

École Polytechnique de l’Université de Tours  
64, Avenue Jean Portalis  
37200 TOURS, France  
Tél. +33 (0)2 47 36 14 14  
[www.polytech.univ-tours.fr](http://www.polytech.univ-tours.fr)

**Manuel d’installation**

**PROJET : DESCRIPTION AUDIO POUR MAQUETTE OU ŒUVRE D’EXPOSITION**

Table des matières

[Introduction 4](#_Toc32837888)

[Contexte 4](#_Toc32837889)

[Objectif 4](#_Toc32837890)

[Problématique 4](#_Toc32837891)

[Objectifs 4](#_Toc32837892)

[Existant 4](#_Toc32837893)

[Livrables 4](#_Toc32837894)

[Planification 4](#_Toc32837895)

[Solution apporté 4](#_Toc32837896)

[Bilan 4](#_Toc32837897)

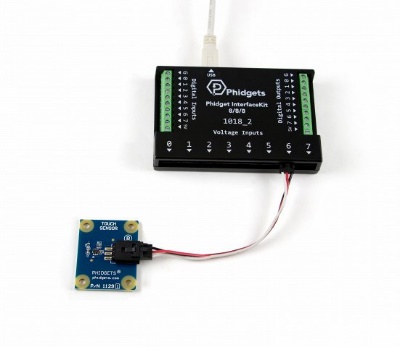
[Bibliographie 4](#_Toc32837898)

# Introduction

# Le projet

## Existant

Le projet a connu un existant sous ordinateur avec un OS windows. Pour qu’il puisse fonctionner, un module Phidget 8/8/8 qui est couplé avec des capteurs capacitifs tout ou rien.



## Objectif

L’objectif de ce projet consiste principalement à miniaturiser le système existant. Pour se faire, une analyse sur deux architectures différentes devra être réaliser pour déterminer, c’est-à-dire choisir entre MCU ou SBC (Single Board Computer).

Pour résumé, il faut :

* Miniaturiser pour tenir dans un carré de 20x20cm (choisir une architecture adaptée)
* Détecter un appui long/court
* Lire un fichier audio
* Prévoir une solution pour changer les fichier audios

## Livrables

Ce projet a pour but d’obtenir un système autonome pouvant se greffer sur une statue ou une maquette. En fonction d’interaction de toucher, elle effectuera une ou plusieurs actions de sortie. Ces actions de sorties seront des informations l’utilisateur par le biais de messages audios.

Un existant sous ordinateur windows existe déjà, il faut étudier la possibilité d’effectuer un portage sur une autre cible. Si un portage doit être effectuer sur un microcontrôleur, il faudra réaliser l’analyse et la conception du PCB.

### Prévue

Le système au niveau matériel doit :

* Détecter le toucher d’une main sur la paroi de l’œuvre
* Gérer un signal audio de sortie
* Concevoir des empreintes/supports physiques
* Concevoir un boitier pour contenir le système

Le système au niveau logiciel doit être :

* Différencier un appuie long et court
* Offrir la possibilité de mettre à jour les pistes audios
* Gérer la lecture de piste audio

Si le choix d’un microcontrôleur est retenu, un PCB sera réalisé.

### Disponible

Au terme de projet, les livrables seront tous disponible sur un dépôt GitHub ainsi que toutes les documentations en plus d’un clone de l’OS utilisé.

Le git comprend :

* Le fichier projet de l’application Phidget
* Le fichier projet de l’application de mise à jour des fichiers audio
* Le clone de l’OS utilisé
* Le cahier de spécification
* Le cahier d’analyse
* Le rapport
* Le manuel d’utilisateur pour le mainteneur
* Le manuel d’installation

