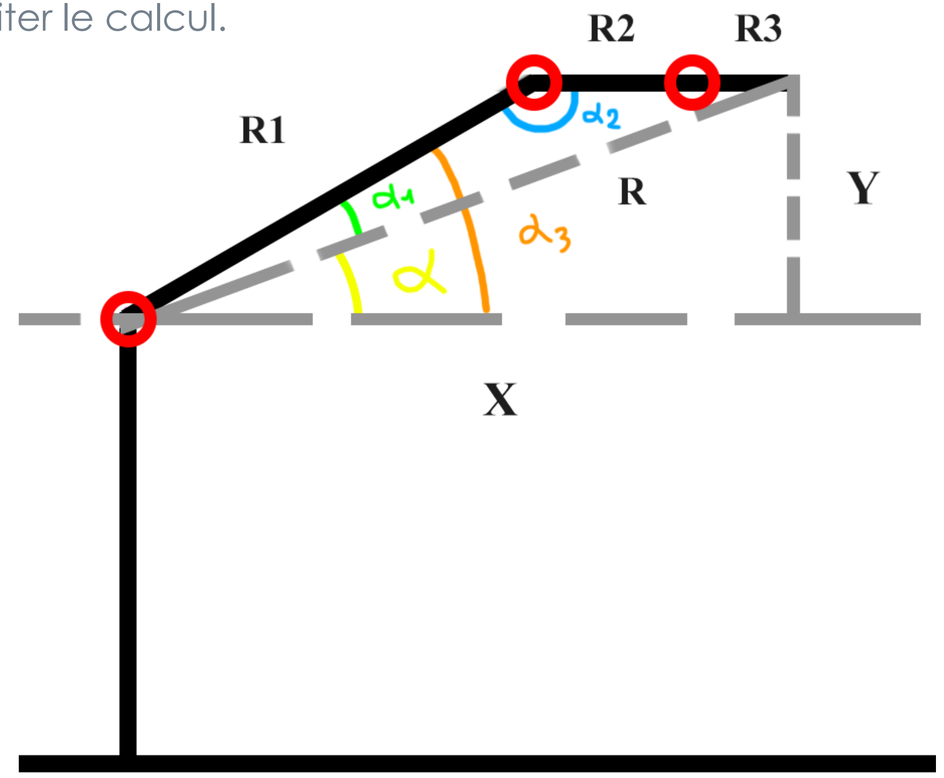
A l'aide d'Al Kashi, on peut:

- Déterminer a3 et a2 à l'aide de la trigonométrie et du théorème d'Al Kashi
- En supposant le dernier servomoteur toujours parallèle au sol pour faciliter le calcul.

$$R = \sqrt{X^2 + Y^2} \qquad \alpha 3 = \alpha + \alpha 1$$

$$\alpha = \tan^{-1}\left(\frac{Y}{X}\right) \qquad \alpha 1 = \cos^{-1}\left(\frac{(R2+R3)^2 - R1^2 - R^2}{-2R1 \cdot R}\right)$$

$$\alpha 2 = \cos^{-1}\left(\frac{R^2 - R1^2 - R2^2 - R3^2}{-2R1 \cdot R2 \cdot R3}\right)$$



Partie 1: Pilotage du bras

Bouger le bras selon 4 indications

CODE CALCUL D'ANGLE POUR LE BRAS SELON DROITE GAUCHE OU RIEN

Partie 2: Traduction des ondes

Comment traduire les informations transmises par le casque pour ordonner les déplacements ?

Partie 2: Traduction des ondes

Récupération de la trame par le module HC-05 bluetooth

Paramètres à implémenter

AT+UART=« 57600 »

