

P2017-p5-Accenture-RPI0-HowDoYouFeel
 Maxime CROTEAU

Dossier technique du projet - partie individuelle

Table des matières

1 -SITUATION DANS LE PROJET.....	3
1.1 -RAPPEL DES TÂCHES PROFESSIONNELLES À RÉALISER :.....	3
1.1 -EXTRAIT DU BACKLOG :.....	4
1.2 -PRÉSENTATION DE LA PARTIE PERSONNELLE :.....	5
1.2.1 -Introduction.....	5
1.2.2 -Synoptique de la réalisation.....	6
1.2.3 -Diagramme de déploiement.....	7
1.2.4 -Aperçu du système de vote.....	8
1.2.5 -Diagramme de cas d'utilisation : Gestion de vote.....	9
2 -RÉALISATION GÉNÉRAL.....	10
2.1 -SOLUTIONS TECHNOLOGIQUES MISES EN ŒUVRE :.....	10
2.2 -INTERNET OF THINGS – INTERNET DES OBJETS – IOT :.....	11
2.3 -RASPBERRY PI COMPUTE MODULE :.....	14
2.4 -SCHÉMA ÉLECTRIQUE DU MODULE DE VOTE :.....	15
2.5 -MISE EN PLACE DU POINT D'ACCÈS Wi-Fi :.....	16
3 -RÉALISATION DE LA TÂCHE CAPTER L'APPUI SUR UN BOUTON.....	17
3.1 -CAS D'UTILISATION :.....	17
3.2 -DIAGRAMME DE DÉPLOIEMENT :.....	17
3.3 -CONCEPTION DÉTAILLÉE :.....	18
3.4 -DIAGRAMME DE SÉQUENCE :.....	18
4 -RÉALISATION DE LA TÂCHE TRANSMETTRE LE VOTE.....	19
4.1 -CAS D'UTILISATION :.....	19
4.2 -DIAGRAMME DE DÉPLOIEMENT :.....	19
4.3 -CONCEPTION DÉTAILLÉE :.....	20
4.4 -DIAGRAMME DE DÉPLOIEMENT :.....	20
5 -RÉALISATION DU FEEDBACK POUR L'UTILISATEUR.....	21
5.1 -CAS D'UTILISATION :.....	21
5.2 -DIAGRAMME DE DÉPLOIEMENT :.....	21
5.3 -CONCEPTION DÉTAILLÉE :.....	22
5.4 -DIAGRAMME DE SÉQUENCE :.....	23
6 -PLANS DES TESTS UNITAIRES.....	24
7 -TEST DE LA GESTION DES VOTES.....	24
7.1 -PROCÉDURE DE TEST :.....	24

7.2 -RAPPORT D'EXÉCUTION :	25
8 -TEST DE L'APPUI SUR LE BOUTON POUSSOIR.....	26
8.1 -PROCÉDURE DE TEST :	26
8.2 -CODE SOURCE DU TEST :	27
8.3 -RAPPORT D'EXÉCUTION :	28
9 -TEST DE LA TRANSMISSION DU VOTE.....	29
9.1 -PROCÉDURE DE TEST :	29
9.2 -CODE SOURCE DU TEST :	31
9.3 -RAPPORT D'EXÉCUTION :	31
10 -TEST DU FEEDBACK UTILISATEUR.....	32
10.1 -DESCRIPTION DU TEST :	32
10.2 -PROCÉDURE DE TEST :	33
10.3 -CODE SOURCE DU TEST :	34
10.4 -RAPPORT D'EXÉCUTION :	34
11 -BILAN DE LA RÉALISATION PERSONNELLE.....	35
11.1 -STATUT DES FONCTIONS À CHARGES :	35
11.2 -CONCLUSION :	35

1 - Situation dans le projet

1.1 - Rappel des tâches professionnelles à réaliser :

<i>Fonction principale de l'application (Fpi)</i>	<i>Description</i>
Fp1	Gérer les votes Cette fonction exécutée en permanence après la mise sous tension du capteur est chargée de la datation et du stockage local des votes, ainsi que d'envoyer régulièrement les votes sur le Cloud How Do You Feel.
Fp2	Déetecter un vote Cette fonction est chargée de détecter le vote d'un utilisateur du système. Le flux en provenance du système à Boutons Poussoirs ou du système de reconnaissance faciale sert d'interface avec le capteur.
Fp3	Dater le vote Cette fonction est chargée de dater le vote d'un utilisateur du système.
Fp4	Transmettre les votes Cette fonction est chargée de transmettre sur le cloud les votes stockés localement.
Fp5	DéTECTER un vote par boutons poussoirs Cette fonction est chargée de détecter l'action associée au bouton poussoir
Fp6	Capter l'appui sur un bouton Cette fonction est chargée de capter le changement d'état d'un bouton poussoir

1.1 - Extrait du backlog :

Applications	Groupe de taches	Tache	Charge
Toutes	Lancement	Installer les services Trac et Subversion en DMZ sur la VM projet	
	Lancement	Mettre en œuvre la plateforme Raspberry	x
	Lancement	Mettre en œuvre la base de données HANA	x
	Lancement	Mettre en œuvre le service Cloud	
	Spécifications	Décrire les cas d'utilisation (Use case ou User Story)	
	Spécifications	Prototypage des interfaces-utilisateur (Mockup)	
	Conception	Décrire les tables et relations entre les tables de la base de données	x
	Raspbian	Installer et configurer le système d'exploitation Raspbian	
Application embarquée	Bouton Pousoir	Choisir un système à Boutons Pousoirs	x
	Bouton Pousoir	Mettre en œuvre le système à Boutons Pousoirs	x
	Bouton Pousoir	DéTECTer les événements du système à Boutons Pousoirs	x
	Bouton Pousoir	Filtrer les événements du système à Boutons Pousoirs	x
	Communiquer	Choisir le protocole de communication avec le Cloud	x
	Communiquer	Mettre en œuvre une communication avec le Cloud	x
	Communiquer	Communiquer les votes au Cloud	x
	Gérer	Ordonnancer les votes	x
FrontOffice HowDoYouFeel	Vues Capteurs	Afficher les courbes des votes pour l'ensemble des capteurs	x
	Vues Capteurs	Afficher les courbes des votes pour un capteur	
	Vues Capteurs	Choisir un capteur	x
	Vues Capteurs	Choisir une question	x
	Vues Capteurs	Choisir une période	x
	Capteurs	Concevoir et coder le Modèle	
BackOffice HowDoYouFeel	BDD	Concevoir les tables de la base de données	x
	BDD	Installer la base de données sur le Cloud	x
	Vue utilisateurs	Afficher/Ajouter/Modifier/Supprimer un collaborateur	
	Vues Capteurs	Afficher/Ajouter/Modifier/Supprimer un capteur	
	Vue mesures	Purger les tables des votes	x
	Agrégation	Coder le mécanisme d'agrégation des tables de votes	
	Surveillance	Mettre en œuvre le système de log sur le Cloud	
Toutes	Maquette Réseau	Réaliser la maquette réseau	
	Management	Participer à la planification des Sprints (Sprint Planning Meeting)	x
	Management	Participer aux Scrum quotidiens (Daily Scrum Meeting)	x
	Management	Participer aux rétrospectives de Sprint (Sprint Review Meeting)	x
	Documentation	Rédiger les manuels d'installation et d'utilisation de la maquette	x

1.2 - Présentation de la partie personnelle :

1.2.1 - Introduction

Je suis chargé, dans ce projet, de mettre en place la partie embarquée du système. C'est à dire, configurer la raspberry, gérer les envois des votes vers l'Internet des Objets SAP, la mise en place des boutons poussoirs pour les votes, et les LED pour le feedback utilisateur.