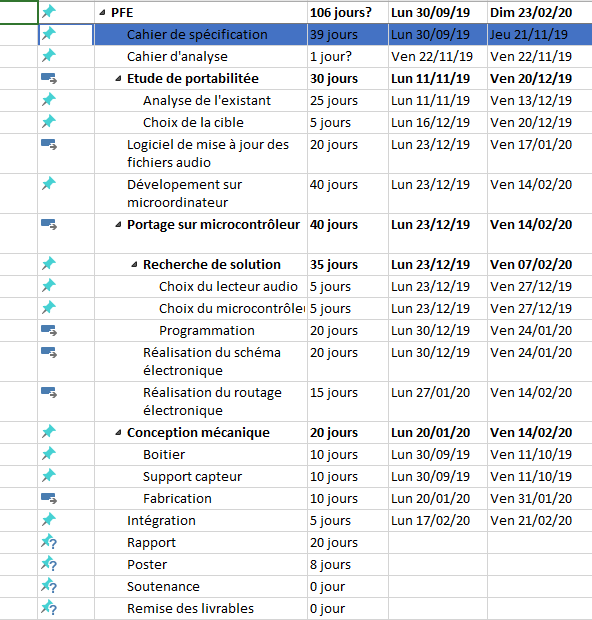
Dépôt GitHub : <https://github.com/VictorBeaulieu/PFE_livrable> )

# Planification

## Prévisionnel

Une image contenant capture d’écran

Description générée automatiquement



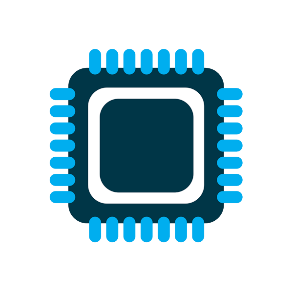
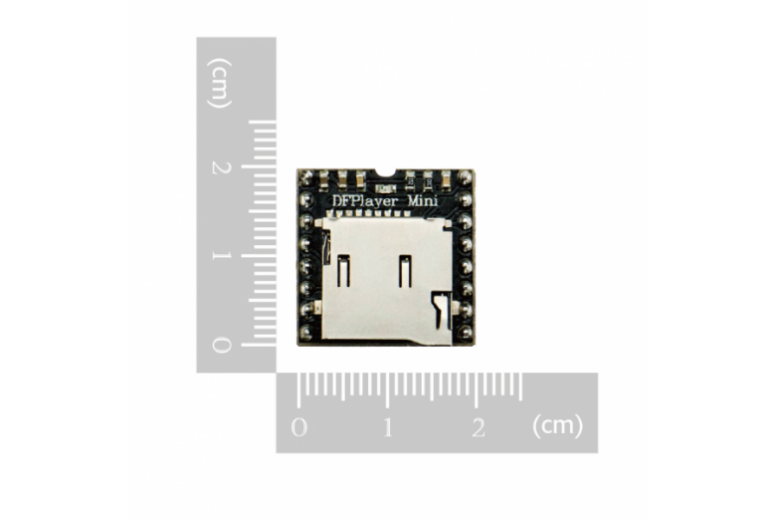
# Solution apporté

Durant ce projet, 2 solutions ont été envisagé. Soit il fallait effectuer le portage sur un microcontrôleur, reprendre donc tout de A à Z, soit récupéré ce qui pouvait l’être et effectuer le portage sur un SBC.

## Première phase

La première solution consiste à tout déporter sur un microcontroleur

### Solution n°1



Module lecteur audio

Capteur de toucher

Microcontrôleur

#### avantages

* Plus faible consommation
* Architecture sur mesure
* Temps de conception plus long

#### Inconvéniants

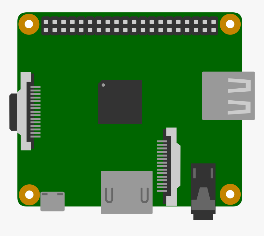
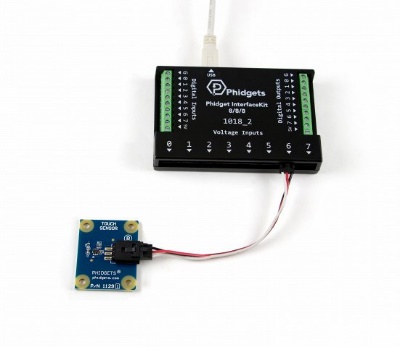
* Temps de développement plus long
* Fabrication d’un PCB
* Fige l’architecture et restreint l’ajout de nouvelles fonctionnalités