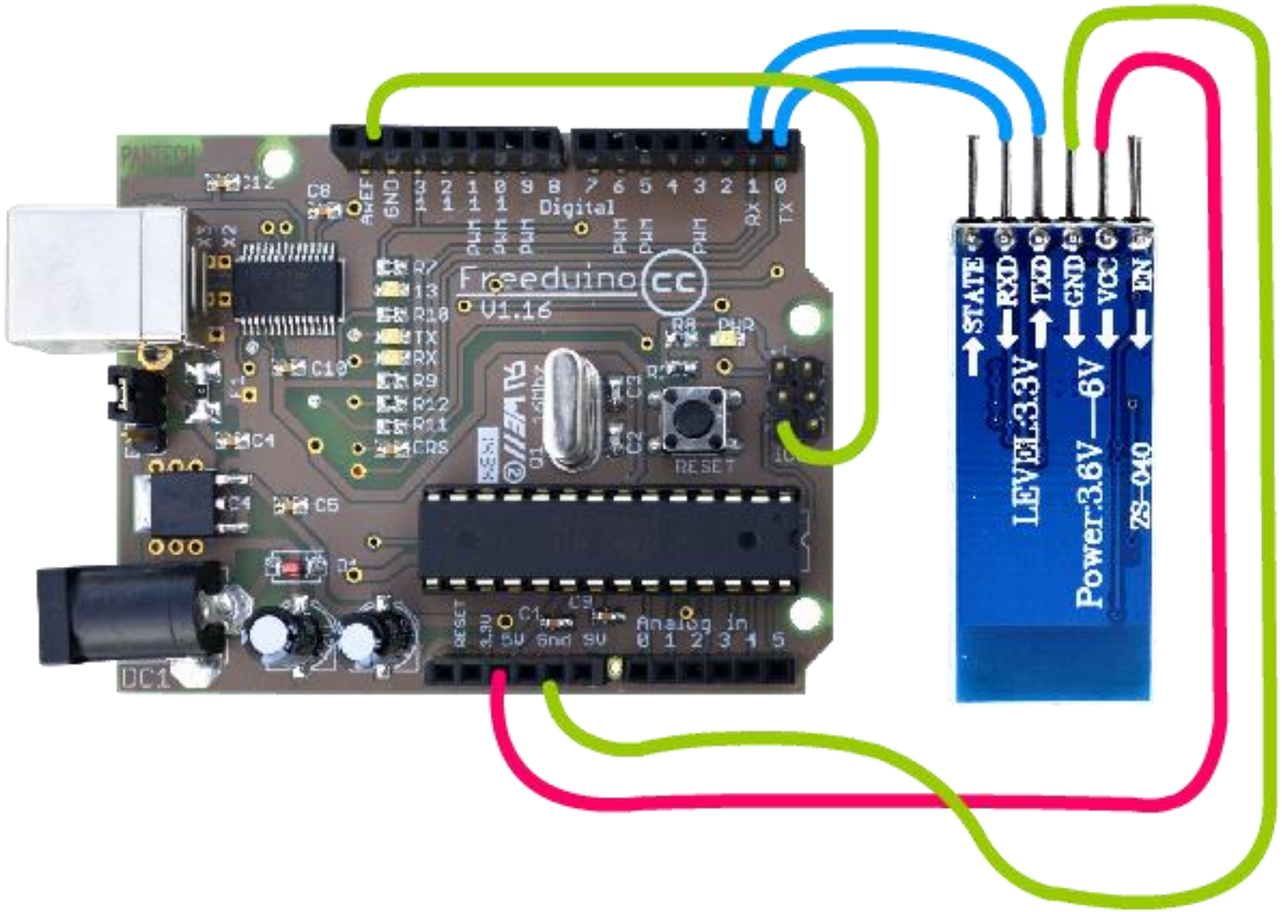
AT+ROLE=« 1 »

AT+PSWD=« 1234 »

AT+BIND=« Mac Adress »

AT+CMODE=« 1 »

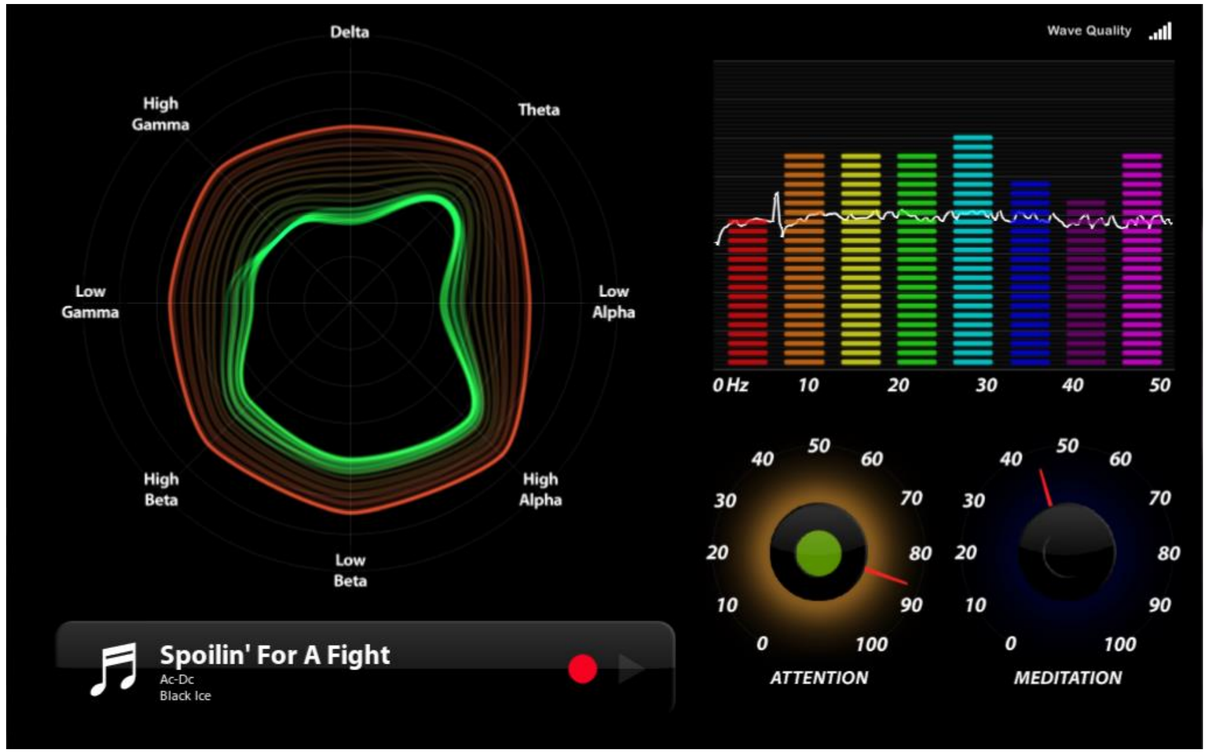
```
import machine

uart1 = machine.UART(1, 57600)
while True:
    if (uart1.any()>0):
        print(uart1.read())
```