La réalisation s'est déroulée en plusieurs phases jusqu'à présent :



1. Lecture du Cahier des charges

2. Analyses

1. Dialogue avec M. Salerne pour éclaircir les points incompris

3. Sprint 1: Création et documentation sur l'environnement de travail

1. Création d'un compte trial SAP
2. Mise en place du système d'exploitation raspbian sur la RPI module dev kit
3. Mise en place de l'environnement nodeJS sur la raspberry
4. Envoi de votes (valeurs binaire) depuis la raspberry vers le Cloud

4. Sprint 2: Mise en place de l'IOT

1. Découverte et mise en place du service IoT de SAP
2. Envoi de votes depuis la raspberry vers le Cloud via événement à l'aide du protocole mqtt et de l'IOT SAP

5. Sprint 3: Mise en place des boutons poussoir et feedback pour l'utilisateur

1. Mise en place de boutons poussoirs pour l'envoi des votes
2. Mise en place des LEDs permettant à l'utilisateur d'être informé des étapes de vote

L'environnement SAP et l'utilisation du protocole MQTT ainsi que l'IOT ont été imposés par l'entreprise Accenture.

Le projet se fait, en totalité, sur cet environnement. La base de données est gérée sur SAP qui est compatible avec Eclipse Luna. L'IHM est codée en SAPUI 5. De plus, l'envoi des votes via la Rasberry module dev kit est effectué via l'Internet des objets (IOT) disponible sur l'environnement SAP, avec un compte trial.

Le service XSJS est un langage de programmation d'application en JavaScript. Il permet d'exposer les données stockées dans les tables de la base de données ou les vues du côté client.

1.2.2 - Synoptique de la réalisation

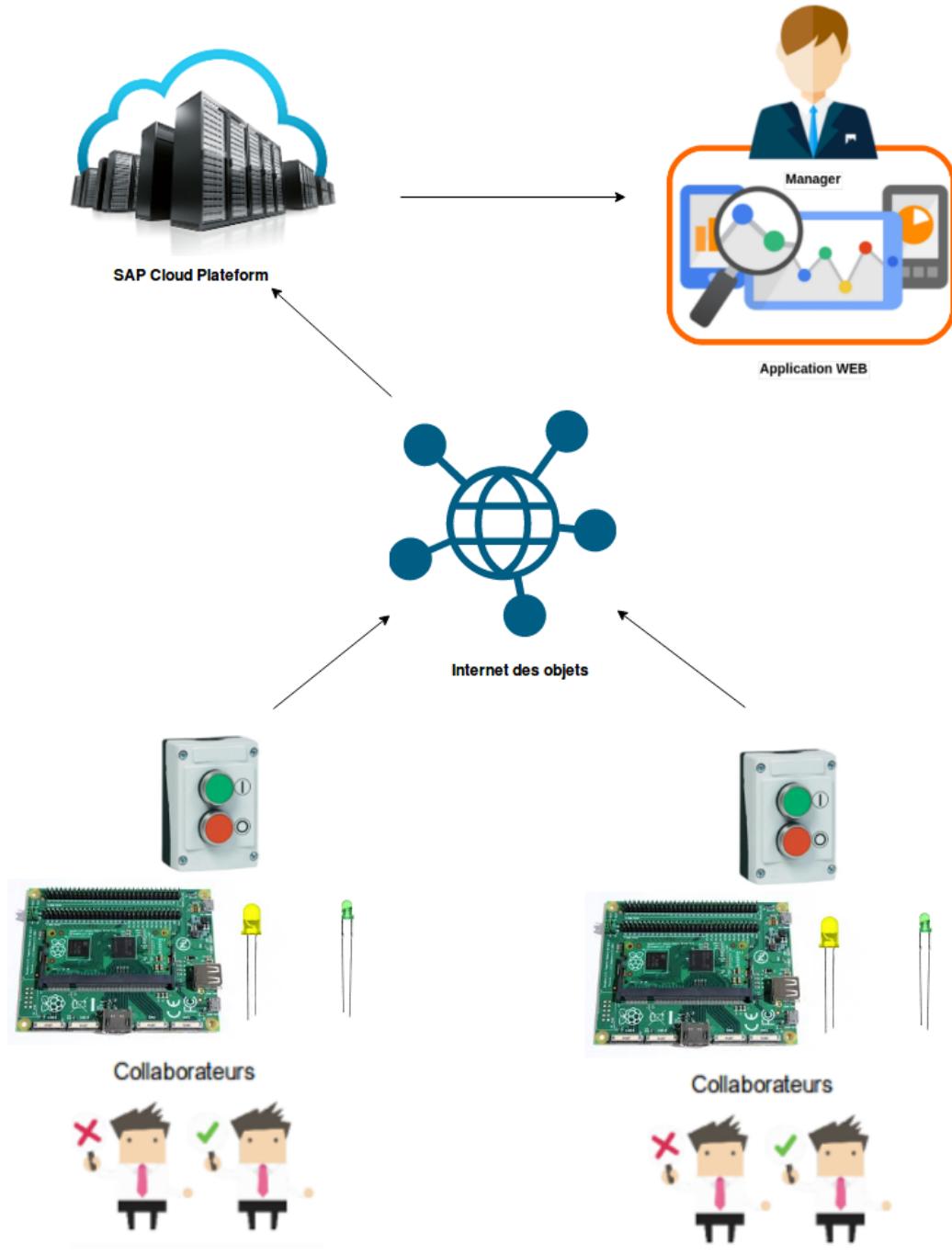


figure 1: Synoptique

L'utilisateur vote en appuyant sur un bouton poussoir, une fois le vote publié à l'aide du protocole MQTT, il est stocké sur un broker. L'IOT de SAP va venir chercher les nouveaux votes sur le broker et se chargera des les envoyer vers la base de données. Une fois sur la base de données, les votes sont récupérés et affichés sous forme graphique et consultable depuis une interface web par les managers.

