

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)

**Cahier d’Analyse**

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

Table des matières

[Avertissement 3](#_Toc30154148)

[Introduction 4](#_Toc30154149)

[Contexte 4](#_Toc30154150)

[Objectif 4](#_Toc30154151)

[Analyse matériel 5](#_Toc30154152)

[Unité de traitement 5](#_Toc30154153)

[Solution avec microcontrôleur 5](#_Toc30154154)

[Solution avec une carte linux Embarquée 5](#_Toc30154155)

[Capteur de toucher 7](#_Toc30154156)

[MPR121 7](#_Toc30154157)

[Solutions phidget 8](#_Toc30154158)

[Analyse logiciel 10](#_Toc30154159)

[Principe de fonctionnement 10](#_Toc30154160)

[Point de vue du système 10](#_Toc30154161)

[Point de vue de la maintenance 10](#_Toc30154162)

[Démarrage de la carte Linux embarquée 11](#_Toc30154163)

[Objectif 11](#_Toc30154164)

[Solution 11](#_Toc30154165)

[Programme principale 12](#_Toc30154166)

[Objectif 12](#_Toc30154167)

[Langage 12](#_Toc30154168)

[UML 12](#_Toc30154169)

[Structure du programme 12](#_Toc30154170)

[Programme de mise à jour des fichiers audios 13](#_Toc30154171)

[Objectif 13](#_Toc30154172)

[Langage 13](#_Toc30154173)

[UML 14](#_Toc30154174)

[Table des illustrations 15](#_Toc30154175)

[BIBLIOGRAPHIE 16](#_Toc30154176)

# Avertissement

Ce document reflète le principe de raisonnement à un instant T. certains éléments, notamment sur l’analyse logiciel peuvent être amené à évoluer en fonction de contraintes non anticipé.

# Introduction

## Contexte

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.

## Objectif

Ce document a pour but de décrire l’architecture matériel et le principe de fonctionnement du système. La contrainte principale mis en évidence par le client est d’obtenir une solution qui fonctionne avec une maintenabilité simplifiée la plus simple. C’est pour aller dans ce sens, que certains choix techniques ont été faits, notamment sur le choix du matériel.

# Analyse matériel

## Unité de traitement

A la demande du client et suite à des échanges concernant la cible de programmation, le choix d’une carte linux embarqué fut décidé. Pour établir ce choix, 2 pistes ont été avancé. Le choix d’une architecture autours d’un microcontrôleur ou autours d’une carte linux embarqué.

### Solution avec microcontrôleur

Cette solution implique de choisir un microcontrôleur avec autour de lui, plusieurs modules ayant leur propre rôle tel que :

* Capteur de toucher
* Module audio
* Module lecteur audio
* Entrée/Sortie

Le choix des différents modules et capteurs utilisé impactera le choix du microcontrôleur. Un module/capteur peut communiquer avec un protocole bien spécifique qui peut avoir une incidence sur les critères de microcontrôleur tel que le cycle d’exécution plus rapide.

Cette solution nécessite une conception électronique plus avancé qu’avec une carte Linux embarqué. En effet, il faut sélectionner chaque modules/capteurs puis les interfacer sur un schéma et routage électroniques pour tout avoir sur une seule et même carte.

Cela implique plusieurs phases dont prototypes sur plaque d’essais, puis sur plaques en bakélites pour enfin terminer sur une à plusieurs versions sur PCB.