Partie 2: Traduction des ondes

Analyse de la trame: les différentes valeurs à recevoir

- POOR_SIGNAL : sur un octet (de 0-200), qualité du signal, 200: électrodes non en contact
- MEDITATION : sur un octet (de 0-100), niveau de calme ou de relaxation mental
- ATTENTION : sur un octet (de 0-100), niveau de concentration ou d'activité mentale
- EGG POWER : sur 4 octets, 8 types de bandes de fréquences EEG (alpha, bêta, gamma..)
- EYE BLINK STRENGTH : sur un octet (de 0-100), la force de clignement des yeux



Partie 2: Traduction des ondes

Analyse de la trame: le fonctionnement

Une trame d'une taille maximale de 1+1+1+169+1 = 173 octets

```
[SYNC] [SYNC] [PLENGTH]  [PAYLOAD...]    [CHKSUM]
_______________________  _____________  ____________
^^^^^^^^(Header)^^^^^^^  ^^(Payload)^^  ^(Checksum)^
```

```
0x02          -  POOR_SIGNAL Quality (0-255)
0x03          -  HEART_RATE (0-255)
                   Once/s on EGO.
0x04          -  ATTENTION eSense (0 to 100)
0x05          -  MEDITATION eSense (0 to 100)
0x06          -  8BIT_RAW Wave Value (0-255)
0x07          -  RAW_MARKER Section Start (0)
```

```
0x81        32  EEG_POWER: eight big-endian 4-byte
                  IEEE 754 floating point values
                  representing delta, theta, low-alpha
                  high-alpha, low-beta, high-beta,
                  low-gamma, and mid-gamma EEG band
                  power values
```

Exemple de trame type:

```
[ 0]: 0xAA  // [SYNC]
[ 1]: 0xAA  // [SYNC]
[ 2]: 0x08  // [PLENGTH] (payload length) of 8 bytes
[ 3]: 0x02  // [CODE] POOR_SIGNAL Quality
[ 4]: 0x20  // Some poor signal detected (32/255)
[ 5]: 0x01  // [CODE] BATTERY Level
[ 6]: 0x7E  // Almost full 3V of battery (126/127)
[ 7]: 0x04  // [CODE] ATTENTION eSense
[ 8]: 0x12  // eSense Attention level of 18%
[ 9]: 0x05  // [CODE] MEDITATION eSense
[10]: 0x60  // eSense Meditation level of 96%
[11]: 0xE3  // [CHKSUM] (1's comp inverse of 8-bit Payload sum of 0x1C)
```

Partie 2: Traduction des ondes

Analyse de la trame: triage de la liste

Plusieurs étapes:

1) *Vérification de la sync*
2) *Stocker les données*
3) *Trier les données*
4) *Afficher les données*
5) *Utilisation des données*

```
Casque = NeuroSky()

while True:
    Casque.GetUsefullData()
    print(*Casque.payloadData, sep=", ")
```