1.2.3 - Diagramme de déploiement

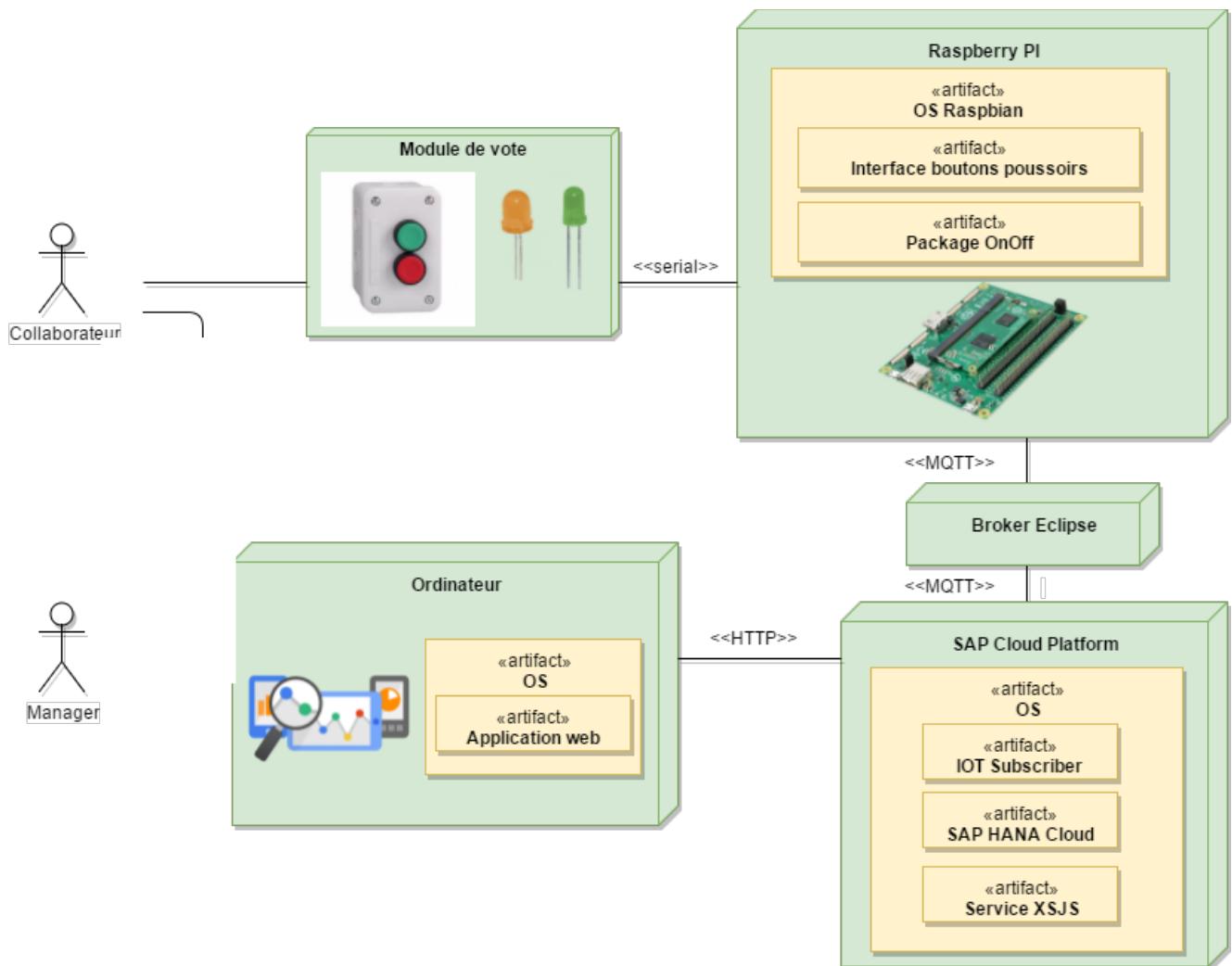


figure 2: Diagramme de déploiement

Le diagramme de déploiement ci-dessus permet de détailler les différents acteurs physiques du système :

- La **carte raspberry module dev kit**, avec ses boutons pousoirs et les LED forment le module de vote, interface permettant aux utilisateurs de voter. On utilise l'opérateur système Raspbian Jessie, sur lequel est installé l'environnement NodeJs. On y retrouve également le package « OnOff », package permettant d'effectuer le traitement des différents GPIO, ainsi que le script développé permettant de l'exploiter, et de publier les votes au broker.
- Le **Broker** est une plateforme d'échange entre les **capteurs** et le **SAP Cloud Platform**. (voir ci-dessous pour une explication détaillée)
- La **SAP Cloud Platform**, serveur Cloud de SAP héberge l'application web, le serveur de base de données, et les différents services proposés par SAP (XSJS).
- L'**Ordinateur**, ou tout autre périphérique équipé d'un navigateur web, permet d'accéder à l'application HowDoYouFeel. Il proposera cependant une interface différente, selon le type d'utilisateur ; (les collaborateurs n'auront accès qu'à l'affichage des votes, tandis que le manager aura accès à la partie administration de l'application).

