

UNIVERSITÉ DE MONTPELLIER - FACULTÉ DES SCIENCES  
ANNÉE UNIVERSITAIRE 2022 - 2023  
L3 INFORMATIQUE  
HAI606I : PROJET DE PROGRAMMATION 2

**Faire Danser l'Eau au Rythme de la Musique sous les Lumières**

Vendredi 12 mai 2023

**Étudiants :**

AYOUB CHENINI  
THOMAS LAGUERRE  
ANESS RABIA  
ABDULLAH OBAD  
MOHAMMED DAFAOUI  
ILYASS EL FARSSI

**Encadrant :**

SERIAI-ABDELHAK



# Sommaire

<b>1.Introduction</b>	<b>2</b>
1.1 Contexte général du projet . . . . .	2
1.2 Objectif et cahier des charges . . . . .	2
1.3 Qu'est ce qu'une fontaine ? . . . . .	3
<b>2.Tехnologies utilisées</b>	<b>4</b>
2.1 Partie matériel . . . . .	4
2.2 Partie logiciel . . . . .	6
<b>3.Développement de La fontaine</b>	<b>9</b>
1. La conceptualisation de la Fontaine . . . . .	9
2. Émulation du fonctionnement de la Fontaine . . . . .	12
3. Intégration de la partie musique . . . . .	15
4. Interface graphique . . . . .	17
<b>4.Algorithmes et Structures de Donnée</b>	<b>18</b>
<b>6.Expérimentation réel et analyse des résultats</b>	<b>20</b>
<b>7.Gestion du projet</b>	<b>24</b>
Outils du travail en collaboration . . . . .	24
Répartition du travail dans le temps . . . . .	24
<b>8.Conclusion</b>	<b>25</b>
<b>9.Bibliographie</b>	<b>26</b>
<b>10.Annexe</b>	<b>27</b>

# 1. Introduction

## 1.1 Contexte général du projet

Les fontaines musicales sont de plus en plus populaires dans le monde entier en raison de leur capacité à combiner les éléments visuels et sonores de manière harmonieuse.

Cependant, la conception et la mise en place d'une fontaine musicale nécessitent une expertise technique et créative dans plusieurs domaines, tels que l'ingénierie, l'informatique et la musique.

Dans le cadre du module (HAI606I : Projet de Programmation 2), nous avons choisi de concevoir une fontaine musicale interactive qui serait capable de synchroniser les jets d'eau avec la musique.

Cette fontaine sera contrôlée par un nano-ordinateur (dans notre cas une Raspberry Pi B) ainsi que des modules électroniques externes pour garantir une synchronisation parfaite entre les jets d'eau et la musique. Nous allons donc devoir utiliser une combinaison de composants matériels et logiciels pour créer cette fontaine musicale interactive.

Ce rapport décrira les différentes étapes de conception et de réalisation de notre projet de fontaine musicale, ainsi que les solutions que nous avons trouvées pour surmonter les défis rencontrés et les idées que nous avons dû abandonner pour différentes raisons.

## 1.2 Objectif et cahier des charges

L'objectif de ce projet est de développer la partie logicielle et matérielle d'une mini-fontaine musicale. Voici les différentes étapes que nous avons suivies :

- Creer des modèles 3D pour les plans du bassin.
- Concevoir le bassin.
- Choisir des pompes, leds et les contrôler.
- Comprendre le fonctionnement d'une raspberry pi pour contrôler les pompes et les leds.
- Comprendre les paramètres de la musique pour contrôler le débit de l'eau des pompes et l'intensité de la lumière.
- Une interface capable de faire changer la hauteur de l'eau manuellement
- Harmoniser la danse provenant de la musique.

## 1.3 Qu'est-ce qu'une fontaine ?

Les fontaines sont des constructions, généralement accompagnées d'un bassin, de laquelle jaillit de l'eau. Elles peuvent être naturelles ou alimentées par un réseau de distribution d'eau.

Les fontaines naturelles fonctionnent grâce à une source d'eau naturelle, telle qu'une rivière ou une source souterraine. L'eau s'écoule à travers un système de canaux et de bassin, créant des cascades, des jets et des effets d'eau similaires à ceux des fontaines artificielles.

Tandis que les fontaines artificielles fonctionnent en utilisant une source d'eau artificielle, par exemple un réservoir. L'eau est pompée à partir de cette source et circule ensuite à travers un système de tuyaux, de jets et de buses pour créer des effets d'eau.

Autrefois les fontaines publiques étaient avec les puits et les cours d'eau les seuls lieux d'alimentation en eau potable. Souvent situées au centre d'une place, elles constituaient un lieu d'échanges et généralement fréquenté par les ménagères et les enfants.

Dans l'histoire, les fontaines ont participé à l'hygiène publique. En limitant le risque de maladie, en donner la possibilité de laver les vêtements ou ustensiles, ou même pour s'abreuver.

Aujourd'hui même si avec l'arrivée de l'eau courante dans les foyers elles ont perdu leur usage domestique, elle n'en reste pas moins des éléments appréciés pour la décoration. Elles sont devenues des constructions artistiques et décoratives.



FIGURE 1 – Fontaine naturelle



FIGURE 2 – Fontaine artificielle

# 2. Technologies utilisées

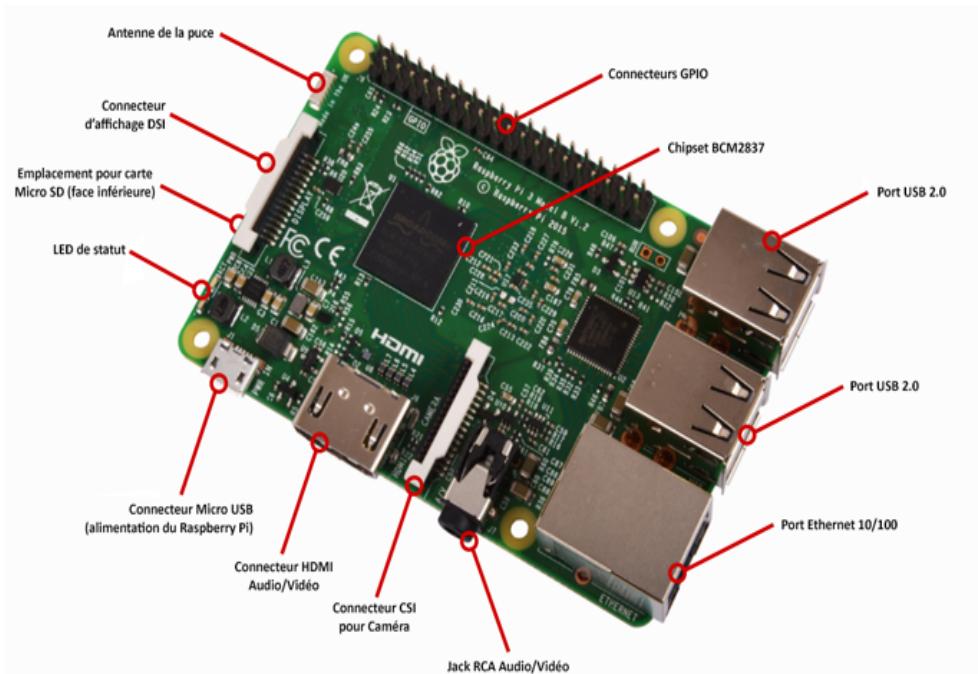
## 2.1 Partie matériel

### ❖ Raspberry pi

Une Raspberry Pi est un micro-ordinateur qui peut être utilisé pour de nombreux projets dans le domaine des systèmes embarqués, tels que l'automatisation, la robotique, les drones etc. Elle est équipée d'un processeur, de ports USB, d'un port Ethernet, d'un connecteur HDMI, d'un port pour carte microSD (servant de disque dur notamment) et de broches GPIO (General Purpose Input/Output).

Les broches GPIO de la Raspberry Pi permettent aux utilisateurs de connecter différents composants électroniques tels que des capteurs, des LED, ou tout autre composant électronique. Ces broches peuvent être programmées pour envoyer ou recevoir des signaux électriques, permettant ainsi aux utilisateurs de contrôler les composants électroniques connectés.

Concrètement dans notre cas les broches serviront dans la commande de pompes 12V et de LEDs, la Raspberry Pi peut être utilisée pour contrôler les relais ou les transistors qui permettent de faire passer l'électricité vers ces composants. En utilisant un logiciel de programmation, il est possible de contrôler les broches GPIO de la Raspberry Pi pour allumer ou éteindre les pompes et les LED et même de contrôler la vitesse de rotation de notre moteur.