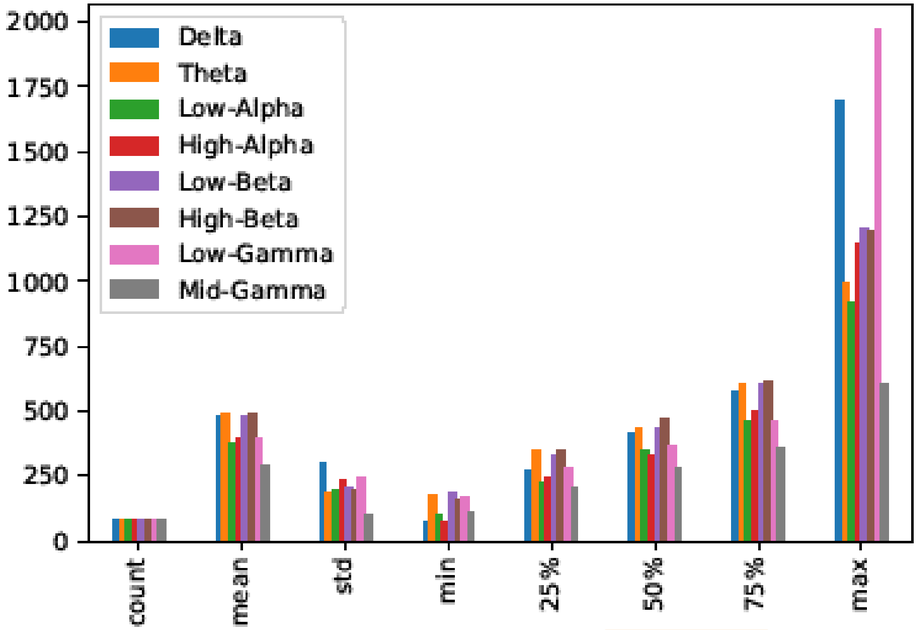
```
import pycom
import utime
import machine

class NeuroSky:

    def __init__(self):
        self.payloadData = []
        self.uart1 = machine.UART(1, 57600)

    def ReadOneByte(self):
        while self.uart1.any()==0:
            utime.sleep_us(1)
        return ord(self.uart1.read(1))

    def GetUsefullData(self):
        if self.ReadOneByte() == 170:
            if self.ReadOneByte() == 170:
                self.payloaData = []
                self.generatedChecksum = 0
                for i in range (self.ReadOneByte()):
                    self.payloadData.append(self.ReadOneByte())
                    self.generatedChecksum += self.payloadData[i]
                if self.ReadOneByte() > 4:
                    self.poorQuality = self.payloadData[1]
                    self.attention = self.payloadData[29]
                    self.meditation = self.payloadData[31]
                    self.delta = self.payloadData[4] * 100 + self.payloadData[5]*10 + self.payloadData[6]
                    self.theta = self.payloadData[7] * 100 + self.payloadData[8]*10 + self.payloadData[9]
                    self.lowAlpha = self.payloadData[10] * 100 + self.payloadData[11]*10 + self.payloadData[12]
                    self.highAlpha = self.payloadData[13] * 100 + self.payloadData[14]*10 + self.payloadData[15]
                    self.lowBeta = self.payloadData[16] * 100 + self.payloadData[17]*10 + self.payloadData[18]
                    self.highBeta = self.payloadData[19] * 100 + self.payloadData[20]*10 + self.payloadData[21]
                    self.lowGamma = self.payloadData[22] * 100 + self.payloadData[23]*10 + self.payloadData[24]
                    self.midGamma = self.payloadData[25] * 100 + self.payloadData[26]*10 + self.payloadData[27]
                    self.payloadData = [self.delta, self.theta, self.lowAlpha, self.highAlpha, self.lowBeta, self.highBeta, self.lowGamma, self.midGamma]
```

Partie 2: Traduction des ondes

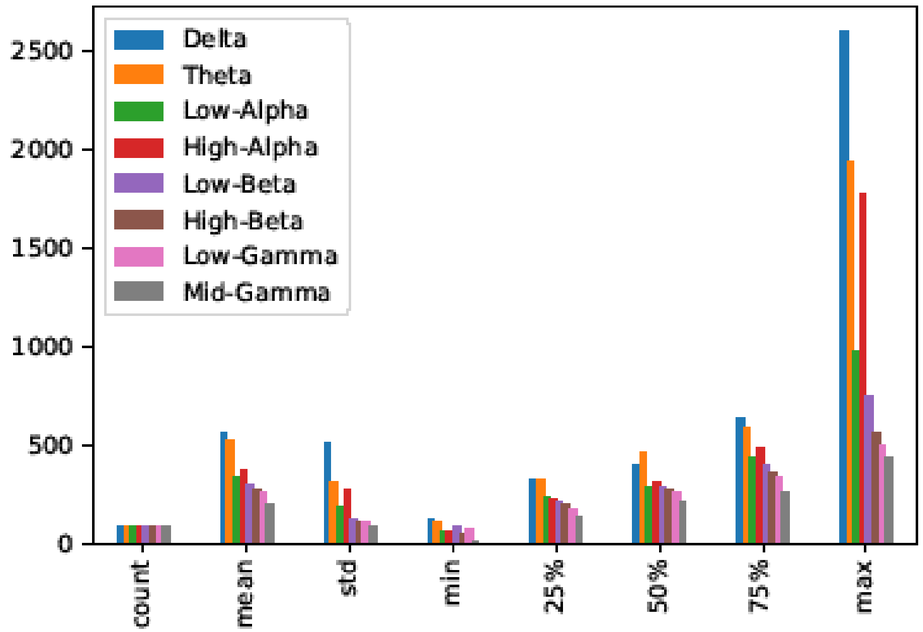
Les dispersions des 8 ondes selon la pensée

(échantillon de 200 mesures)

Pensée à gauche



Pensée à droite



Ecarts expérimentaux entre une pensée à gauche et à droite

Partie 2: Traduction des ondes

Différentes options pour classer les patterns

MiniSom

Carte auto-organisée, type de réseau neuronal artificiel qui est capable de convertir des relations statistiques complexes et non linéaires entre des éléments de données à haut dimension en relations géométriques simples.