### Solution n°2

#### Principe

Cette solution consiste à un SBC

Module Phidget 8/8/8  
+  
Capteur capacitif



SBC

#### avantages

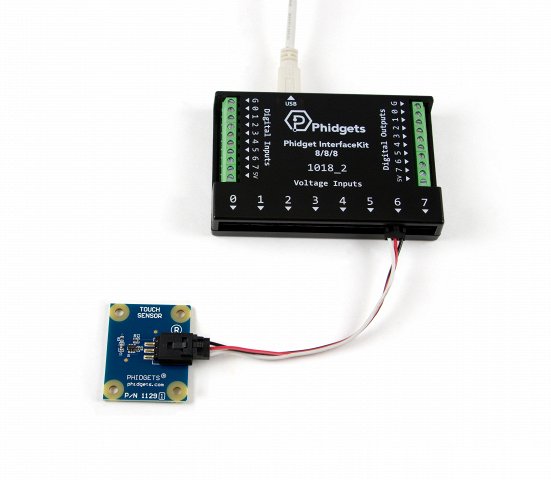
* Possibilité de réutiliser le capteur de toucher existant
* Temps de programmation plus rapide grâce à l’API du module Phidget
* Possibilité d’ajouter des fonctionnalités

#### Inconvéniants

* Consommation plus élevée face à un microcontrôleur

## Deuxième phase

Au terme de la première phase, la solution retenue avec l’encadrant fut la solution n°2.



Elle offre les avantages suivants :

• Possibilité de réutiliser le capteur de toucher existant

• Temps de programmation plus rapide grâce à l’API du module Phidget

• Possibilité d’ajouter des fonctionnalités

## Développement

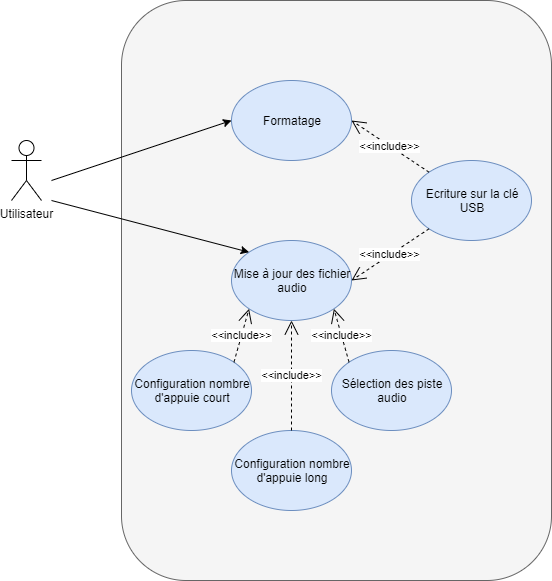
Pour le développement, j’ai utilisé le langage de programmation C# avec le framework DotNet Core.3 En effet, les capteurs disposent d’une API en C# et le DotNet Core 3 est un framework du C# le rendant multi plate-forme (Linux, Windows, Max OS, linux arm, etc…).

L’avantage de cette solution, c’est qu’il est possible de développer sur un ordinateur Windows, puis de déposé le logiciel sur la cible. Cette dernière n’a besoin d’aucun paquet particulier à installer, le logiciel est exécutable seul.

## Mise à jour des fichiers audios

Une fois l’architecture retenue, il était possible de réfléchir à la mise à jour des fichiers audio. La solution retenue fut de créer un logiciel qui à pour but de copier les fichiers audios sur un espace de stockage type clé USB. Cette clé USB sera lue au démarrage du système, et copiera les nouveaux fichiers audios. Ce support de stockage comportera des éléments permettant de configurer le nombre d’appui long, le nombre d’appui court.