---

*Liens : Le site officiel la Raspberry Pi ; La page Wikipédia de Raspberry Pi ; Source de l'image*

## ❖ Fontaines

Nous avons choisi une pompe 12V pour notre fontaine musicale, qui est normalement conçue pour un aquarium.

Cette pompe a un débit de 200L/H et est submersible, ce qui la rend facile à intégrer dans notre fontaine et facilite l'accès à l'eau pour nos fontaines puisqu'elles seront déjà immergées dans leur source, ainsi impossible de manquer d'eau et donc pas besoin de se raccorder à la tuyauterie.

De plus, le moteur DC utilisé dans la pompe nous permet de contrôler sa vitesse de rotation en utilisant la modulation de largeur d'impulsion (PWM), ce qui nous donne un meilleur contrôle sur le débit d'eau dans notre fontaine. Enfin, la pompe est silencieuse, ce qui est important pour notre projet de fontaine musicale afin de pouvoir correctement entendre la musique.



FIGURE 3 – Fontaine 12V DC utilisé

## ❖ Module L298N

Tout d'abord, le module L298N est un pont en H, cela signifie qu'il peut inverser le sens de rotation d'un moteur DC en changeant la polarité des broches d'entrée. Même si cela n'est pas utile à notre projet une autre particularité du module une autre fonctionnalité nous intéresse dans ce module. Premièrement, il va nous permettre d'isoler la raspberry pi du courant externe fourni aux moteurs. Mais également, il nous permet de moduler l'intensité et la fréquence qui est fourni au moteur ainsi on va pouvoir moduler la hauteur des jets d'eau.

De plus, le module L298N dispose de deux canaux de contrôle, ce qui nous permet de contrôler deux pompes avec un seul module. Cela est très pratique pour notre projet car nous avons besoin de contrôler plusieurs pompes pour créer différents effets de jet d'eau dans notre fontaine. Le module L298N est donc un choix parfait pour nous car il nous permet de contrôler plusieurs pompes à la fois.

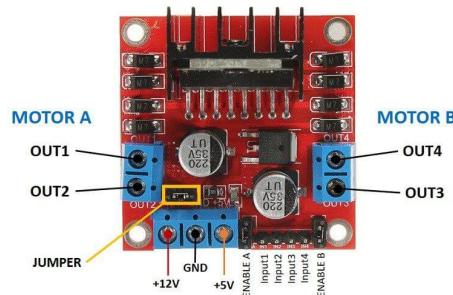


FIGURE 4 – Module L298N

## ❖ Module MCP23017

Le MCP23017 est un circuit intégré d'extension d'E/S qui permet de contrôler 16 broches d'E/S supplémentaires à partir de deux broches d'E/S de contrôle du microcontrôleur. Cela nous donne la possibilité de contrôler plusieurs pompes indépendantes avec un seul module. Ce dernier est utile dans notre projet car, le branchement du module L298N nécessite 8 ports GPIO ainsi pour 10 pompes il est impossible de pouvoir toutes les brancher à la Raspberry pi sans ports supplémentaires.

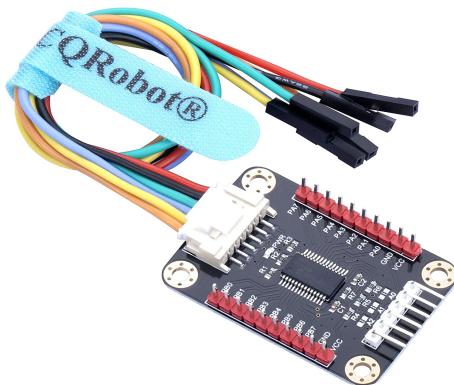


FIGURE 5 – Module L298N

## 2.2 Partie logiciel

### ❖ RPI.GPIO

La bibliothèque RPI.GPIO est une bibliothèque Python qui permet de contrôler les broches d'E/S de la carte Raspberry Pi. Elle est facile à utiliser et à programmer. Elle dispose de fonctions simples pour définir les broches d'E/S en entrée ou en sortie, et pour modifier leur état en fonction de notre programme. De plus, elle dispose de fonctions pour contrôler la vitesse de rotation de nos pompes en utilisant la modulation de largeur d'impulsion (PWM), ce qui nous permet de contrôler la vitesse de nos pompes.

### ❖ Sketchup

Sketchup est un logiciel de modélisation qui permet de créer des modèles en 3D. Ce qui nous a plu avec sketchup c'est qu'il permet de faire des conceptions de qualité tout en étant facile d'utilisation. Nous l'avons utilisé pour concevoir la fontaine. Dans un premier temps pour faire des modèles, nous permettre de modéliser différentes idées et voir le rendu. Puis pour créer notre modèle final. Sketchup permet une fois la conception faite de la convertir en format STL, qui permet par la suite d'être lu par une imprimante 3D.

## ❖ Librosa

Librosa est une bibliothèque Python open source utilisée pour l'analyse et la manipulation de fichiers audio. Elle permet d'extraire des caractéristiques audio comme les fréquences, les amplitudes, les temps de début et de fin des événements sonores, ainsi pour effectuer des transformations de Fourier rapides (FFT) et des transformations de cosinus discrètes (DCT) sur les signaux audio.

Nous avons utilisé Librosa pour faire danser l'eau au rythme de la musique en analysant le signal audio et en extraire des informations comme tempo, la fréquence fondamentale et des battements qui sont ensuite utilisées pour contrôler et animer la fontaine en synchronisation avec la musique.

## ❖ Matplotlib

Matplotlib est une bibliothèque Python qui nous permet d'afficher des graphiques et des animations qui peuvent être mis à jour au fil du temps, et elle nous permet également de créer les diagrammes dont nous avons besoin pour nos simulations.

Nous l'avons utilisé pour créer une visualisation animée de la fontaine et pour synchroniser le mouvement de l'eau avec la musique. Par exemple, la hauteur d'une courbe peut contrôler le débit d'une pompe. De plus, il permet un réglage facile de la taille, de la forme, de la fréquence des éléments graphiques pour correspondre au mieux au rythme de la musique. Nous avons plus particulièrement utilisé la fonctionnalité "FuncAnimation" de Matplotlib pour mettre à jour la visualisation dans le temps en fonction d'un signal audio en direct.

## ❖ Raspberry Pi OS

Raspberry Pi OS ou (Raspbian) est le système d'exploitation officiel basé sur Linux, créé spécifiquement pour les ordinateurs embarqués Raspberry Pi. Ainsi Il offre une interface utilisateur familière de type "bureau" avec le gestionnaire de fenêtres LXDE. De plus, Il inclut un grand nombre de logiciels pré-installés pour faciliter l'utilisation du Raspberry Pi, comme un éditeur de texte, un navigateur web, VLC.... Et Il prend en charge le langage de programmation Python dès l'installation.

Nous l'avons utilisé pour contrôler des électrovannes ou des pompes qui actionnent les jets d'eau en temps réel.

## ❖ Turtle

Turtle est une bibliothèque en Python qui nous permet de dessiner des formes géométriques, des motifs et des graphismes à l'écran de manière interactive en utilisant des fonctions fournies par cette bibliothèque. Une tortue est représentée par un curseur en forme de tortue qui peut se déplacer sur l'écran et laisser une trace derrière lui.

Nous avons utilisé la bibliothèque "turtle" en combinaison avec d'autres bibliothèques comme la bibliothèque "PyAudio" pour lire la musique, la bibliothèque "numpy" pour analyser le signal audio, et enfin "turtle" pour contrôler la simulation de Led en fonction de la musique.

## ❖ Wokwi

Wokwi est une plateforme en ligne qui permet aux utilisateurs de créer des modèles électroniques et de les simuler. Une raspberry pi est disponible sur ce site ce qui nous a permis de l'utiliser pour nos tests. Nous avons dans un premier temps testé sur une led, nous l'avons allumée et éteinte. Cela nous a permis d'en apprendre plus sur les branchements ainsi que sur le code à réaliser. PHOTO Nous avons ensuite découvert une matrice de led (dot matrix led). Nous avons créé un montage permettant de reproduire nos quatre mouvements grâce aux leds. Elles s'allumaient et s'éteignaient dans le temps recréant ainsi notre danse qui avait été modélisée.