-> Trop de faux positif, et peu adapté

MLPClassifier

Algorithme d'apprentissage supervisée qui apprend à l'aide d'une fonction par une formation

KNeighborsClassifier

Classification K-NN: Pour estimer la sortie associée à une nouvelle entrée x, la méthode des *k* plus proches voisins consiste à prendre en compte (de façon identique) les *k* échantillons d'apprentissage dont l'entrée est la plus proche de la nouvelle entrée x, selon une distance à définir.

Par exe

Partie 2: Traduction des ondes

Création du KNeighborsClassifier

- Importations des librairies
- Importation du fichier excel
- Affichage des données importées
- Création du model
- Déclaration des entrées et sorties pour le KNN
- Entrainement du model
- Taux de réussite

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
```

```
data = pd.read_excel('data.xlsx')
```

```
data.head()
```

	Classe	Delta	Theta	Low-Alpha	High-Alpha	Low-Beta	High-Beta	Low-Gamma	Mid-Gamma
0	1	524	504	357	75	469	481	259	254
1	1	355	154	445	538	416	221	351	218
2	1	732	494	729	228	341	273	282	102
3	1	472	351	282	545	466	284	272	183
4	1	355	776	181	368	281	195	85	127

```
model = KNeighborsClassifier(3)
```

```
y = data['Classe']
```

```
X = data.drop('Classe', axis=1)
```

```
model.fit(X,y)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

```
model.score(X,y)
```

```
0.8214285714285714
```

Partie 2: Traduction des ondes

Exportation du KNeighborsClassifier

Utilisation de Pickle, une librairie qui permet d'exporter les réseaux de neurones pour pouvoir les réimporter n'importe d'où. Dans notre cas, sur la pycom

```
Entrée [139]: import pickle

Entrée [140]: knnPickle = open('knnpickle_file', 'wb')

Entrée [141]: pickle.dump(model, knnPickle)

Entrée [142]: loaded_model = pickle.load(open('C:\\Users\\Quentin\\knnpickle_neurosky', 'rb'))
```

Partie 2: Traduction des ondes

Exportation du KNeighborsClassifier

- Importation des librairies
- Déclaration de l'onde à classer (x)
- Importation du model via Pickle
- Appel de la fonction de prédiction
- Afficher l'endroit où l'on pense

```
"""
Created on Tue Apr 28 18:28:12 2020

@author: Quentin
"""

import pickle
import numpy as np

x = np.array([946,1228,905,239,109,315,105,1806]).reshape(1,8)

knnmodel = pickle.load(open('knnpickle_neurosky', 'rb'))

knnmodel.predict(x)

print(knnmodel.predict(x))
if knnmodel.predict(x) == 0:
    print('Vous pensez à gauche')
elif knnmodel.predict(x) == 1:
    print('Vous pensez à droite')
else:
    print('Vous ne pensez à rien')
print(knnmodel.predict_proba(x))
print('[gauche, droite, rien]')
```

Partie 2: Traduction des ondes

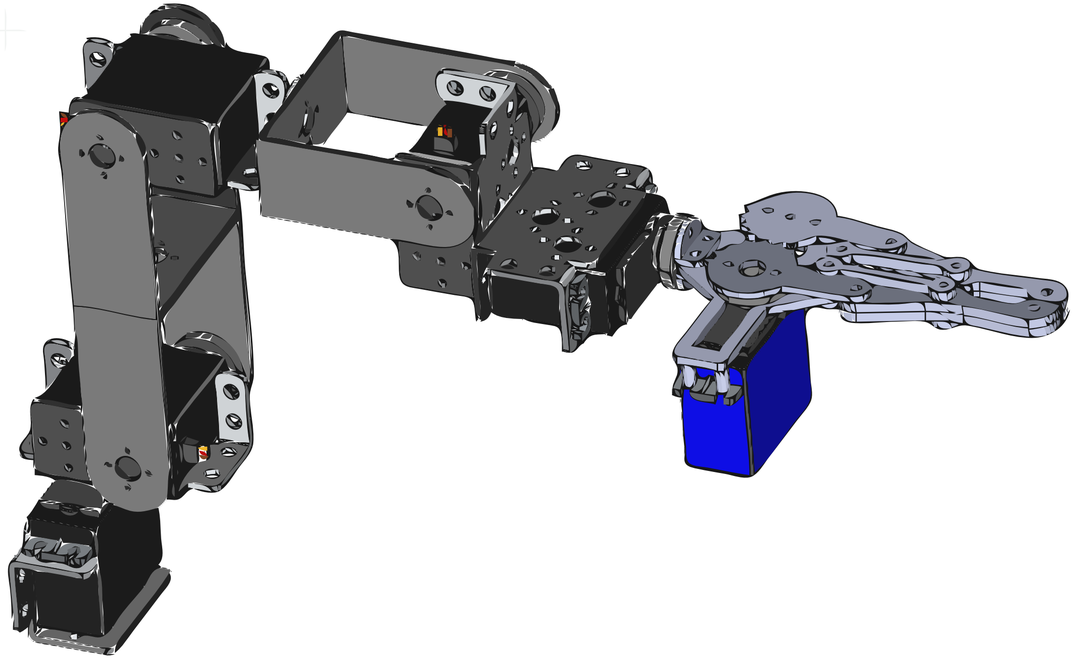
Utilisation du KNeighborsClassifier et alternative