1.2.4 - Aperçu du système de vote

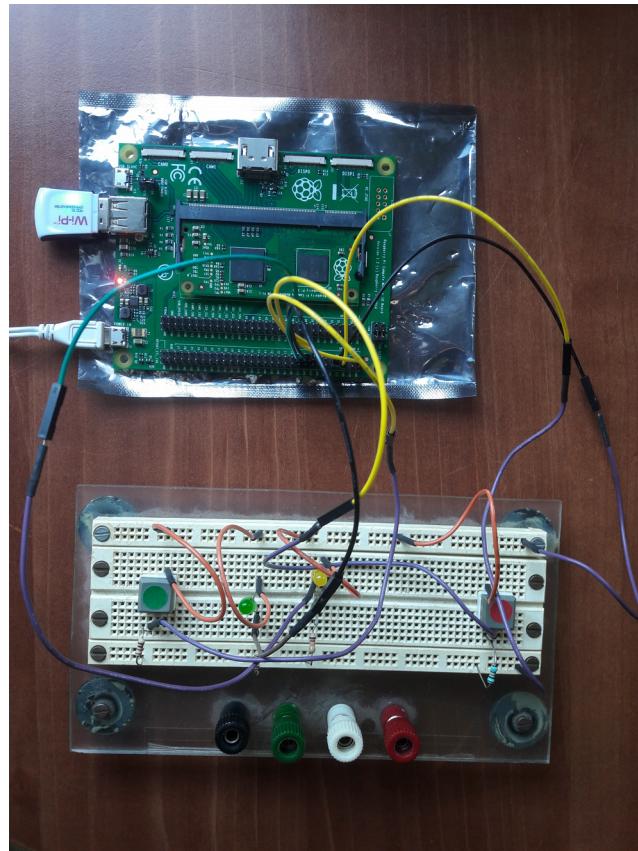


figure 3: Maquette du système de vote complet

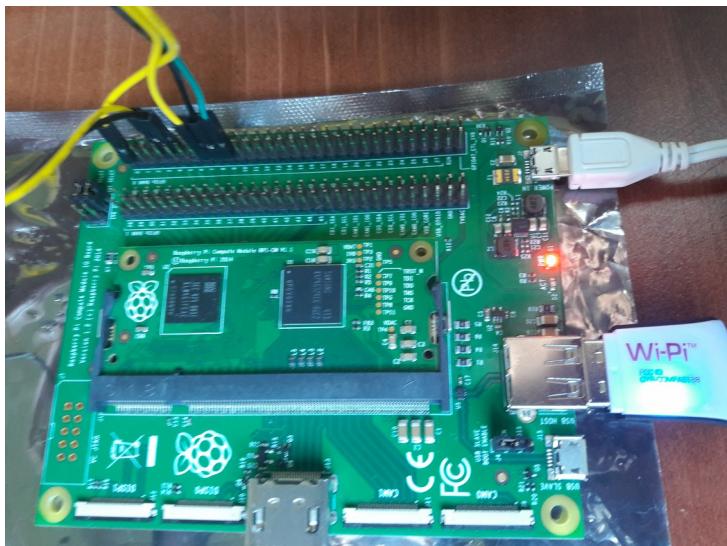


figure 4: Raspberry Compute module + Wipi

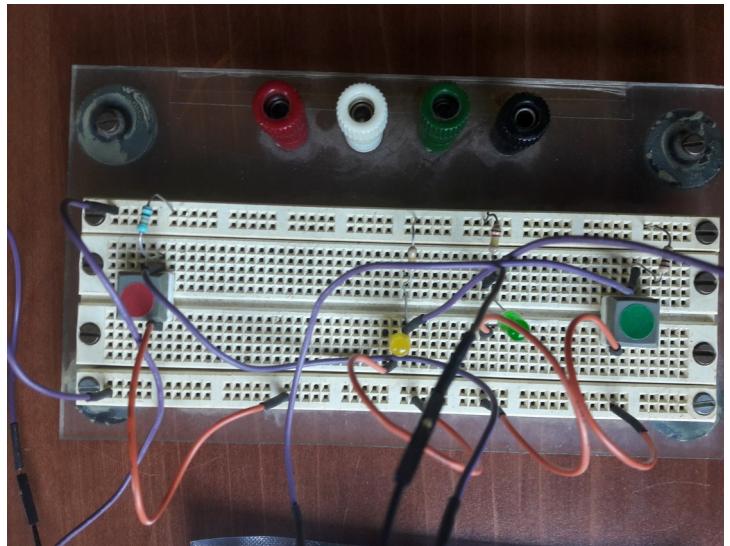


figure 5 : Boutons poussoirs + LEDs

1.2.5 - Diagramme de cas d'utilisation : Gestion de vote

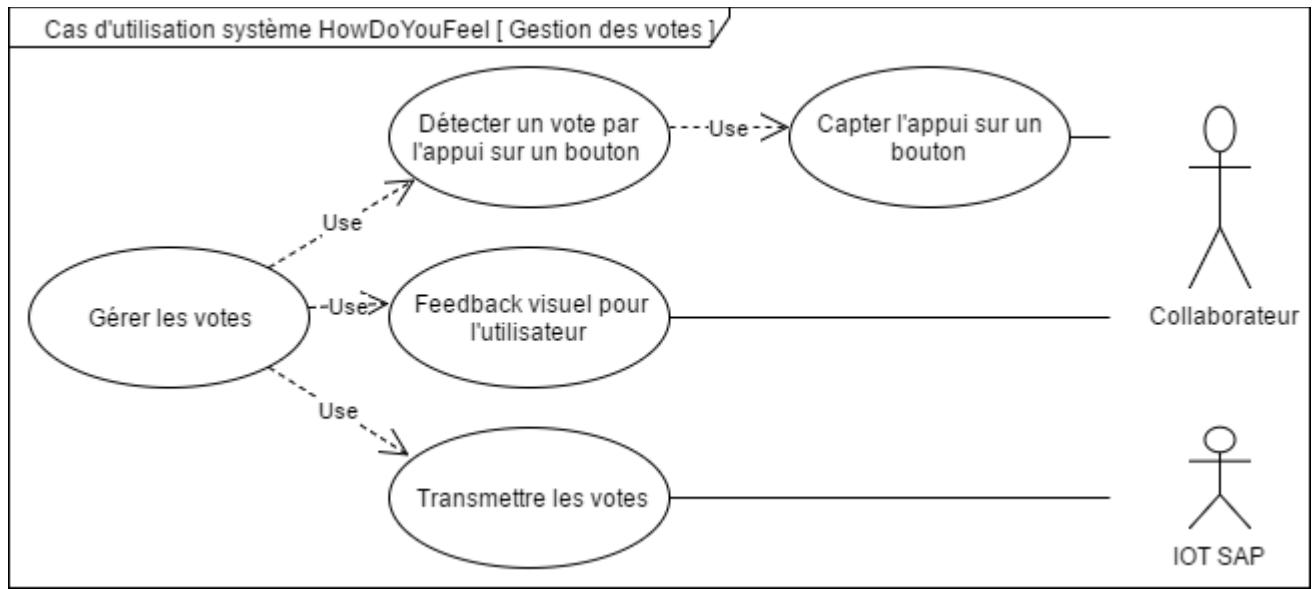


figure 6: Diagramme de cas d'utilisation : Gestion de vote

2 - Réalisation Général

2.1 - Solutions technologiques mises en œuvre :

	<p>Systems, Applications and Products for data processing.</p> <p>SAP est un progiciel destiné à la conception de systèmes logiciels. Il prend en charge le système dans son cycle de vie intégral : dès sa conception jusqu'à la réalisation de l'application (SAPUI5), ou même le stockage des données (SAP HANA).</p> <p>SAP propose également de nombreux services (souvent payants, mais accessibles par le biais d'un compte d'essai) comme le service IoT (Internet of Things), le service de traduction (SAP Translation Hub), un gestionnaire de version GIT etc. L'ensemble des services et conceptions réalisées avec SAP est hébergé sur la SAP Cloud Platform.</p>
	<p>SAP Hana :</p> <p>Système de base de données proposé par SAP, polyvalent par sa capacité de lecture verticales des tables (lecture des colonnes, et non pas des entrées), et son utilisation de la mémoire vive afin d'accélérer les traitements.</p>
	<p>Node js :</p> <p>Environnement de développement JavaScript orienté vers les applications réseau.</p>

	CPU :	La carte Raspberry Pi SODIMM (Carte SODIMM : 6,5 cm par 3 cm) contient la puce BCM2835 de 512 Mo de RAM comprenant une carte mémoire Flash eMMC
	Mémoire :	512 Mo de RAM eMMC Flash 4GB pour le démarrage de l'OS.
	Communication	Un Raspberry Pi Compute Module Une Carte IO Compute Raspberry Pi

Ces différents outils et matériels nous ont été imposés par le cahier des charges.

2.2 - Internet Of Things – Internet des objets – IOT :

2.2.1 - Rappel du synoptique :

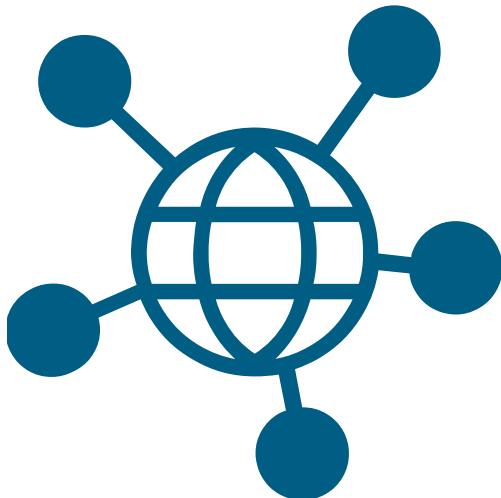
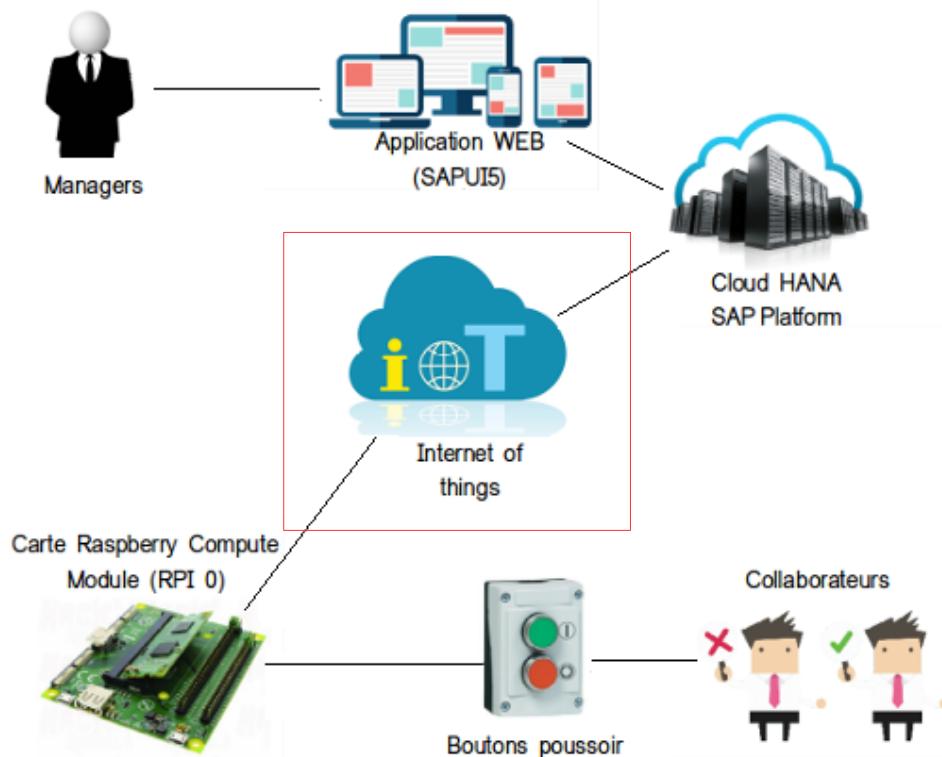