## ❖ Cura

Cura est un logiciel pour l'impression 3D, il découpe le fichier déposé par l'utilisateur en couches et génère son G-Code spécifique à l'imprimante. L'interface de Cura permet de modifier un grand nombre de paramètres et donc d'avoir un grand contrôle sur l'impression. Il est compatible avec beaucoup d'imprimante 3D, et prend en charge différents formats comme le STL que nous utilisons.

# 3.Développement de La fontaine

## 1. Conceptualisation de la Fontaine

Dans le cadre de notre projet, nous avons choisi d'utiliser la modélisation 3D pour conceptualiser la fontaine, nous permettant ainsi de créer un modèle numérique qui nous permettra facilement de créer une reproduction physique de la fontaine avec une imprimante 3D.

Pour créer la modélisation 3D de la fontaine, nous avons utilisé le logiciel SketchUp. Nous avons créé des formes géométriques de base pour représenter les différents composants de la fontaine, tels que le bassin et le support, tout en respectant les contraintes liées à l'impression 3D.

Notre première idée de fontaine était de créer un bac de 80\*40\*16 qui serait le contenant de l'eau et de tout le mécanisme de la fontaine. Ce bassin avait une ouverture des mêmes dimensions qu'un tuyau pour l'entrée d'eau, et une autre ouverture (cette fois-ci en hauteur pour ne pas que l'eau s'évacue) qui nous permettait de faire passer tous les câblages. Nous voulions placer une grille au-dessus et mettre des pierres qui permettraient de cacher les installations. Grâce à sketchup nous avons donc créé ce bac ainsi que la grille.

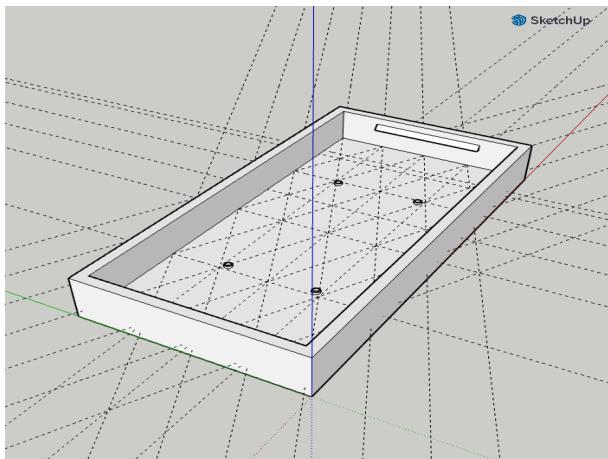


FIGURE 6 – Bac

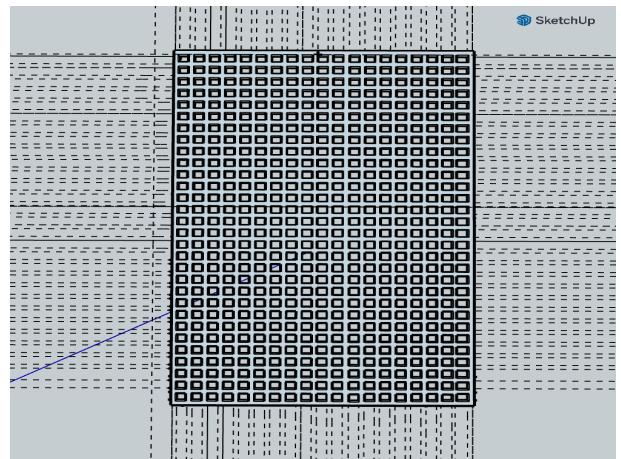


FIGURE 7 – Grillage

Après que cette conceptualisation soit réalisée, nous avons remarqué que cette idée ne faisait pas l'unanimité. Toutes les possibilités que nous offrait l'impression 3d, nous permettaient de faire des modèles plus extravagants. Nous sommes donc partis sur une nouvelle idée. Toujours avec le même bassin mais cette fois-ci en créant une structure surélevée qui recueillerait l'eau et la ferait descendre telle une cascade. Ce qui nous permettait de cacher tout le système de pompe de manière esthétique.

Il fallait une solution pour assembler le bassin et cette structure, nous avions opté pour des tiges qui se viseraient sur le bac. Il a fallu créer le filtre.

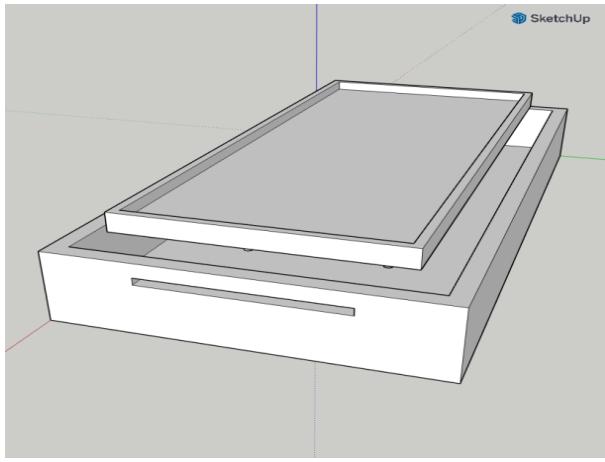


FIGURE 8 – Bassin avec support

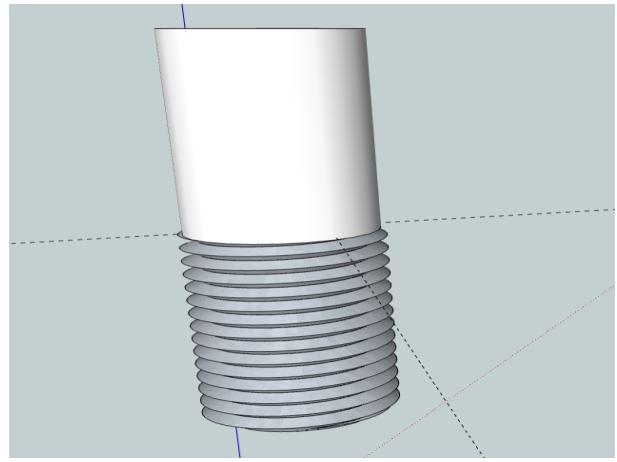


FIGURE 9 – Filtrage

Il restait qu'un problème à résoudre. L'imprimante ne pouvant imprimer des morceaux de plus de 20cm\*20cm\*20cm, nous devions trouver une manière de les assembler de sorte à ce que ce soit étanche. Nous avions pensé à une pâte ou colle à joints comme pour les piscines, qui nous permettrait de coller en garder l'étanchéité de notre structure.

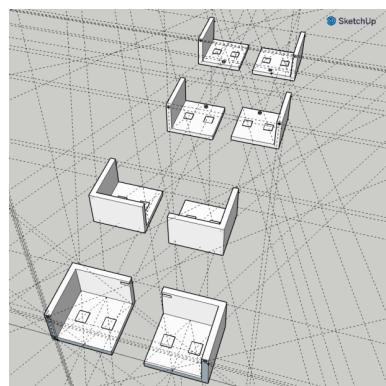


FIGURE 10 – Bac découpé en morceaux

Enfin, une fois la modélisation 3D est terminée, nous avons commencé à se renseigner sur l'imprimante 3D disponible à la faculté des Sciences. Nous avons consulté les ressources disponibles sur internet et discuté avec des personnes qui ont déjà utilisé cette imprimante afin de comprendre le bon fonctionnement de l'imprimante 3D.

Pour gérer tous les paramètres liés à l'impression, il fallait utiliser Cura. Après avoir sélectionné le modèle de l'imprimante. Nous devions configurer tous les paramètres, comme la qualité, la vitesse d'impression, les remplissages des surfaces, etc. Ces paramètres jouent un rôle très important, car par exemple la vitesse d'impression influe directement sur la qualité du produit, si elle est trop rapide cela peut créer des bavures ou des pertes de précisions. Tandis que si elle est trop lente, il peut y avoir des surchauffes. La qualité d'impression est aussi à prendre méticuleusement en compte, car pour un modèle nécessitant une grande précision, il faut mettre une qualité élevée, ce qui par contre augmente considérablement le temps d'impression. Le remplissage des surfaces permet de choisir la densité du remplissage du modèle ainsi que la structure des blocs qui composera les surfaces pleines de notre modèle, comme par des triangles, des lignes, des zigzags, etc. Cela joue un rôle dans la stabilité et la solidité du matériel.

Nous n'avions pas besoin d'une structure très précise, ni très solide, du coup nous avons paramétré en conséquence.