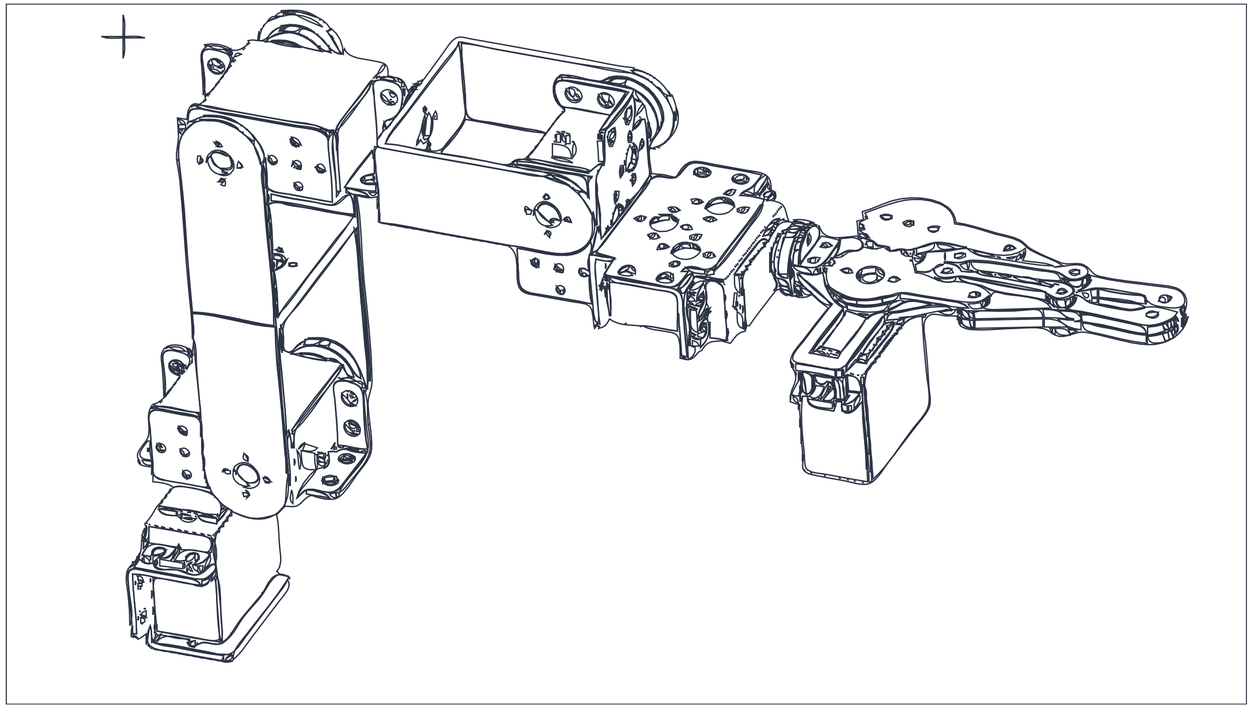
Partie 3: Gestion du bras

Comment déplacer le bras sur les axes X,Y et Z de façon uniforme ?

Partie 3: Gestion du bras

Modélisation du bras sous SolidWorks





Partie 3: Gestion du bras

Inertie du bras robotisé

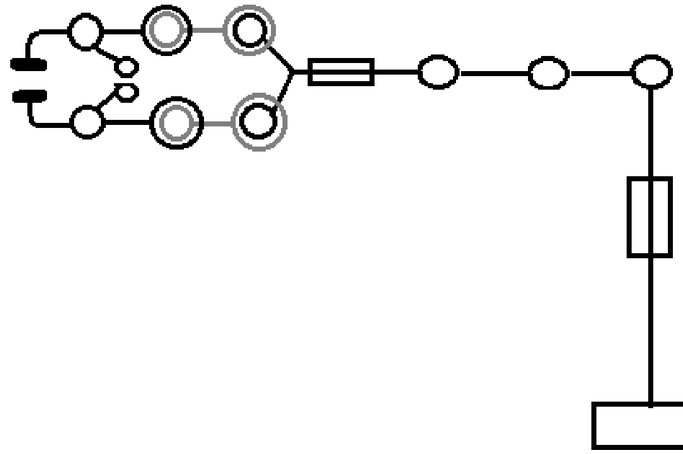
Différentes solutions concernant l'inertie du bras

- Amortisseurs mécaniques
- Compensation via un code informatique

On optera donc pour la solution informatique:

- Facilement réglable grâce aux impulsions
- Non coûteuse

Schéma cinématique du bras.