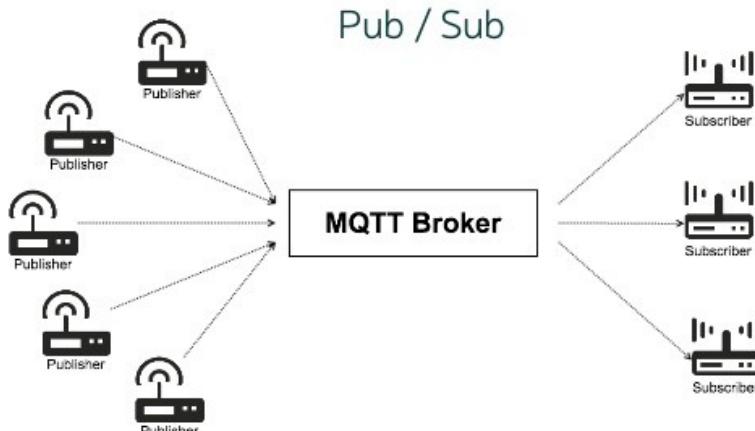
figure 7: Internet des objets

L'internet des objets est une technologie permettant de connecter les objets du quotidien à internet. On peut, par exemple, connecter son frigo à internet afin de relever des données comme la température, la consommation, etc ... depuis un smartphone.

Cette technologie est innovante et en plein essor, elle est notamment à l'origine de l'augmentation exponentielle des données circulant sur internet.

Dans le cadre de notre projet, l'IOT est une technologie adaptée : elle permet de facilement configurer et de connecter plusieurs modules de vote si nécessaire. De plus, SAP Cloud Plateform possède un service « Internet des objets » permettant de faciliter la configuration des ces objets.

Dans le cadre du projet, c'est le protocole MQTT qui est utilisé pour envoyer les votes. MQTT est un protocole de messagerie utilisant le modèle Publish / Subscribe (Pub/Sub), soit Publier / Souscrire en Français, basé sur le protocole TCP/IP.



8

figure 8: Schema Pub/Sub

Le modèle pub/sub est dit de publication et de souscription de messages dans lequel les émetteurs (*publisher* sur le schéma ci-dessus) ne destinent pas leurs messages à des destinataires distincts (*subscriber*). À la place, une catégorie est associée aux messages émis sans savoir s'il y a des destinataires, ces catégories sont initialisées sur un Broker. De la même manière, les destinataires souscrivent aux catégories les intéressent, et ne reçoivent que les messages correspondant, sans savoir s'il y a des émetteurs.

Dans notre projet, les Publishers sont les modules de vote. Ils envoient les votes sur un broker. Ici, nous avons utilisé le Broker public d'Eclipse qui permet de remplir totalement les fonctionnalités qui lui sont demandées, et qui a permis une réelle économie de temps étant donné le peu de configuration qu'il demande, ce qui a permis de se focaliser sur la configuration de l'IOT. C'est l'IOT SAP qui a le rôle de Subscriber, et qui a pour charge de récupérer les votes se trouvant sur le répertoire du Broker et de les transmettre à la base de données.

Le choix d'un broker public est un choix peu sécurisé étant donné que n'importe qui peut accéder au répertoire de données, mais vu le type de données envoyées, il a été convenu que l'utilisation d'un tel broker pouvait être convenable.

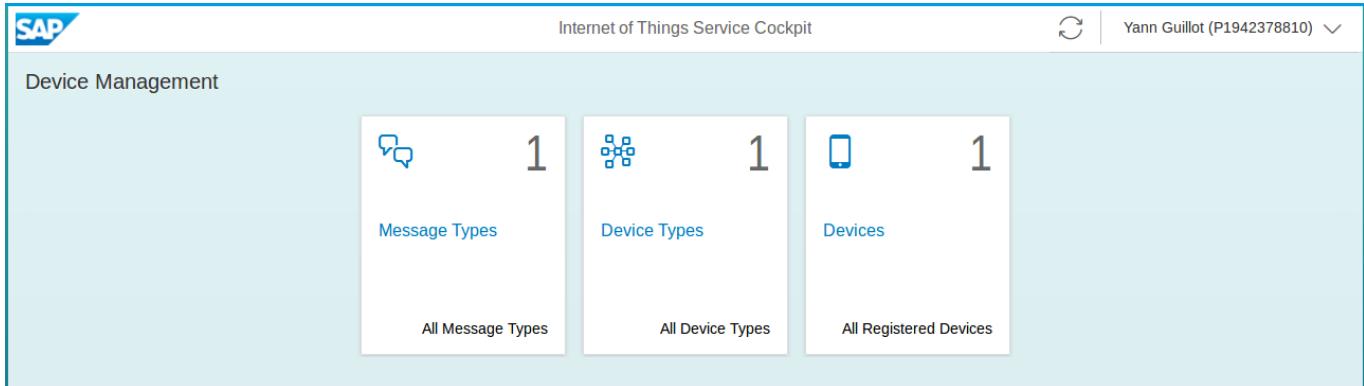


figure 9: Internet of Things SAP

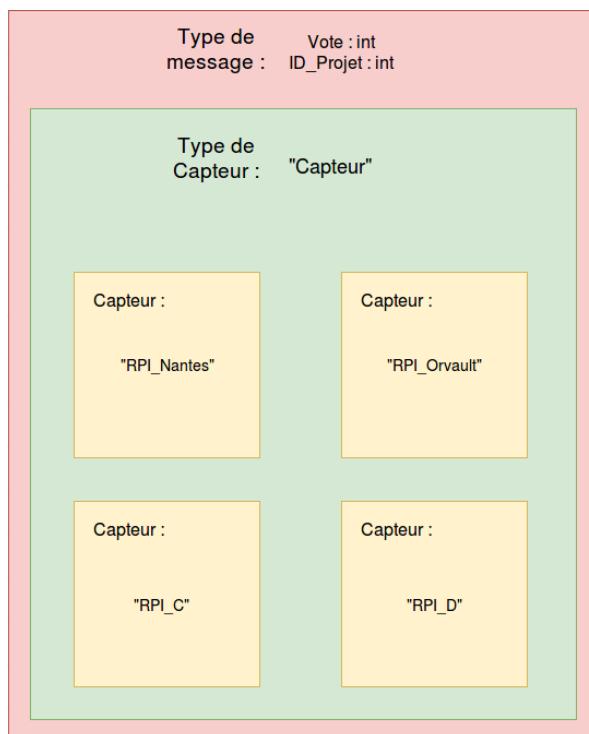
La configuration des capteurs se fait en plusieurs étapes :