Une fois que tous les paramètres sont configurés, Cura génère automatiquement le G-Code qui sera envoyé à l'imprimante pour imprimer le modèle. Le G-Code est un langage de programmation qui contrôle les mouvements d'une imprimante 3D. Il est composé de commandes simples qui permettent à l'imprimante 3D de savoir quoi faire lors de l'impression.

Malgré nos efforts, nous n'avons pas réussi à obtenir une impression de notre modèle avec l'imprimante 3D, car elle était hors service, nous avons contacté le département de mécanique pour demander l'accès à leur imprimante, après discussion avec le responsable, nous avons appris que ce ne serait pas possible, car la façon d'impression des imprimantes de la faculté qui ne permettait pas d'avoir une structure étanche se rajouter à d'autres problèmes tel que le temps trop long nécessaire pour l'imprimer. Et de toute manière, il n'était pas possible d'avoir un bassin qui ne serait pas étanche.

Nous avons finalement renoncé à l'idée de l'impression 3D et décidé de garder notre premier modèle (figure 6) comme référence. Nous avons ensuite choisi une méthode plus rudimentaire pour fabriquer les composants de la fontaine, nous avons utilisé du ciment et le polystyrène pour créer une structure solide et légère pour la fontaine tout en utilisant notre modèle 3D comme référence afin d'obtenir des résultats similaires à notre modélisation.

Pour fabriquer le bassin (figure 11) de notre fontaine. Il fallait :

- du ciment
- deux blocs de polystyrène
- de la colle

Après avoir reproduit notre modèle avec le polystyrène et assemblé avec la colle à spéciale. Nous avons versé le ciment sur la structure et lissé toutes les surfaces de façon homogène. Il ne restait plus qu'à laisser sécher une demi journée.



FIGURE 11 – Bassin fabriqué à la main

## 2. Émulation du fonctionnement de la Fontaine

### 2.1 Modélisation de manière itérative

Nous avons tout d'abord représenté les mouvements de la danse en 2D en utilisant des boucles. Cela nous permettait de pouvoir visualiser les mouvements que pourrait avoir la fontaine. Nous avons travaillé en groupe de deux pour chaque mouvement et avons ainsi créé quatre mouvements qui forment une première danse. Nous avons utilisé la librairie Matplotlib pour afficher la visualisation de la danse sous forme d'un histogramme avec quinze barres représentant les valves et une hauteur maximale de 150 (représentant la hauteur maximal de l'eau). Les valeurs sont stockées dans un tableau de quinze cases et la hauteur varie de 10 en 10 en fonction du temps. Car nous au départ, nous voulions avoir quinze pompes d'où les quinze barres. L'objectif ici était de pouvoir visualiser concrètement l'objectif à terme.

Voici le code. .

### 2.2 Modélisation avec des fonctions

Ensuite, il a fallu remplacer les valeurs itératives par des fonctions. Nous avons donc trouvé des fonctions capables de reproduire les mêmes mouvements que ceux réalisés avec les valeurs initiales. Grâce à Géogebra notamment, nous avons pu trouver des fonctions qui se rapprochait de nos mouvements initiaux.

L'objectif ici était de réussir à comprendre comment nous pouvions utiliser des valeurs génériques afin de pouvoir rendre un aspect visuel harmonieux avec n'importe quel donnée en entrée. Car nous ne connaissons pas à l'avance les valeurs que nos traitements de la musique vont retourner. Voici le code.

### 2.3 Modélisation 3D

Une fois nos mouvements reproduits à l'aide des fonctions, notre objectif était de réaliser la même danse en 3D. Grâce à Matplotlib, nous avons pu réaliser la visualisation en 3D en utilisant une matrice de 15 par 15 avec une hauteur maximale de 150. Voici le code.

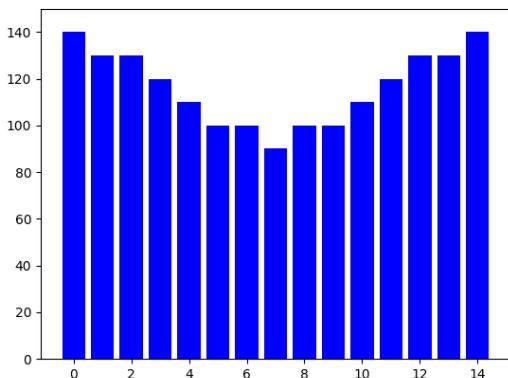


FIGURE 12 – Modélisation itérative 2D

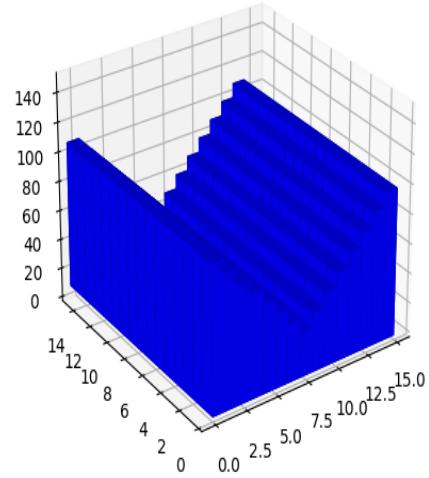


FIGURE 13 – Modelisation 3D

# Emulation de la partie LED

Afin d'émuler, nous avons modélisé la LED sous forme de cercle plein dont la couleur change. La couleur du cercle représente la couleur de la LED, ainsi pour faire cela, nous avons utilisé la bibliothèque turtle de python :

Code ci-dessous :

```
1 import turtle
2 import time
3 import random
4 import librosa
5 import numpy as np
6 import pygame
7 import matplotlib.pyplot as plt
8
9 def palette_couleur(nombre_couleurs):
10     """Create a palette of RGB colors ranging from warm to cool.
11     """
12     palette = []
13     for i in range(nombre_couleurs):
14         r, g, b = 0, 0, 0
15         if i < nombre_couleurs / 2:
16             r = int(255 * (i / (nombre_couleurs / 2)))
17             g = 0
18             b = 255 - r
19         else:
20             r = 255 - int(255 * ((i - nombre_couleurs / 2) / (
21                 nombre_couleurs / 2)))
22             g = int(255 * ((i - nombre_couleurs / 2) / (
23                 nombre_couleurs / 2)))
24             b = 0
25     palette.append((r, g, b))
26     return palette
27
28 def rgb_to_hex(rgb):
29     if len(rgb) != 3:
30         raise ValueError("Le triplet RGB doit contenir 3 valeurs")
31     if any(c < 0 or c > 255 for c in rgb):
32         raise ValueError("Les valeurs du triplet RGB doivent
33     tre comprises entre 0 et 255")
34
35     hex_code = "#{:02x}{:02x}{:02x}".format(*rgb)
36     return hex_code
37
38 window = turtle.Screen()
39 window.bgcolor("white")
40
41 t = turtle.Turtle()
42 t.shape("turtle")
43 t.speed(0)
44
45 audio_file = 'musique2.mp3'
46 y, sr = librosa.load(audio_file)
47
48 n_fft = 2048
49 hop_length = 512
50 frame_time = (3*60+3)/519
51
52 pitch_list = []
53 for i in range(0, len(y)-n_fft, hop_length):
```

```

50     frame = y[i:i+n_fft]
51     pitch, _ = librosa.piptrack(y=frame, sr=sr)
52     if pitch.any():
53         pitch_mean = librosa.pitch_tuning(pitch[pitch > 0])
54         pitch_mean = int((pitch_mean+1)*5)
55         pitch_list.append(pitch_mean)
56
57 color = palette_couleur(10)
58 tempo, beat_frames = librosa.beat.beat_track(y=y, sr=sr,
59     hop_length=1024, start_bpm=240)
60 tempo_frames = librosa.tempo_frequencies(len(beat_frames)-1,
61     hop_length=1024, sr=sr)
62
63 tab = [tempo_frames[i] for i in range(len(tempo_frames))]
64 tab[0] = 0
65
66 print(len(beat_frames))
67 print(color)
68
69 max_tempo = max(tab)
70 coef = 10/max_tempo
71
72 tab_echelle = [tab[i]*coef for i in range(len(tab))]
73
74 print(tab_echelle)
75
76 volume = np.abs(librosa.stft(y))
77
78 volume_int = volume.astype(int)
79
80 print(volume)
81
82 pygame.init()
83 pygame.mixer.music.load("musique.mp3")
84 pygame.mixer.music.play(-1)
85 for i in range(len(y)):
86     if(y[i]!=0):
87         t.color(rgb_to_hex("#FFFFF"))
88         t.begin_fill()
89         t.circle(y[i]*100)
90         t.end_fill()
91         time.sleep(1)
92         t.clear()

```

Listing 1 – Simulation LED

Le principe est le suivant, un tableau de couleur créé par la fonction : palette\_couleur (paramètre nombre de couleurs voulu) transformé sous format hexadécimal grâce à la fonction rgbtohex(rgb). Les couleurs prises par le cercle se font de manière assez simple, plus un le beat de la couleur est bas plus la couleur prise par le cercle sera une couleur froide et inversement plus le beat est haut plus la couleur prise par le cercle sera une couleur dite chaude.