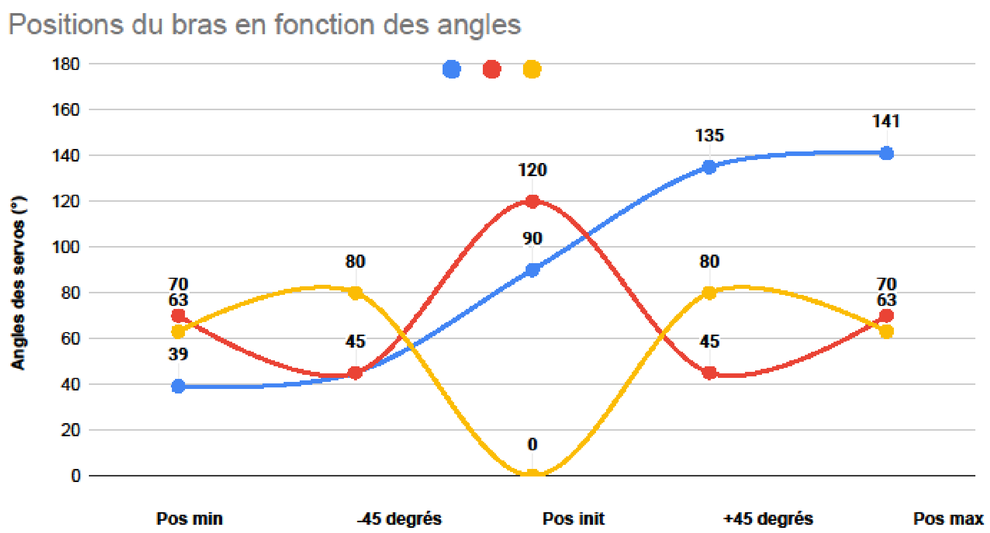


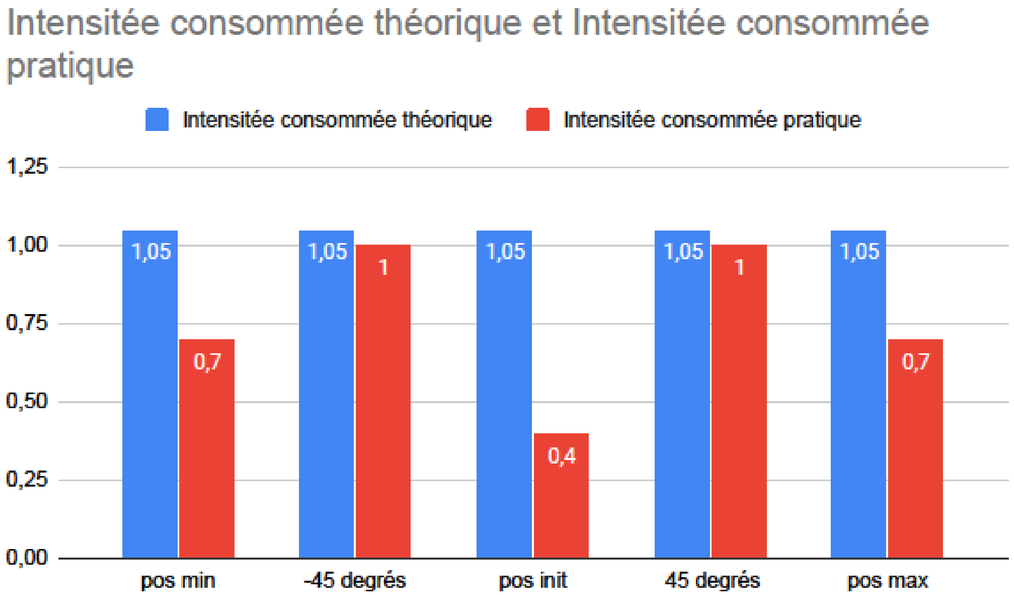
Partie 3: Gestion du bras

Consommation électrique

<u>Etudes expérimentales de la consommation électrique</u>

- Calculer au préalable les valeurs théoriques (en bleu)
- Relever les valeurs pratiques avec un multimètre selon la position du bras