Dans un premier temps, il faut créer un Type de Message (*Message Types sur le schéma ci-dessus*) dans lequel on définit les données à envoyer et sous quel type. Dans notre projet nous avons donc un Message Types contenant deux données à envoyer, une donnée « Vote » et une donnée « ID_Projet », toutes les deux dans le type Integer (int). La donnée « Vote » contient le vote envoyé (0 ou 1), la donnée « ID_Projet » contient l'identifiant du projet vers lequel le vote est envoyé (Ex : « 1 » pour le projet numéro 1). Nous avons intitulé ce type de message « TypeVote ».

Il faut ensuite créer un type d'objet qu'on associera au type de message précédemment créé. Pour notre projet nous avons donc un Device Types intitulé « Capteur » associé à notre type de message « TypeVote ».

Pour finir, il reste à créer des capteurs associés à notre type d'objet. Nous avons donc créé un capteur intitulé « RPI_Nantes » lié à notre type d'objet « Capteur ».

A partir de ce moment, il est simple de créer plusieurs capteurs et de les associer au même type d'objet afin de recevoir des votes de plusieurs capteurs différents, chaque capteur créé ayant un identifiant différent, il est simple de distinguer d'où viennent les différents votes.



Ce schéma ci-contre illustre la configuration de l'IOT SAP. On peut y voir que le Type de message, créé en premier, englobe le type de capteur qui, lui, englobe les différents capteurs.

figure 10: Configuration IOT SAP

2.3 - Raspberry Pi Compute Module :

2.3.1 - Rappel du synoptique :

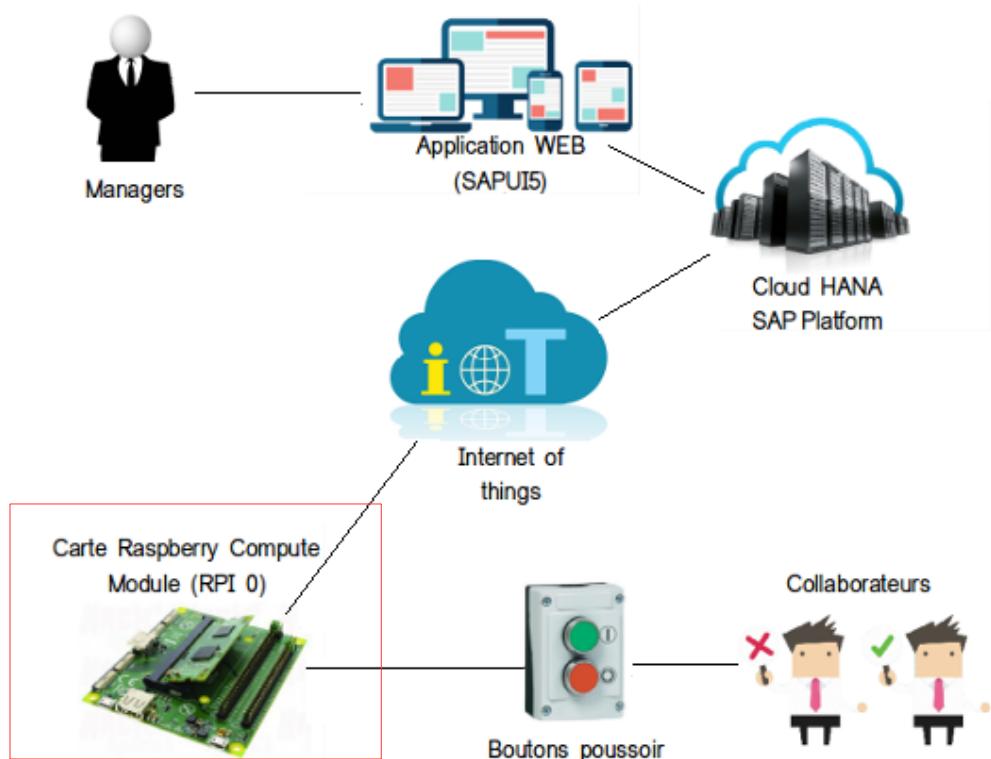


figure 11: Raspberry Pi Compute Module

La mise en place du système de vote a été effectuée sur une carte Raspberry Pi Compute Module qui propose de nombreux ports GPIO. Ceci permet d'une part d'y associer facilement les deux LEDs et les deux boutons poussoirs, et d'autre part de profiter de l'alimentation en 3,3 Volts et de la masse. Ce modèle est dépourvu de port ethernet et de slot carte SD et nécessite donc une liaison wi-fi pour pouvoir communiquer vers le broker pour envoyer des votes. Sur ce module, nous avons installé le système d'opération Raspbian, couplé à l'environnement NodeJS.

Raspbian est la distribution Linux la plus adaptée aux modules RPI. Elle est légère et consomme peu de ressources.

NodeJS est un environnement JavaScript adapté aux besoins d'un serveur. Nous l'utilisons afin de créer un script permettant l'envoi des votes en réaction à un événement (appui sur bouton poussoir).

2.4 - Schéma électrique du module de vote :