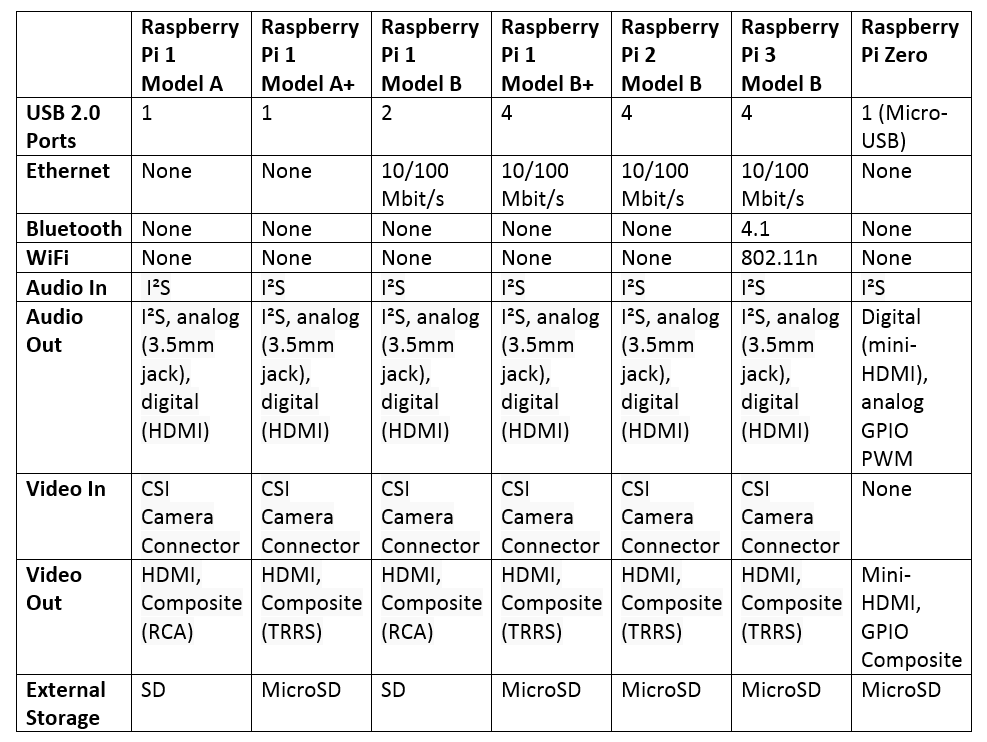
Cette solution offre des avantages sur la miniaturisation du système mais nécessite plus de temps ou de ressource.

### Solution avec une carte linux Embarquée

Cette solution simplifie certains problèmes, notamment sur l’aspect audio. En effet, la majorité des carte Linux embarquée dispose de module audio intégré (logicielle et matériel). De plus, ces cartes proposent des connectivités classiques de microcontrôleur tel que :

* I²C
* SPI
* UART
* Entrée/Sortie
* Analog Numeric Converter
* Digital Analog Converter
* Connectivité réseau (Wifi, Ethernet)
* Bluetooth

Cependant en fonction des cartes, toutes ces fonctions ne sont pas forcément présentes. C’est le cas pour les cartes Raspberry.

Figure 1 : tableau comparatif des modèles Raspberry

Ce qui nous intéressé est principalement la sortie physique audio avec un connecteur jack afin de pouvoir jouer le son sur un appareil dédiée.

Cette solution apporte donc un avantage et une simplification pour la tâche lecture audio. Elle apporte même des avantages pour des évolution future afin d’exploiter les autres caractérise de la carte tel qu’une mise à jour du programme ou des fichiers audios par le biais d’un réseau.

Cette solution est préférée par le client puisqu’il dispose de plus d’affinité avec les environnements Raspberry que sur microcontrôleur.

## Capteur de toucher

Il existe plusieurs types de capteur de toucher, la plupart fonctionne sur une technologie capacitive. Dans une optique ou il fallait choisir entre un microcontrôleur et une carte linux embarqué. J’ai exploré 2 pistes.

### MPR121

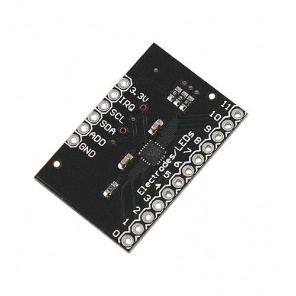


Figure 2 : MPR121

La première est celui d’un capteur de toucher capacitive, le MPR121. Ce dernier est un capteur communiquant via une liaison I²C. Il dispose de 12 channels indépendant et propose de modifier les seuils de déclenchement via ces registres.

Cette solution peut être intéressante afin de créer une solution sur une carte toute intégrée. Cependant, il faudra réaliser la librairie du composant ou le portage afin de l’exploité correctement.

