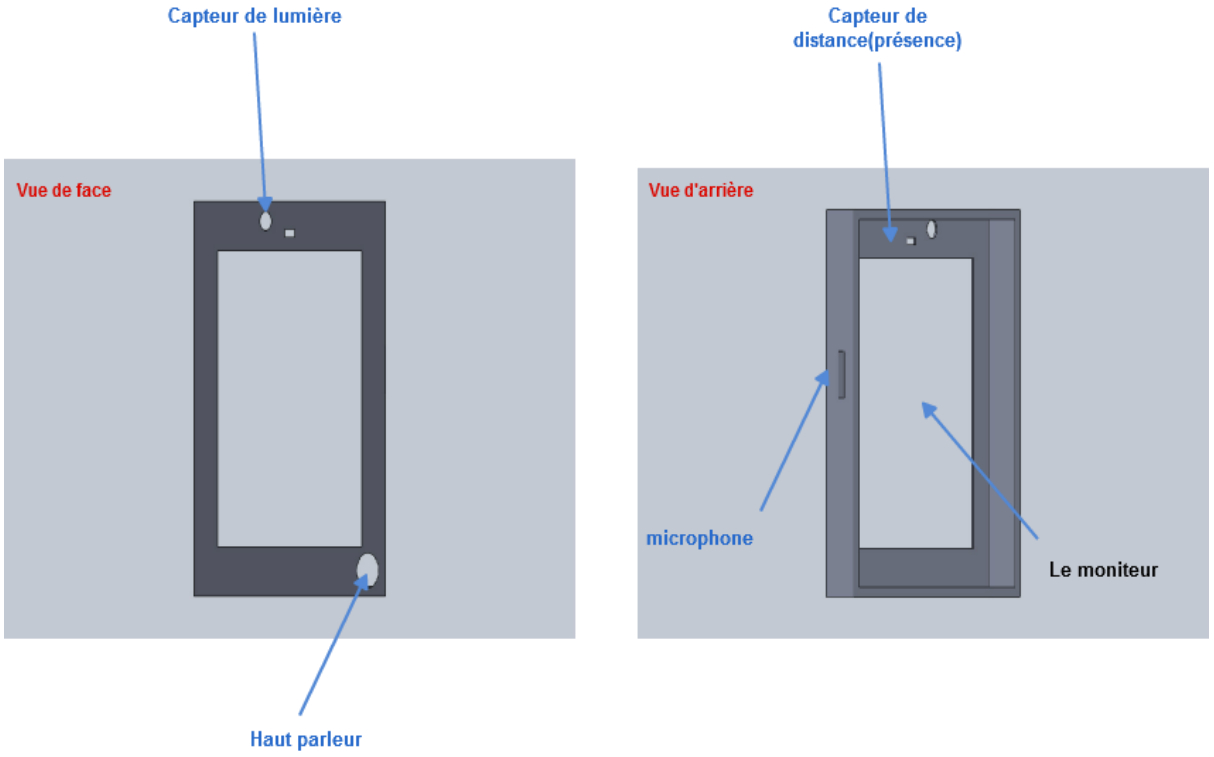
# RAPPORT - MIROIR MAGIQUE

## RAPPEL DU PROJET

L'idée de ce projet est le développement d'un miroir connecté capable d'interagir avec l'utilisateur. Ce miroir sera un assistant domestique pour l'utilisateur qui permet de faciliter le quotidien en collectant des informations à partir des services web qui seront par la suite affichées sur le miroir. Par exemple, avant de s'habiller ou de sortir l'utilisateur pourra être renseigné sur la météo, de l'état du trafic ou tout simplement les actualités du jour. Il faut juste que le miroir détecte sa présence.

Ce miroir est équipé d'un capteur de présence, un microphone (pour que l'utilisateur puisse interagir avec le miroir s'il veut une information particulière), un haut-parleur et un capteur de lumière (cela nous offre la possibilité de vérifier si la pièce où se trouve le miroir est bien éclairée).

Pour que le miroir soit capable d'afficher ses informations, nous allons utiliser le moniteur d'un PC ou d'une tablette avec un vitre sans tain. Le cadre sera comme ci-dessous :




## MATERIEL

Lors de l'entretien nous avons soulevé un problème au niveau du détecteur de présence. En effet celui que nous avons choisi permet uniquement de détecter une présence qui ne

dépasse pas les 3.5 cm. Il est évident que cela ne peut pas nous indiquer si oui ou non une personne est présente devant notre miroir.