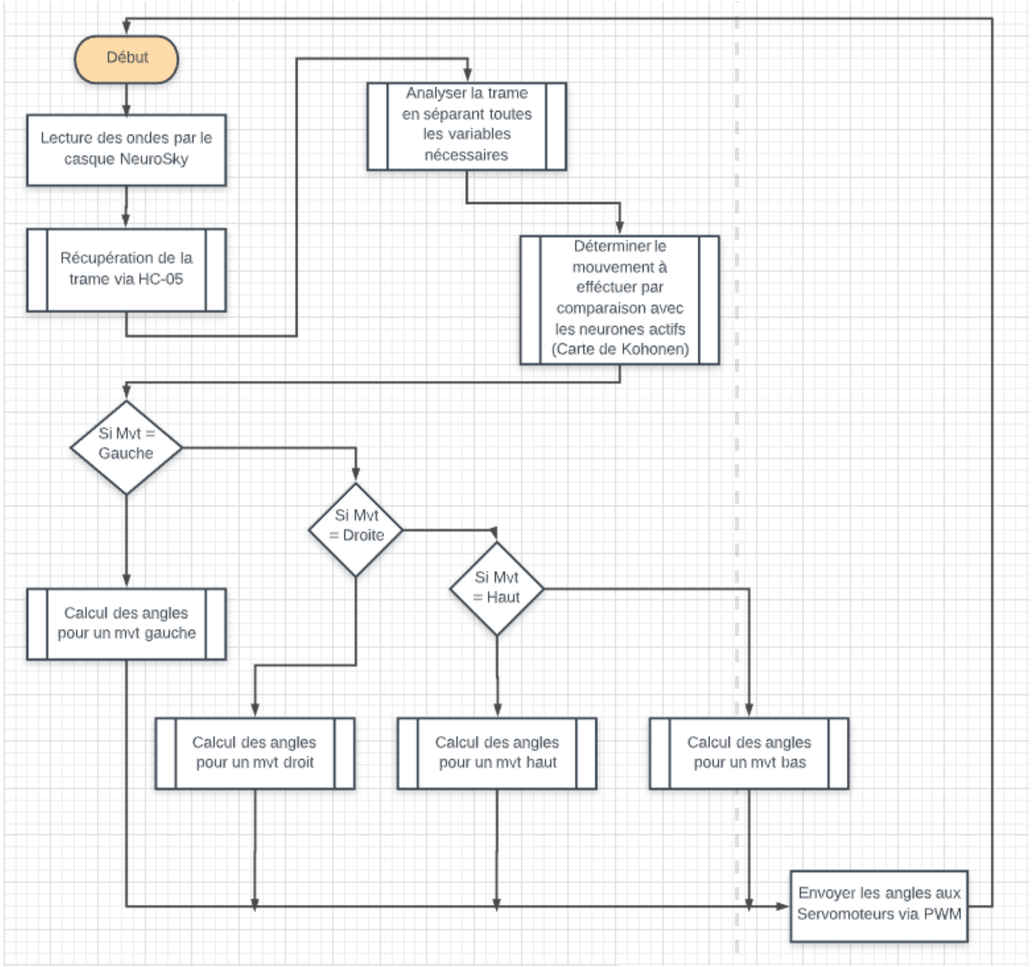


Partie 3: Gestion du bras

Fonctionnement global du projet

<u>Différentes étapes majeurs du projet</u>

- Récupérer les ondes cérébrales (EEG)
- Organiser & traiter ces ondes EEG
- Trouver le modèle le plus ressemblant à la pensée
- Calculer les angles à fournir aux Servomoteurs
- Envoyer les angles aux Servomoteurs



Conclusion

Réponse aux problématiques, poursuite…

Conclusion

Réponses aux problématiques

Comment avoir un déplacer du bras le plus fidèle possible à la demande numérique ?

A l'aide d'Al Kashi, nous avons pu facilement déterminer les angles à fournir aux Servomoteurs pour correspondre au mieux à la pensée initiale.

Comment traduire les informations transmises par le casque pour ordonner les déplacements ?

Grâce au système KNN, nous avons pu créer une intelligence artificielle qui nous détermine si nous pensions à gauche, à droite, en haut, en bas ou ne rien penser.

Comment déplacer le bras sur les axes X,Y et Z de façon uniforme ?

Par l'intermediaire du PWM, nous avons réussi à définir les impulsions pour laquelle l'inertie est la plus réduite possible pour avoir la meilleur déplacement possible.

Conclusion

Poursuite éventuelles

<u>Différentes initiatives à entreprendre pour la suite:</u>

- Finir le code en assemblant les deux premières parties pour appliquer le réseau de neurones directement au bras.

- Améliorer la probabilité de réussite en lui fournissant plus de patterns

- Optimiser au mieux le bras pour avoir une consommation plus minime

- Essayer d’optimiser le code pour avoir un déplacement plus fluide