### 3. Intégration de la partie musique

Dans le cadre de notre projet de fontaine musicale interactive, nous avons dû relever un défi important : celui de synchroniser les jets d'eau avec la musique.

Nous nous sommes dans un premier temps renseigné sur comment faire et quels paramètres pourraient nous servir. Après quelques recherches, nous avons découvert une bibliothèque en python, *librosa* qui permet d'analyser des audios. Il fallait ensuite penser à des paramètres qui feraient varier l'eau de manière harmonieuse. Nous avions vu que le tempo et le rythme étaient à prendre en compte.

Pour en savoir plus et approfondir nos connaissances sur la partie musique, nous avons contacté Monsieur Sylvain Daudé spécialiste de la musique. Suite à notre réunion avec lui, nous avons appris qu'il fallait détecter les pulsations par minutes ainsi que les sous pulsations ce qui permet d'identifier le tempo et le rythme d'une musique. Il fallait aussi prendre en compte la mélodie, donc récupérer un chromagramme qui est une matrice dont les lignes représentent les hauteurs des notes de musique et les colonnes les moments dans le son. Le timbre moins intéressant à analyser mais qui pourrait capturer les différentes fréquences dans la musique (les graves, les médiums et les aigus).

Ayant connaissance de cela, nous avons tout d'abord exploré l'utilisation du tempo (la vitesse à laquelle le musicien va jouer la musique) pour synchroniser les jets d'eau avec la musique, mais nous avons rapidement réalisé que cette méthode ne fonctionnait pas pour toutes les chansons. Lorsqu'il s'agit de détecter les débuts des notes dans une musique, plusieurs facteurs peuvent rendre la tâche plus difficile. Par exemple, certains instruments, tels que le violon ou la guitare, peuvent produire des notes avec des changements minimes d'amplitude, ce qui rend difficile la détection des débuts de notes en utilisant l'enveloppe d'amplitude comme unique indicateur. De plus, certains styles de musique, comme le jazz ou le blues, peuvent utiliser des notes liées ou une improvisation libre, ce qui rend difficile la détection des débuts de notes en utilisant une méthode basée sur la durée ou l'intensité des notes.

Après avoir exploré l'utilisation du tempo pour synchroniser les jets d'eau avec la musique, nous avons choisi les fonctions de nouveauté comme outil pour détecter les débuts de notes dans la musique. Les fonctions de nouveauté sont des fonctions qui mesurent les changements locaux dans les propriétés du signal telles que l'énergie ou le contenu spectral. Notre idée était de choisir le contenu spectral, car ce dernier peut détecter les moments où les caractéristiques spectrales du son changent, par exemple lorsqu'une note change de hauteur ou lorsqu'un nouvel instrument commence à jouer.

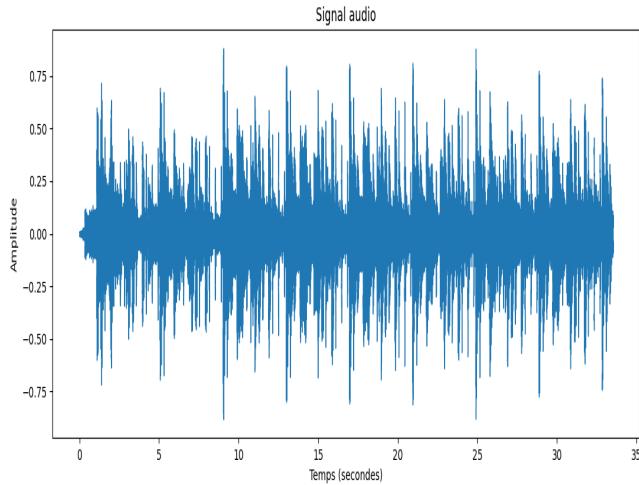


FIGURE 14 – signal audio

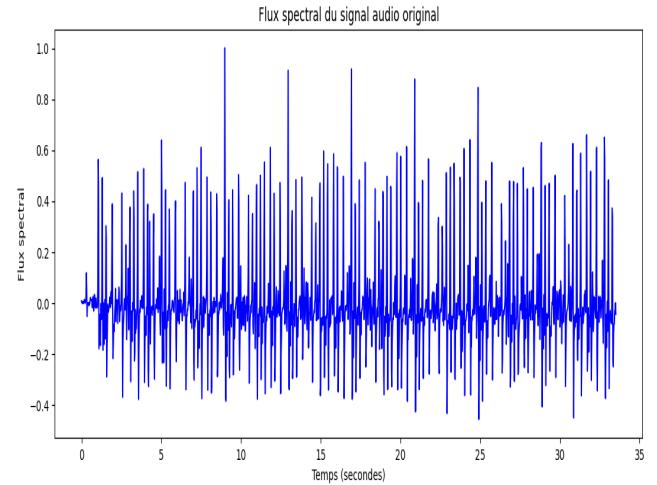


FIGURE 15 – flux spectral

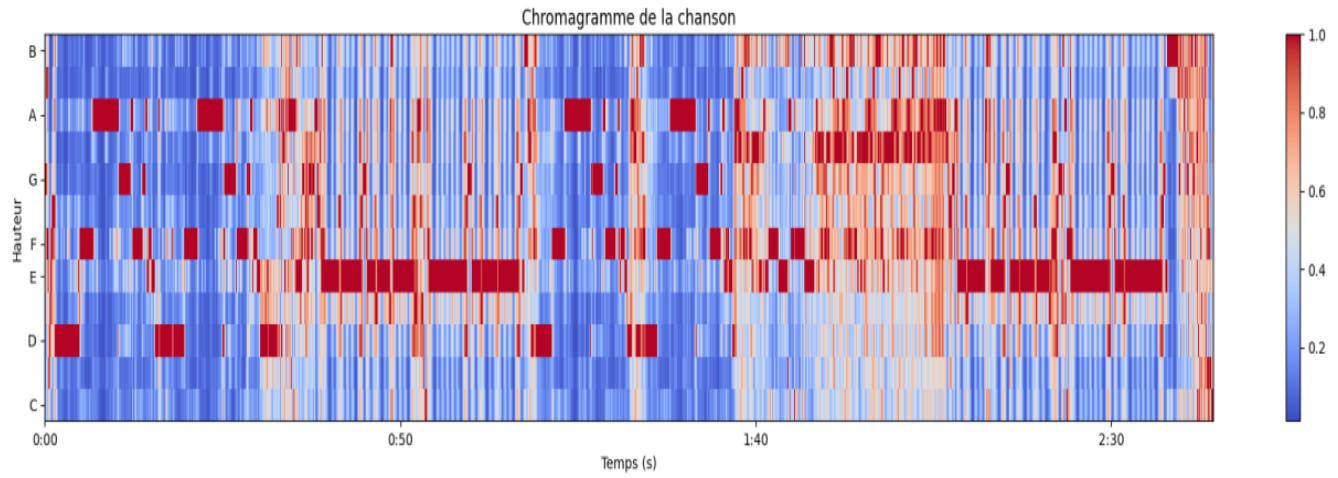


FIGURE 16 – Chromagramme de la chanson

Prenons l'exemple d'un pianiste jouant une mélodie. En utilisant une fonction de nouveauté basée sur l'énergie pour détecter les débuts de notes, il y a une grande possibilité qu'elle détecte plusieurs débuts de notes pour une seule note jouée au piano, et cela est dû à la variance naturelle de la puissance sonore du piano pendant que le pianiste joue une note.

En revanche, si l'on utilise une fonction de nouveauté basée sur le contenu spectral, elle pourra détecter les moments où le pianiste change de note, même si la puissance sonore du piano reste relativement constante. Cela est dû à l'unicité des signatures spectrales des notes de piano. La fonction de nouveauté basée sur le contenu spectral peut donc détecter avec plus de précision les changements de note et éviter les fausses détections de début de note causées par des fluctuations d'énergie.