Nous avons évoqué la possibilité d'un télémètre qui peut nous permettre de détecter la présence jusqu'à 4 mètres. Mais le problème est qu'on voudrait éteindre la lumière si plus personne est présent dans la pièce. Pour cela il faudrait que l'on détecte s'il y 'a du mouvement dans la pièce. Il faudrait alors combiner détecteur de présence et détecteur de mouvement, mais le détecteur de présence reste plus important dans la continuité de notre projet.

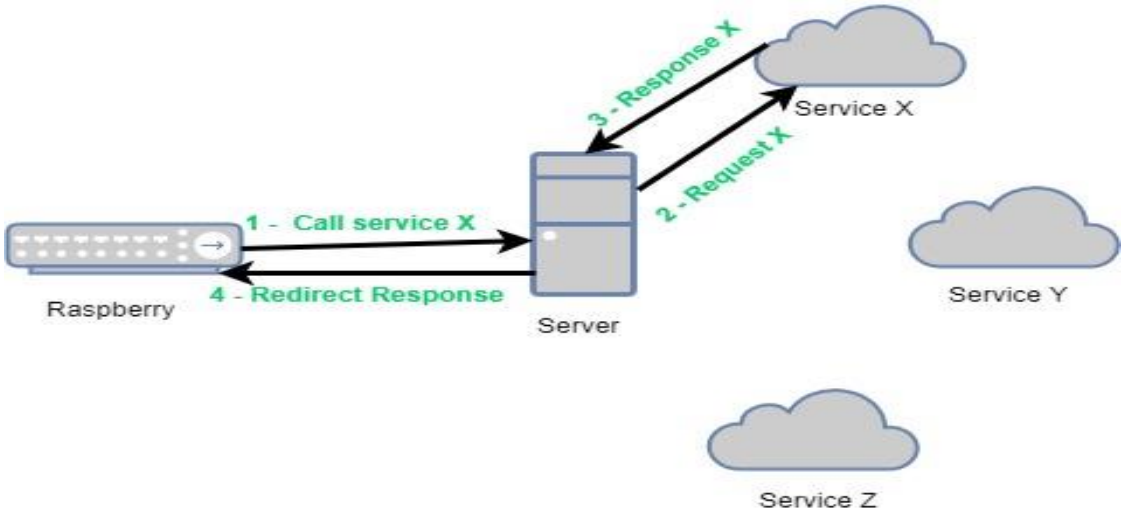
Aujourd'hui on en est au stade du prototype on peut se contenter du capteur jusqu'à 3,5 cm que l'on peut l'utiliser en passant la main devant pour signifier notre présence ; et le capteur de mouvement nous permet juste de rajouter une fonctionnalité : il n'est pas nécessaire.

Un autre problème soulevé est celui du haut-parleur. Celui que nous avons commandé n'était pas assez puissant pour l'utilisation que nous souhaitons à savoir informer l'utilisateur de manière sonore. Un autre haut-parleur a été commandé par M. Lavirotte.

## ARCHITECTURE

### ACTUELLE

Actuellement nous avons une raspberry pi qui, en fonction d'un évènement donné sur le **dispositif,** le traite et demande à la partie applicative (qui est un **serveur Node JS**) d'utiliser diffèrent Web services. Pour l'instant nous utilisons l'API Distance matrix de Google.