Le diagramme de cas d’utilisation ci-dessous représente les fonctions majeures que dois faire le programme. :



Voici un exemple d’interface :



# Fonctionnalités envisageables

Au terme de ce PFE, des idées d’amélioration sont apparue. Puisque la solution SBC fut retenue, il est possible de choir un SBC disposant d’une sortie vidéo afin de lire des clips vidéo sur un écran.

Une autre idée consiste à faire communiquer le système actuel avec un driver de LED haute puissance (projet 4A SmartSystem) pour avoir des animations sonore et lumineuse.

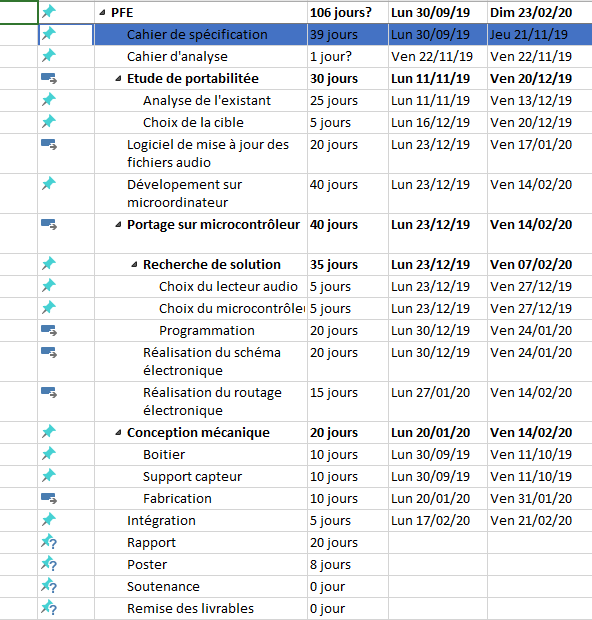
Enfin, la dernière idée serait de faire héberger une page internet sur le SBC afin de pouvoir mettre à jour les fichiers audios via cette interface.

# Bilan

## Comparatif planning prévisionnel/reel

Une image contenant capture d’écran

Description générée automatiquement

Une image contenant capture d’écran

Description générée automatiquement

On remarque que la partie « Conception mécanique » n’a pas pu être traité. Cela s’explique par la nécessité d’effectuer la maintenance sur le projet Orgue Lumineux de 4ème année du projet SmartSystem. J’ai dû apporter mon aide pour trouver puis corriger un problème qui empêchait le bon fonctionnement de l’Orgue Lumineux.

En effet ce projet devait continuer d’être exposé dans différents endroits. Une solution temporaire fut trouvée, puis en parallèle du PFE, je m’efforçais de trouver le problème et de le corriger.

Cela a impacté la tâche conception mécanique.

## Bilan personnel

Ce projet m’a permis d’améliorer mes compétences en programmations orienté objets en C# avec la notion de programme cross-platform. Ce projet m’a surtout permis de développer mes compétences de l’outil Git avec principalement GitHub.

Ce que je retiens de ce projet ce sont mes points d’améliorations qui sont une meilleure estimation des tâches ainsi que prévoir une marge d’erreur pour un projet.

# Annexes

## Planning prévisionnel

Une image contenant capture d’écran

Description générée automatiquement

## Planning Réel

Une image contenant capture d’écran

Description générée automatiquement

# Bibliographie

Norme niveau de ligne audio :

<https://fr.wikipedia.org/wiki/Niveau_ligne>

Comparatif Raspberry Pi

<http://socialcompare.com/fr/comparison/raspberrypi-models-comparison>

Comparatif BeagleBone / Raspberry

<http://socialcompare.com/fr/comparison/rpi-and-beagle-models-comparison-4wypyz79>

MPR121

<http://micropython.org/resources/datasheets/MPR121.pdf>

Module Phidget :

<https://www.phidgets.com/?tier=3&catid=2&pcid=1&prodid=1021>

<https://www.phidgets.com/?tier=3&catid=15&pcid=13&prodid=1063>