Finalement, notre choix était sur la méthode de détection des impulsions à l'aide du contenu spectral pour de nombreuses raisons, notamment la détection des changements de fréquence dans la musique afin de les utiliser pour synchroniser les jets d'eau avec une meilleure précision.

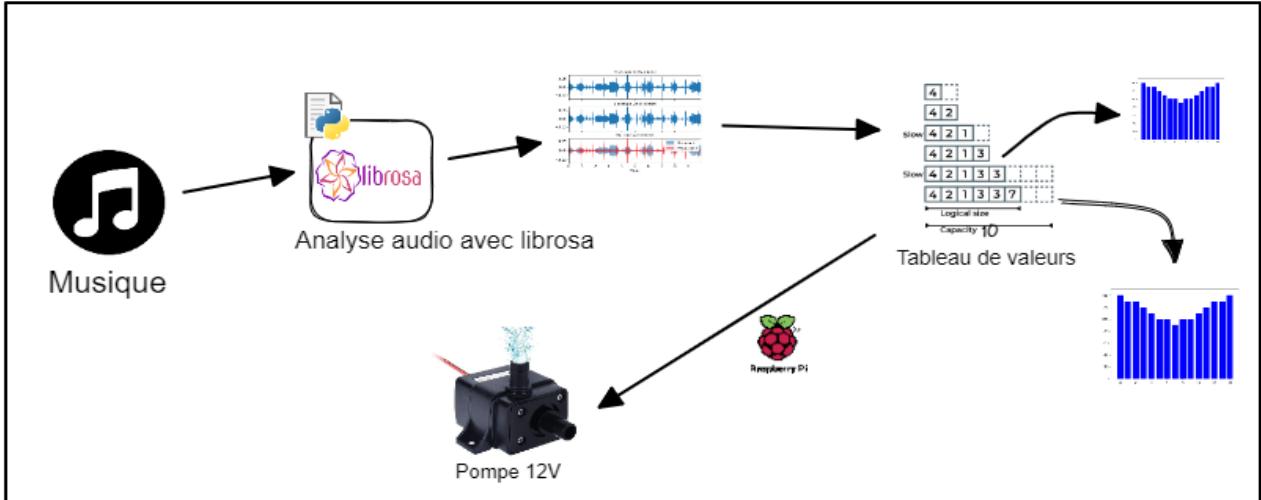


FIGURE 17 – schéma fonctionnel

## 4. Interface graphique

On a implémenté une interface graphique pour créer des danses manuellement. L'objectif était de simuler le mouvement de l'eau dans notre fontaine en ajustant les hauteurs des jets d'eau à l'aide de curseurs interactifs.

Notre interface contient une grille de curseurs qui représentent les différentes pompes de la fontaine. La hauteur correspondante à chaque jet d'eau est réglée par le curseur, et après chaque étape, les valeurs seront enregistrées dans un tableau.

Le bouton "Passer à la prochaine étape" permet de procéder à la simulation afin que les valeurs des hauteurs des jets d'eau seront enregistrées dans une liste de tableaux. Ensuite, toutes les hauteurs seront réinitialisées pour la prochaine étape.

Le bouton "Passer à la prochaine étape" permet de visualiser les résultats de la simulation. Un graphique animé avec des barres représentant l'évolution des hauteurs des jets d'eau au fil du temps s'affiche dans une nouvelle fenêtre.

Enfin, pour enregistrer les résultats de simulation, on peut tout simplement cliquer sur "Télécharger les données".



FIGURE 18 – Interface graphique

# 4.Algorithmes et Structures de Données

Pour détecter les notes jouées dans une chanson, notre algorithme utilise plusieurs méthodes de la bibliothèque Librosa.

Tout d'abord, la fonction onset.strength de Librosa est utilisée pour calculer la fonction de détection des impulsions à l'aide du contenu spectral, qui mesure la salience des événements sonores à chaque instant dans le signal audio. Ensuite, la fonction onset.detect de Librosa est appliquée sur cette fonction de détection d'attaque pour détecter les onsets, c'est-à-dire les instants précis où un événement sonore a lieu dans la chanson.

Pour identifier les notes jouées, le chromagramme est calculé à partir du signal audio à l'aide de la fonction chroma.stft de Librosa. Le chromagramme est une représentation de la répartition spectrale des notes dans un signal audio, en fonction du temps. En d'autres termes, il représente l'intensité de chaque note présente dans le signal audio à un instant donné.

Pour faciliter l'analyse musicale et permettre une détection plus précise des notes jouées, le chromagramme est normalisé à l'aide de la fonction normalize de Librosa. Cette normalisation permet d'ajuster la distribution des notes de manière à ce qu'elles soient plus facilement comparables et identifiables. Ainsi, la représentation de chaque note devient comparable à celle des autres notes, indépendamment des variations de volume ou de dynamique.

Enfin, la méthode frames.to.time de Librosa est utilisée pour convertir les onsets détectés en temps réel, ce qui permet de synchroniser l'affichage des notes jouées avec la chanson.

```
1 import librosa
2 import numpy as np
3 from matplotlib import pyplot as plt, animation
4 import pygame
5 from tkinter import Tk, Button
6 from tkinter.filedialog import askopenfilename
7
8
9 # Initialiser pygame
10 pygame.mixer.init()
11
12 # Charger le fichier audio
13 root = Tk()
14 root.withdraw()
15 audio_path = askopenfilename()
16 y, sr = librosa.load(audio_path)
17
18 # Charger la chanson avec pygame
19 pygame.mixer.music.load(audio_path)
20
21
22
23
24 # Dectecter les onsets dans la chanson
25 onset_frames = librosa.onset.onset_detect(y=y, sr=sr)
26
27 # Nombre d'onsets dans la chanson
28 n_onsets = len(onset_frames)
29
```

```

30 # Calcul du chromagramme
31 chroma = librosa.feature.chroma_stft(y=y, sr=sr, n_fft=2048,
32                                         hop_length=512)
33
34 # Selection des temps pertinents
35 temps = librosa.frames_to_time(onset_frames, sr=sr, hop_length
36                                 =512)
37
38 # Selection des colonnes du chromagramme correspondant aux temps
39 # pertinents
40 chroma = chroma[:, onsetframes]
41
42 # Normalisation du chromagramme
43 chromanorm = librosa.util.normalize(chroma, norm=2, axis=0)
44
45 # Initialiser la liste de hauteurs avec les valeurs du
46 # chromagramme
47 hauteurs = [[] for _ in range(n_onsets)]
48 for i in range(n_onsets):
49     for j in range(12):
50         hauteur = chromanorm[j, i]
51         hauteurs[i].append(hauteur)
52
53 fig, ax = plt.subplots()
54 plt.ylim(0, 1)
55 ax.set_ylim(0, 1)
56 plt.xlim(0, 13)
57
58 bar_collection = ax.bar(range(1, 13), [0]*12, align='center',
59                         linewidth=0.5)
60
61 # La duree d'une frame en millisecondes est gale a la moyenne
62 # des intervalles de temps entre chaque onset multipliee par
63 # 1000 pour avoir des millisecondes
64 frame_duration = np.mean(np.diff(temps)) * 1000
65
66 def animate(i):
67     for j, b in enumerate(bar_collection):
68         b.set_height(hauteurs[i][j])
69     return bar_collection
70
71 # Fonction pour lancer la chanson avec pygame
72 def play_music():
73     pygame.mixer.music.play()
74 play_music()
75
76 # D finir les parametres de l'animation
77 anim = animation.FuncAnimation(fig, animate, frames=n_onsets,
78                               interval=frame_duration, blit=True)
79
80 anim.repeat = False
81 plt.show()

```

Listing 2 – detecte notes

# 6. Expérimentation réel et analyse des résultats

## 6.1 Première expérimentation réel

Par faute de matériel, nous avons décidé de commencer par contrôler une LED, nous avons branché la LED à un breadboard (qui permet de tester des circuits électroniques sans avoir à les souder) et relié au port du gpio de la raspberry. Nous avons créé notre premier programme en Python qui permet d'éteindre et d'allumer la LED, de plus nous pouvions contrôler le temps où elle restait allumé ou éteinte.