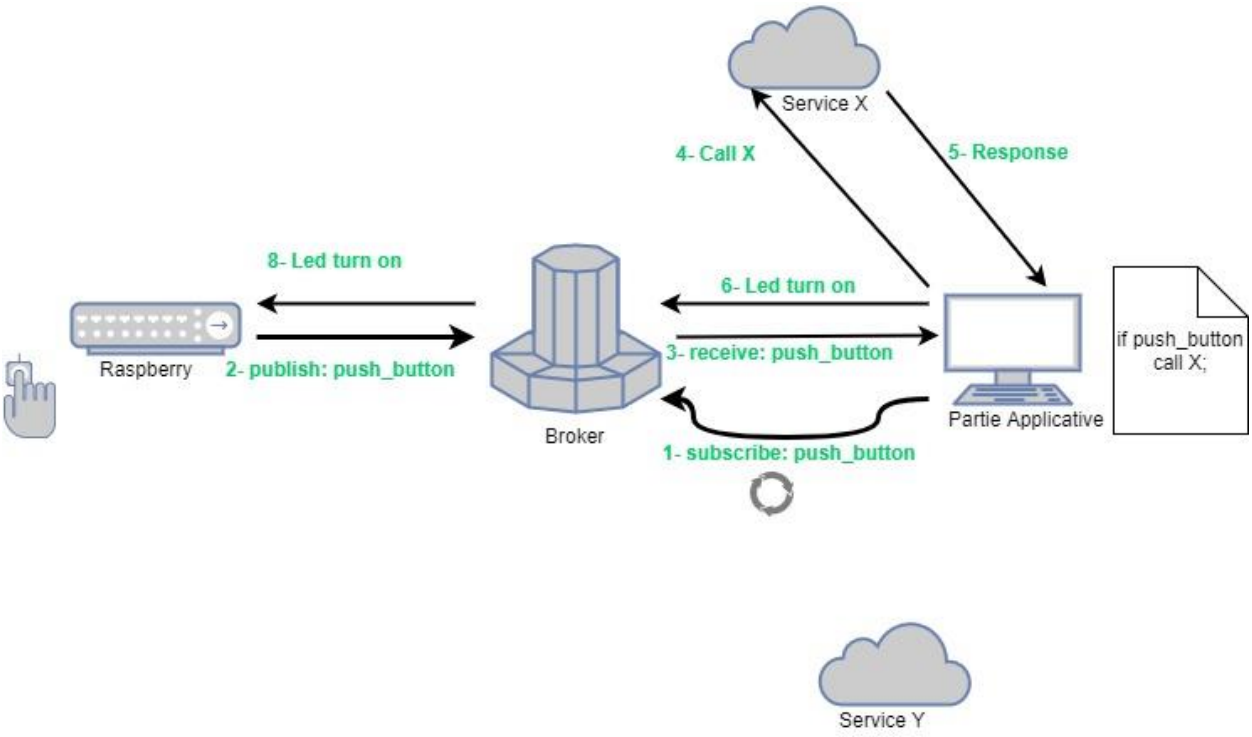

Le problème de cette architecture est que pour rajouter des services on est obligé de toucher la couche software présente sur la raspberry. Ce qui dans la cas d'un déploiement réel obligerait tous les utilisateurs à faire une mise-à-jour ou nous ramener leurs objets connectés pour qu'on change le software.

### FUTURE

Nous allons devoir bien séparer la partie applicative de la partie dispositif. Pour cela nous allons transformer la raspberry en serveur permettant de l'input/output sur le miroir. La

partie applicative/code métier va interroger des services extérieurs et sera donc un client des serveurs les proposant.

On aurait également la présence d'un broker permettant de récupérer les outputs du serveur raspberry. Par exemple, pour signifier la présence d'une personne devant le miroir, le serveur raspberry publierait cela auprès du broker, ainsi le broker serait au courant qu'une personne est devant le miroir. La partie applicative serait également notifié par un mécanisme d'abonnement.




Cette architecture a l'avantage de pouvoir permettre à d'autres objets de communiquer avec le miroir sans pour autant changer le code présent sur la raspberry.

## FUTURS SPRINTS

### SPRINT 2

Dans ce Sprint, la détection de présence reste toujours à travers le bouton puisque nous n'avons pas encore le capteur de présence.

Donc, la première étape est d'appliquer la nouvelle architecture présentée ci-dessus. C'est à dire, lorsqu'on clique sur le bouton, le Raspberry pi se connecte au Broker et publie les données au format JSON dans le Topic « push_button ». Pour exploiter ses données la partie applicative de notre système doit s'inscrire au même Topic. Dès que notre partie applicative détecte la présence de la personne (bouton cliqué), il se connecte au API

Distance matrix de Google qui présente notre premier service pour récupérer l'état du trafic. Ce résultat obtenu est transféré au Broker qui par la suite l'affiche sur le Raspberry.

Ensuite, nous allons utiliser la même architecture lors de la récupération des données du capteur de lumière. De façon que notre Raspberry se connecte au Broker et publie les données au format JSON dans le Topic "push_lumière". Le serveur fait un subscribe au même Topic, récupère les données et fait un simple traitement en vérifiant la luminosité de la chambre et renvoie une réponse au Broker, s'il faut changer la luminosité de la lampe. On peut simuler ce scénario en utilisant Wcomp pour communiquer notre broker à une lampe connectée. D'où l'utilité de cette architecture.

Finalement, on doit permettre la communication avec un service météo qui est déjà implémentée avec l'ancienne architecture.

### SPRINT 3

Le but est de prendre en charge le capteur microphone. L'idée est que à la fin de ce Sprint l'utilisateur puisse choisir le service qu'il veut affiché sur son miroir par le biais du traitement vocal des mots captés par le microphone. La personne doit prononcer par exemple "meteo" ou "trafic" ou "Agenda", etc… Ces mots clés seront envoyées au Broker dans le Topic "push_micro", la partie applicative s'inscrira au même Topic et traite les données récupérées par la suite le service qui sera exploité dépend du données récupérées.

Si nous avons récupéré le capteur de présence, nous allons l'utiliser au lieu du bouton sinon ça sera pour le prochain Sprint.