### Solutions phidget



Figure 3 : Phidget Interface Kit 8/8/8

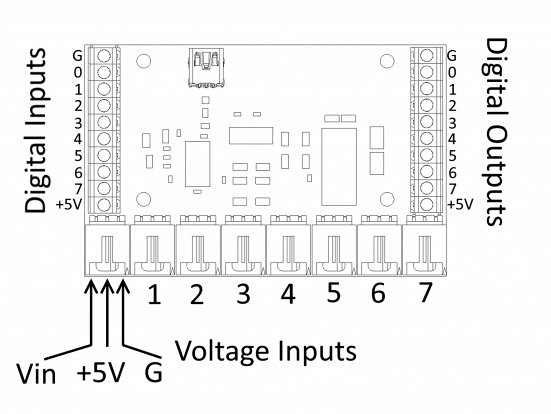


Figure 4 : Phidget Interface Kit 8/8/8 plan

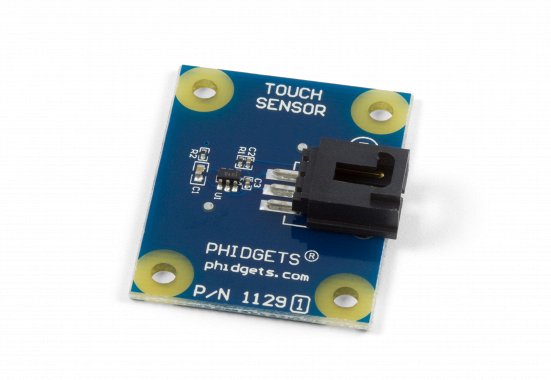


Figure 5 : Phidget capteur capacitive 1129

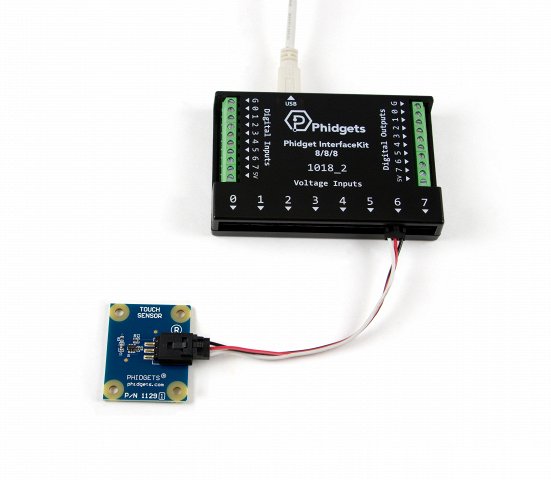


Figure 6 : Phidget interface 8/8/8 avec capteur capacitive

Cette solution consiste à des modules actionneurs se raccordant à un module interfaçage pour les connecter à un ordinateur. Les modules pouvant être raccorder à cette interface sont varier. Dans notre cas, ce sont des capteurs capacitifs.  
Une API en plusieurs langages de programmation par le constructeur est fournie par le constructeur pour tous ces modules d’interface.

* C
* C#/VB.NET
* Java
* JavaScript
* Max/MSP
* Python
* Swift

Si on rentre plus en profondeur, le module d’interface Phidget 8/8/8 est en réalité un contrôleur pour ordinateur se connectant en USB. Il propose les fonctionnalités suivantes :

* Lecture de niveau de tension analogiques
* Lecture de ratio niveau de tension (ration entre 5 et la tension analogique entrante)
* Entrée numérique Tout Ou Rien
* Sortie numérique Tout Ou Rien

Cette solution est privilégiée par le client puisqu’un existant fonctionne avec ce matérielle.

# Analyse logiciel

## Principe de fonctionnement

### Point de vue du système

L’objectif globale du système est de pouvoir :

1. Démarrer le système
2. Vérifier la présence d’une mise à jour des pistes audio
3. Lancer le programme principal

Pour vérifier la mise à jour des pistes audios, un système avec un fichier de configuration peut être étudié. Ce fichier peut comprendre les informations sur :

* Le nombre de capteur à utiliser
* Le nombre d’appui court à détecter
* Le nombre d’appui long à détecter
* Les pistes audios associé aux capteurs

### Point de vue de la maintenance

L’objectif de la maintenance est de mettre à jour les fichiers audios ainsi que de sélectionner le nombre de zone d’appui.

## Démarrage de la carte Linux embarquée

### Objectif

La carte Linux embarquée doit au démarrage vérifier la présence de la clé USB de mise à jour des fichiers. Si cette clé contient des nouveaux fichiers audios ou une nouvelle configuration, alors il récupère les nouvelles pistes audios ainsi que la nouvelle configuration.