En attendant l'achat du matériel par notre encadrant, nous avons décidé d'acheter une première pompe afin de pouvoir tester de notre côté et de se familiariser avec le matériel. Après plusieurs essais, on s'est rendu compte que la pompe seule avec la raspberry pi ne fonctionnait pas. Le problème était que les ports GPIO 3,3V ou même le port de sortie 5V ne permettait pas un apport suffisant en tension électrique, il nous fallait une source plus puissante. Pour résoudre ce problème, nous avons trouvé une pile 9V, ce qui n'était toujours pas suffisant, mais qui nous permettait de contrôler un moteur DC. On a réussi à contrôler les intensités du moteur en le faisant tourner à peu près rapidement.

On a réussi à contrôler les intensités du moteur en le faisant tourner plus ou moins rapidement.

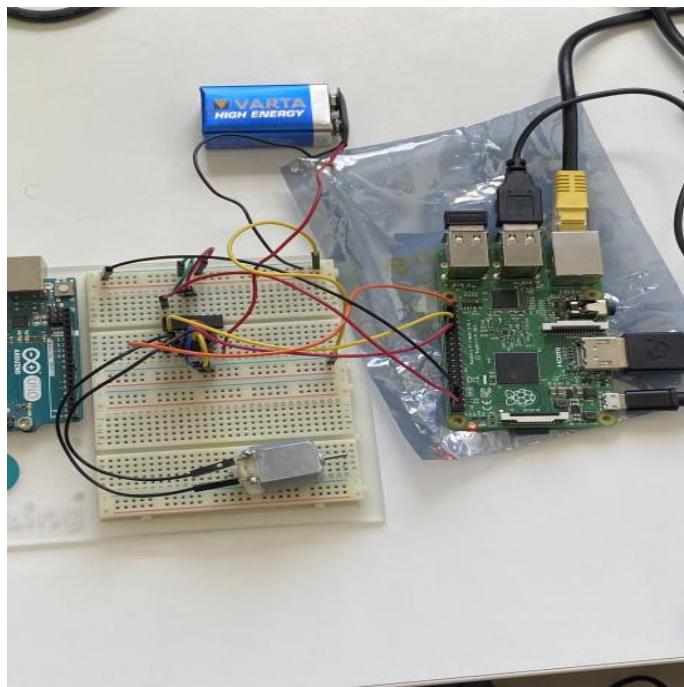


FIGURE 19 – Moteur DC

## 6.2 Deuxième expérimentation réel

### Branchement de la pompe

Afin de comprendre le branchement que nous avons effectué, voici un schéma des ports de la raspberry pi :

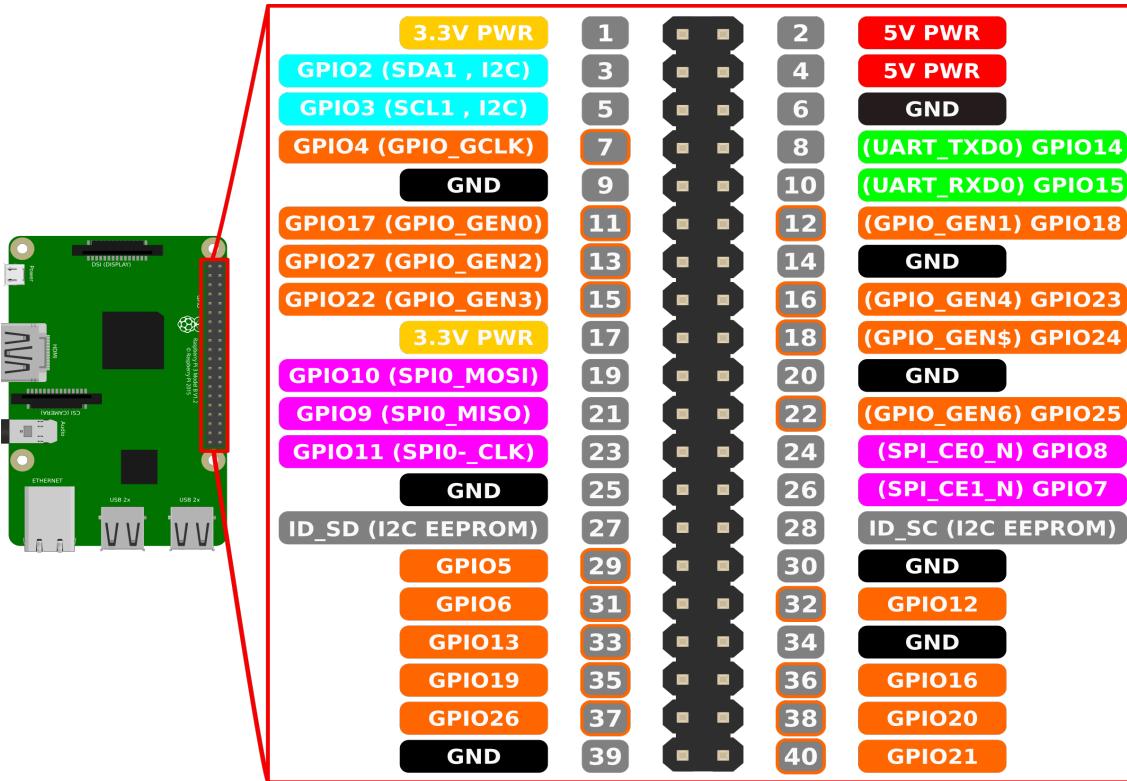


FIGURE 20 – broches GPIO

Ici les ports GND nous serviront de tension, de plus dans notre cas peu importe le port GPIO que nous allons utiliser ils fonctionnent tous pour moduler l'intensité du courant envoyé aux pompes. Ci-dessous le schéma de branchement pour lequel nous avons opté :

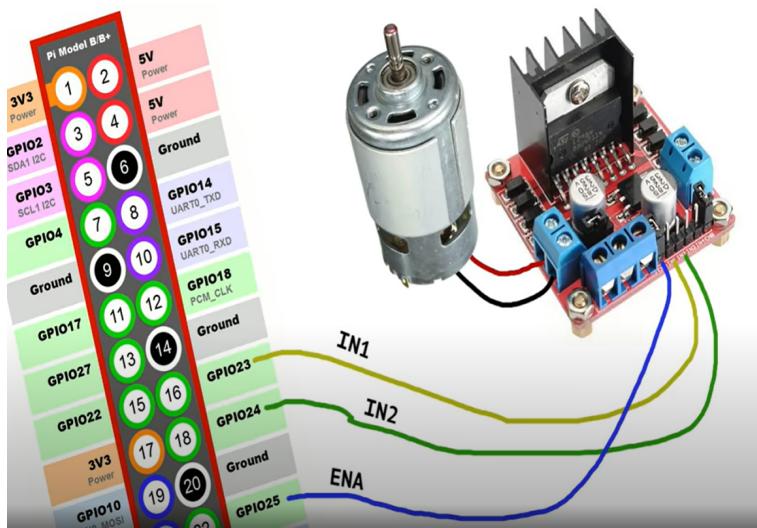


FIGURE 21 – Schéma de branchement

Ici, nous branchons le moteur aux deux broches OUT1 et OUT2 du module L298N. De plus, notre source d'alimentation (dans notre cas un chargeur 12V) a été branchée sur les ports 12V et GND du même module. Cela nous permet d'alimenter le moteur, d'isoler la Raspberry Pi, mais également de pouvoir contrôler la puissance fournie aux moteurs à travers les broches IN1, IN2 et ENA du module qui sont respectivement branchées sur les ports GPIO 23,24 et 25.

## Premier contrôle avec L298N statique

La première utilisation était de simplement contrôler l'intensité de la pompe de manière statique, d'où le code ci-dessus. Cela nous a permis de comprendre comment le moteur s'allume et comment moduler la vitesse du moteur à l'aide des ports GPIO.