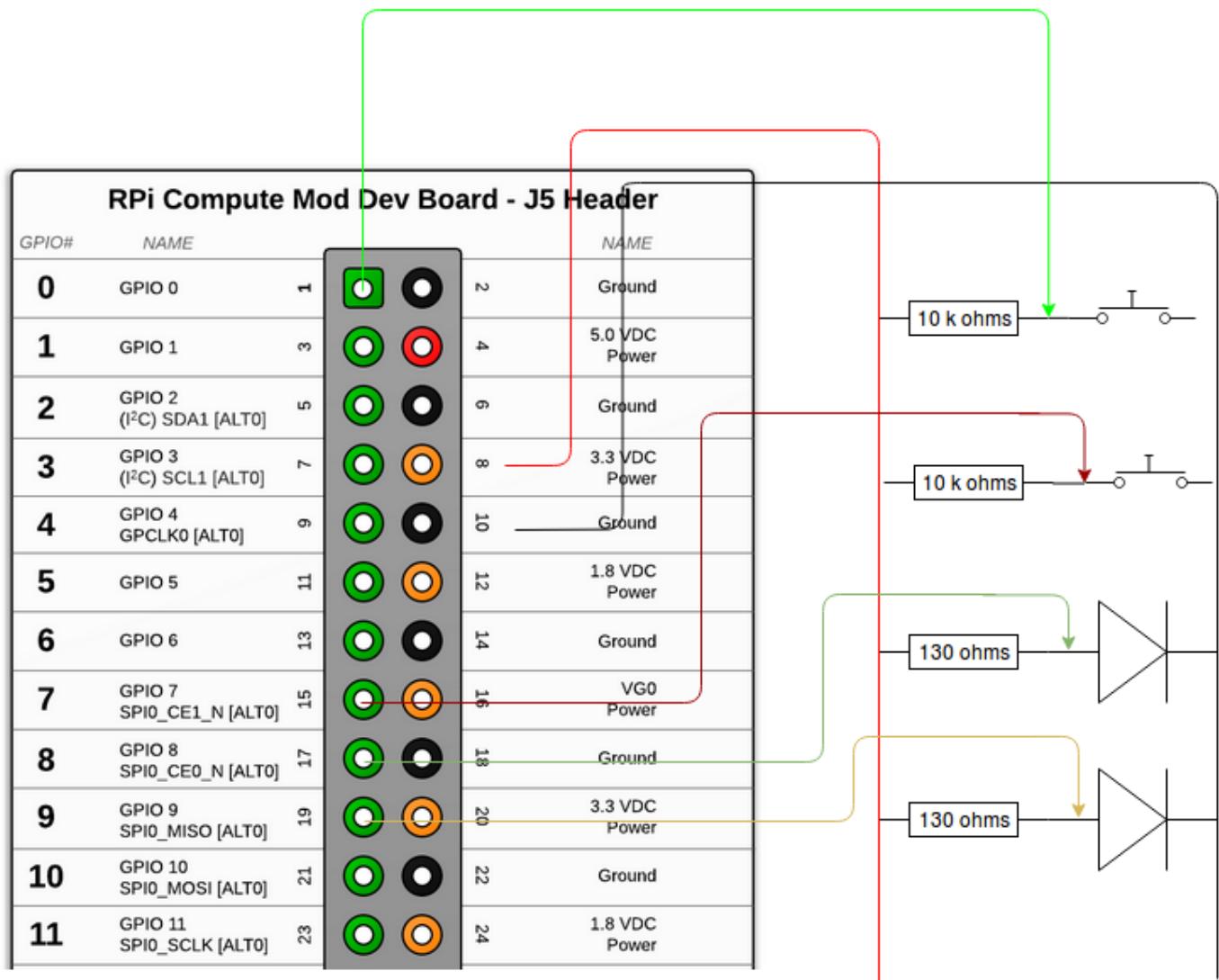


figure 12: Schéma électrique du montage

Voici le schéma électrique du module de vote. Nous avons le schéma des GPIOs sur le Header J5 du RPi Compute Module, seul les 11 premiers GPIOs apparaissent, les autres sont coupés pour un soucis de visibilité, et ne sont pas utilisés.

Le système est donc composé de deux boutons poussoirs normalement ouverts, liés à deux résistances de 10k ohms, et deux LEDs, verte et jaune, liées à deux résistances de 130 ohms. Le calcul de résistance pour une LED est, par exemple, dans notre cas :

$$R = \frac{U_{\text{alim}} - U_{\text{LED}}}{I}$$

Valeur en Ohms

Tension d'alim (en Volts)

Tension de seuil de la LED (en Volts)

Courant souhaité dans la LED (en Ampères)

Dans notre cas nous avons :

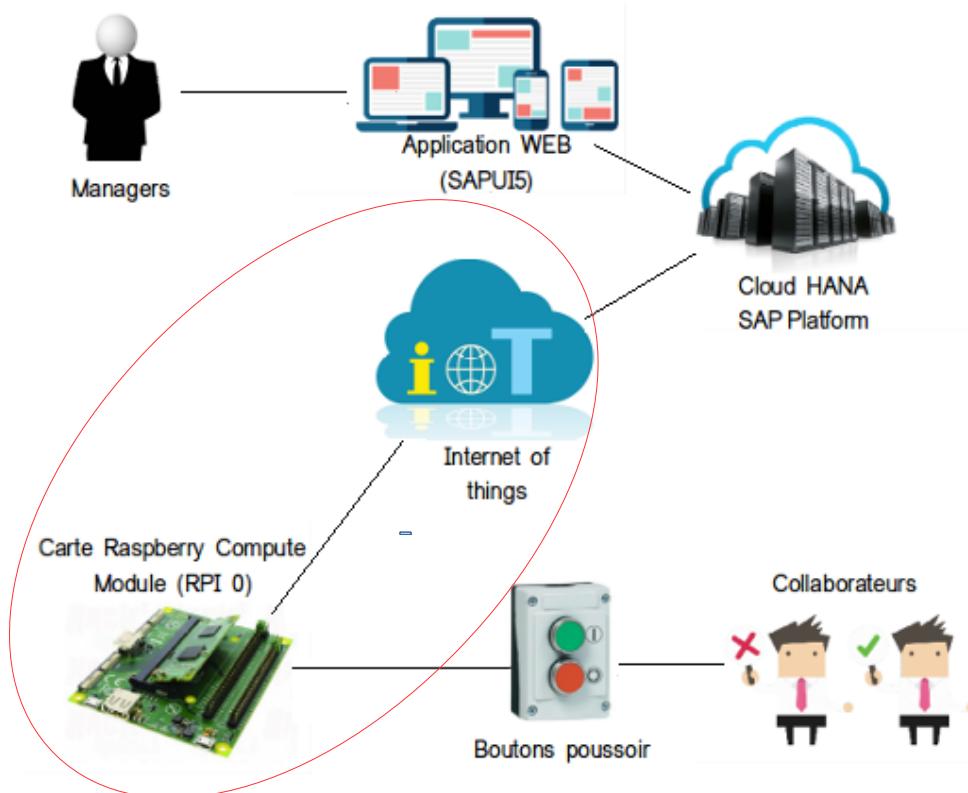
- $U_{\text{alim}} = 3,3\text{V}$ (tension en sortie de la RPI)
- $U_{\text{LED}} = 2\text{V}$ (moyenne pour les LEDs verte et jaune)
- $I = 10\text{ mA}$ (moyenne pour les LEDs verte et jaune)

Soit,

$$\begin{aligned} R &= (3,3 - 2) / 10 \times 10^{-3} \\ R &= 1,3 / 0,01 \\ R &= 130 \text{ Ohms} \end{aligned}$$

2.5 - Mise en place du point d'accès Wi-fi :

2.5.1 - Rappel du synoptique :



La mise en place d'un accès Wi-fi était essentielle pour le projet afin de pouvoir connecter le module RPI dépourvu de prise ethernet. En plus d'utiliser une clé Wi-Fi, il a donc fallu configurer un point d'accès permettant au module de se connecter à internet pour envoyer les différents votes.