### Solution

Plusieurs solutions sont en cours d’étude pour réaliser cette tâche. Il est possible de réaliser un programme spécifique qui se lancera au démarrage du système. Il déterminera s’il doit mettre à jour les fichiers audios et la configuration. Si oui, il met à jour puis lance le programme principal, sinon il lance le programme principal.

Cela peut être un programme spécifique ou un script qui sera exécuter au démarrage. C’est en cours d’étude.

## Programme principale

Ce programme consiste au programme de fonctionnement classique du système. C’est-à-dire, au démarrage du programme il doit rester en attente d’instruction. Lorsqu’il détectera des toucher, il devra lire les morceaux audio correspondant.

### Objectif

Le programme doit détecter des appuis long et courts et joué la piste audio correspondantes. Le système doit donc démarrer et exécuter le programme si une phase d’initiation est accomplie avec succès.

### Langage

La solution de capteur retenue par le client à une incidence sur le langage de programmation. En effet, le capteur propose de plusieurs API pour différents langages. Le choix retenu est le C#. Un Framework développé par Microsoft avec le dotNet rend compatible des programmes développés en C# pour une architecture ARM tournant sous une distribution Linux.

Ce Framework rend accessible en C# les entrées sorties des cartes Linux embarqué, les protocoles de communication tel que i²C/SPI mais aussi des lecteurs audios. Il s’agit du dotNet Core version 3.

### UML

#### Diagramme de cas d’utilisation

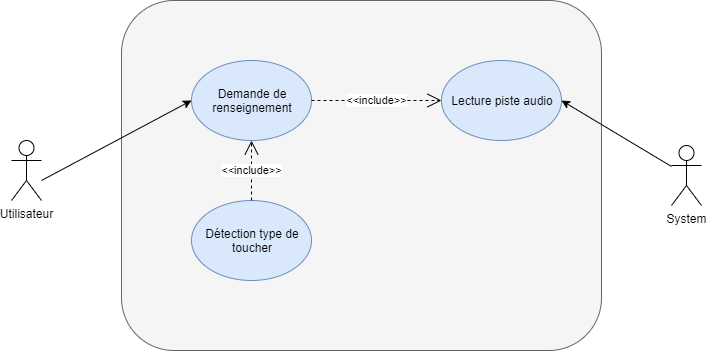


Figure 7 : Digramme cas d'utilisation du programme principale

### Structure du programme

Le programme sera orienté sous la forme de plusieurs threads qui seront exécuter en parallèle. Chaque thread aura ça tâche comme par exemple la détection d’appuis ou la lecture de piste audio. Il faudra donc faire attention à ne pas faire d’interblocage.

## Programme de mise à jour des fichiers audios

### Objectif

Ce programme a pour but de permettre à une personne d’effectuer la mise à jour des fichiers audio sur le système. En effet, au cours d’une utilisation, les informations évoquées par les pistes audio peuvent évoluer. Dans cette optique, il faut donc pouvoir mettre à jour ces fichiers. Dans un premier temps, la solution retenue est un logiciel sous Windows qui copiera les fichiers audios dans un espace de stockage type clé USB. Afin de limiter les mauvaises interactions, la clé USB sera configurée pour être reconnu par le système de la carte Linux embarqué. Ce logiciel doit être le plus simple possible pour l’utilisateur mais aussi le plus intuitif. Il doit être guidé à chaque étapes.

Ce logiciel doit donc :

* Proposer un « formatage » de la clé USB
* Associer un capteur à une piste audio
* Gérer le nombre de capteur de toucher
* Gérer le nombre d’appuie court
* Gérer le nombre d’appuie long
* Copier / coller les fichiers audios sur un espace de stockage

### Langage

Le logiciel a besoin d’une interface graphique pour l’utilisateur et devra être exécuter sous Windows. Dans cette optique, un programme en C# avec un Framework dotNet semble adapté.  
D’autre solution sont possible comme QtCreator ou encore java. Cependant le programme de la cible embarqué est réalisé en C#, il est donc intéressant de garder le même langage sachant l’environnement de développement VisualStudio offre une grande palette de composant graphique pour réaliser rapidement un logiciel avec IHM.