## Deuxième contrôle avec L298N (expérimentation avec les beats)

Nous avons utilisé le retour de notre fonction de traitement de la musique (donc un tableau) dans la raspberry pi afin de pouvoir avoir une première visualisation des mouvements de la pompe contrôlé par les beats de la musique

Code ci-dessous :

```
1 import RPi.GPIO as GPIO
2 from time import sleep
3 import math
4 import tab
5 import pygame
6
7 in1 = 24
8 in2 = 23
9 en = 25
10 temp1=1
11
12 GPIO.setmode(GPIO.BCM)
13 GPIO.setwarnings(False)
14
15 GPIO.setup(in1,GPIO.OUT)
16 GPIO.setup(in2,GPIO.OUT)
17 GPIO.setup(en,GPIO.OUT)
18 GPIO.output(in1,GPIO.LOW)
19 GPIO.output(in2,GPIO.LOW)
20 p=GPIO.PWM(en,1500)
21
22 p.start(100)
23
24
25 pygame.init()
26 pygame.mixer.music.load("musique2.mp3")
27 pygame.mixer.music.play()
28
29 for i in range(len(tab.tab_echelle)):
30     p.start(tab.tab_echelle[i])
31     GPIO.output(in1,GPIO.LOW)
32     GPIO.output(in2,GPIO.HIGH.i)
33     sleep(tab.trame_duration)
```

Listing 3 – code raspberry

Ici, la méthode 'pwm()' permet de moduler la vitesse de rotation du moteur, donc la hauteur de l'eau, par principe étant donné que les valeurs de retour de notre algorithme de traitement de la musique échelonne notre tableau entre des valeurs allant de 0 à 100. Ce sont ces valeurs qui vont être prises par notre moteur. Un 100 indique une vitesse maximale et un 0 indique un moteur éteint. La méthode output indique si le GPIO doit fournir une tension électrique ou pas (d'où le paramètre GPIO.LOW et GPIO.HIGH)

Vidéo du premier test

### Troisième contrôle avec L298N (expérimentation avec les onsets)

Comme dit plus haut le meilleur paramètre à utiliser pour faire danser l'eau est l'onset. Ainsi, dans cette troisième expérimentation seul les valeurs du tableau change.

## 6.3 Analyse des résultats

Finalement, le traitement de la musique nous a permis de faire danser la simulation de la pompe en 2D, mais également la pompe que nous avions achetée de notre côté en attendant le matériel. Notre analyse est la suivante, la musique contient énormément de paramètres qui peuvent être expérimentés. De plus, avec nos recherches et expérimentation, nous nous sommes rendus compte d'un point important, chaque musique est unique et les paramètres exploités dépendent donc de la musique. Malgré tout, le paramètre le plus important et le plus satisfaisant reste l'onset. Celui-ci nous a permis des résultats très harmonieux avec la majorité des musiques utilisées et encore plus lorsque l'on utilise des fonctions mathématiques telles que les fonctions cosinusoïdales qui permettent d'harmoniser et d'organiser les valeurs rentrées par notre algorithme de traitement de la musique.

Par manque de matériel, ce sont les seules conclusions que nous ayons pu tirer de notre projet.

# 7.Gestion du projet

## Outils du travail en collaboration

**Trello** : est une application de gestion de projet en ligne qui permet de créer des tableaux de tâches, de les organiser en listes et de suivre leur progression.

**Overleaf** : un éditeur LaTeX en ligne permettant de travailler sur les documents de manière collaborative et en temps réel. Nous l'avons utilisé pour la rédaction des documents hebdomadaires (compte-rendu, bilan) ainsi que le rapport.

**Google Drive** : un service de stockage et de partage de fichiers dans le cloud lancé par la société Google. Nous l'avons utilisé pour le partage des documents ainsi que les vidéos de démonstration

## Répartition du travail dans le temps



FIGURE 22 – Diagramme de Gantt

## 8. Conclusion

Le projet que nous avons choisi est un projet multidisciplinaire, qui comprend une partie informatique, une partie systèmes embarqués et une partie conception manuelle. L'idée de découvrir de nouveaux domaines nous a particulièrement attiré. Ce projet représentait un véritable challenge pour nous étant donné que les systèmes embarqués et la musique sont un domaine qu'aucun membre du groupe ne maîtrise à la base.

Ainsi, ce projet nous a permis de découvrir à la fois le lien étroit entre l'informatique et les systèmes embarqués, mais également les différences entre ces domaines à part entière qui sont très différents.

Nous avons donc, malgré nous, dû consacrer une grande majorité de notre temps à comprendre le fonctionnement de tous les composants que nous avons utilisés. C'est-à-dire dans un premier temps comprendre le fonctionnement global de la Raspberry Pi, installer l'OS nécessaire dans la Raspberry pi, comprendre comment les broches fonctionnent et comment les manipuler, même choses pour tous les modules externes utilisés.

À cause de cela, même simplement l'allumage d'un moteur était un véritable mystère. Et par exemple, détecter que cela ne fonctionnait pas à cause du voltage insuffisant à était très compliqué. Nous avons dû tout apprendre de A à Z. Nous voulions avancer plus rapidement, mais comme c'était un domaine assez nouveau, cela était compliqué. De plus, certaines péripéties comme avec l'imprimante nous ont fait perdre un temps considérable. Mais malgré tout cela, ce projet nous a appris énormément de choses et nous a beaucoup plu.

Nous avons découvert un nouvel univers de passionnés que nous ne soupçonnions pas. Nous avons contacté quelques adeptes des fontaines et sommes entrés dans un groupe discord spécialisé sur ce domaine.

Cette expérience fut très enrichissante.

## 9.Bibliographie

1. <https://alain-michel.canopprof.fr/eleve/tutoriels/raspberry>.
2. <https://colab.research.google.com/github/stevetjoa>.
3. <https://www.researchgate.net/publication>.
4. <https://www.raspberrypi-france.fr/guide/>.
5. <https://passionelectronique.fr/tutoriel-l298n>.
6. <https://www.researchgate.net/publication>.
7. <https://www.youtube.com/watch?v=2bganVdLg>.
8. <https://www.youtube.com/watch?v=0kOXgIxvdEU>.

# 10. Annexe

```
1 import matplotlib
2 import matplotlib.pyplot as plt
3 import numpy as np
4 def f1(t,tab):
5     current_tab = tab
6     for i in range(len(tab)):
7         tab[i]=-10*i+(150-t*10)
8         if tab[i]<0:
9             tab[i]=0
10    return current_tab
11 def f2(t,tab):
12    current_tab = tab
13    for i in range(len(tab)):
14        tab[i]=10*i+(-150+t*10)
15    return current_tab
16 def f3(t,tab):
17    current_tab = tab
18    for i in range(len(tab)):
19        tab[i]=abs(10*i-70)-70+t*10
20    return current_tab
21 def mouvement(function,init_y):
22    for i in range(0,16):
23        plt.clf()
24        print(function(i,init_y))
25        plt.bar(range(15),init_y,color="blue")
26        plt.ylim(0,150)
27        plt.pause(0.5)
28    return init_y
29
30 init_y = [0 for i in range(15)]
31 mouvement(f1,init_y)
32 mouvement(f2,init_y)
33 mouvement(f2bis,init_y)
34 mouvement(f3,init_y)
35 plt.show()
```

Listing 4 – Modélisation de manière itérative

. Revenir.

```
1 def f1(t,tab):
2     current_tab = tab
3     for i in range(len(tab)):
4         tab[i]=-10*i+(150-t*10)
5         if tab[i]<0:
6             tab[i]=0
7     return current_tab
8
9 def f2(t,tab):
10    current_tab = tab
11    for i in range(len(tab)):
12        tab[i]=10*i+(-150+t*10)
13    return current_tab
```

```

14
15 def f2bis(t,tab):
16     current_tab = tab
17     for i in range(len(tab)):
18         tab[i]=10*i-(t*10)
19     return current_tab

```

Listing 5 – Modelisation avec des fonctions

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3
4
5 def f(t, tab):
6     return [round((1.4285 + 2.857(3 - t)) * abs(i - 7) + (10 * t
7     + 80)) for i in range(len(tab))]
8
9 # Generation de donnees aleatoires pour l'histogramme
10 data = np.random.rand(15)
11
12 # Creation du figure et du plan 3D
13 fig = plt.figure()
14 ax = fig.add_subplot(111, projection='3d')
15
16 # Param tres pour la position des barres
17 x_pos = np.arange(len(data))
18 y_pos = np.zeros(len(data))
19 z_pos = np.zeros(len(data))
20
21 # Param tres pour les dimensions des barres
22 dx = np.ones(len(data))
23 dy = np.ones(len(data))
24 dz = data
25
26 z = [0 for i in range(15)]
27 x = [i for i in range(15)]
28 y = [1 for i in range(15)]
29 z = f(1, z)
30
31 # Creation des barres
32 ax.bar3d(x_pos, y_pos, z_pos, dx, dy, z)
33 ax.set_ylim(0, 15)
34
35 # Affichage de l'histogramme
36 plt.show()

```

Listing 6 – Modelisation 3D

Revenir.