### UML

#### Diagramme de cas d’utilisation

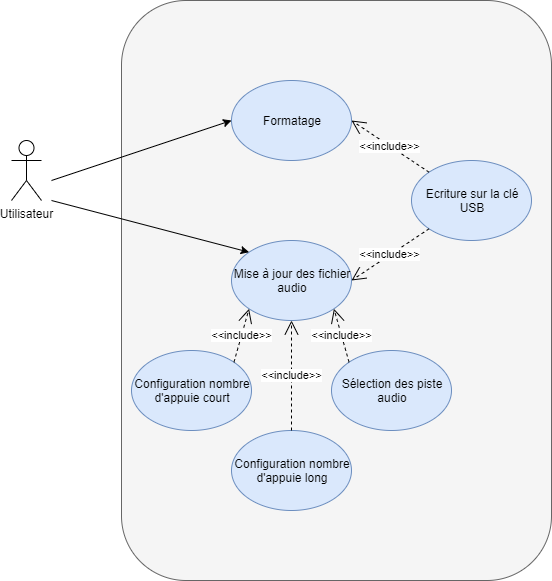
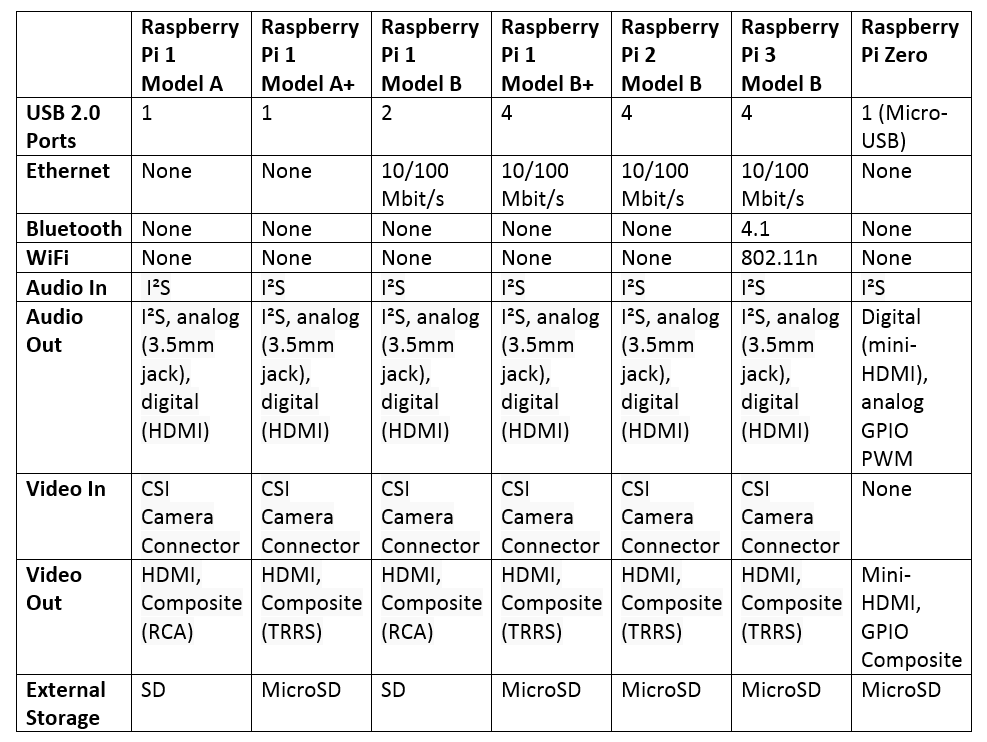


Figure 8 : Diagramme de cas d'utilisation logiciel mise à jour des fichiers audios

# Table des illustrations

[Figure 1 : tableau comparatif des modèles Raspberry 6](#_Toc30153718)

[Figure 2 : MPR121 7](#_Toc30153719)

[Figure 3 : Phidget Interface Kit 8/8/8 8](#_Toc30153720)

[Figure 4 : Phidget Interface Kit 8/8/8 plan 8](#_Toc30153721)

[Figure 5 : Phidget capteur capacitive 1129 9](#_Toc30153722)

[Figure 6 : Phidget interface 8/8/8 avec capteur capacitive 9](#_Toc30153723)

[Figure 7 : Digramme cas d'utilisation du programme principale 12](#_Toc30153724)

[Figure 8 : Diagramme de cas d'utilisation logiciel mise à jour des fichiers audios 14](#_Toc30153725)

# 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>