Les différents paramètres de l'AP wi-fi sont :

- Adresse IP : 172.31.6.211
- Passerelle par défaut : 172.31.6.1
- Masque : 255.255.255.0
- ESSID : APGroupe5
- Mot de passe : Accesgroupe5

figure 13: Point d'accès Wifi : Cisco Aironet 1100

3 - Réalisation de la tâche Capter l'appui sur un bouton

3.1 - Cas d'utilisation :

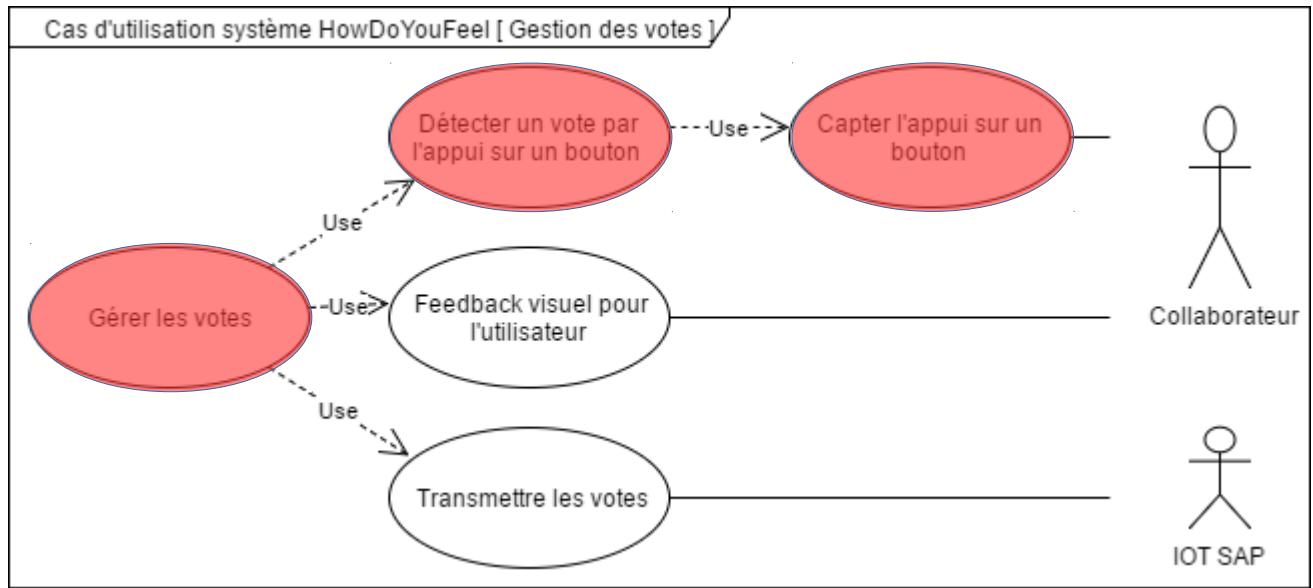


Figure 14: Cas d'utilisation : Capter l'appui sur un bouton

Le collaborateur à besoin d'une interface pratique pour voter rapidement. L'utilisation de deux boutons poussoirs pour les votes positifs et négatifs est donc le système idéal pour réaliser cette tâche. Pour comptabiliser le vote effectué par l'utilisateur il faut être en mesure de capter l'appui sur un des boutons poussoirs.

3.2 - Diagramme de déploiement :

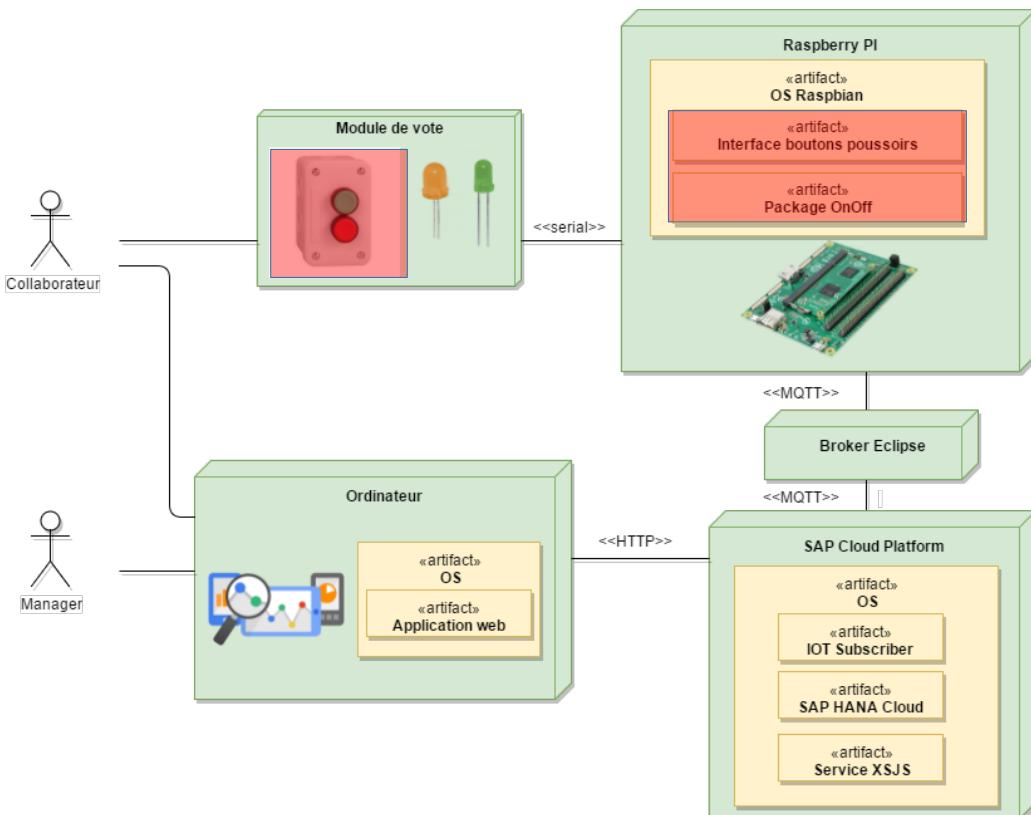


Figure 14: Diagramme de déploiement : Capter l'appui sur un bouton

3.3 - Conception détaillée :

Pour pouvoir envoyer un vote, l'utilisateur doit tout d'abord voter à l'aide des boutons poussoirs. Il y a alors deux choix qui s'offrent à lui, le vote positif (bouton vert) et le vote négatif (bouton rouge).

Pour réaliser le traitement des boutons poussoir au niveau logiciel sur nodeJS il a fallu utiliser le package OnOff qui permet de gérer tout les différents GPIO.

Dans un premier temps il faut donc initialiser les différents boutons poussoirs afin de pouvoir les utiliser :

```
var Gpio = require('onoff').Gpio,           //Appel du package "onoff"
buttonNo = new Gpio(7, 'in', 'rising');    //Initialisation du bouton non sur GPIO 7
buttonYes = new Gpio(0, 'in', 'rising');   //Initialisation du bouton oui sur GPIO 0
```

Voici la partie initialisation du programme. On peut voir que 2 boutons sont initialisés à savoir le buttonNo (bouton de vote négatif) et le buttonYes (bouton de vote positif) respectivement connectés au GPIO 7 et 0. L'élément « in » indique que ces deux GPIO sont des entrées et vont donc envoyer des informations vers la raspberry. L'élément « rising » signifie que seul le front montant sera pris en compte, à savoir, lorsque qu'un appui sera effectué, le changement de niveau passera de 0 à 1, puis de 1 à 0, dans notre cas, seul le passage de 0 à 1 sera pris en compte par le programme.

```
buttonNo.watch(function(err, value){          //Attente d'un changement d'état sur le bouton non
    ledgreen.writeSync(0);                    //LED Verte éteinte
    ledred.writeSync(1);                     //LED Jaune allumé
    buttonNo.unwatch();                     //Bouton non plus sous écoute
    buttonYes.unwatch();                   //Bouton oui plus sous écoute
    sendVote.send(0);                      //Envoie de vote
})
```

Le programme se sert donc du package pour « écouter » sur les boutons poussoirs et va réagir à chaque événement. Ici, le bouton non est sous écoute, et si l'état de ce bouton passe de 0 à 1 alors différentes actions sont effectuées jusqu'à l'envoi du vote.

3.4 - Diagramme de séquence :

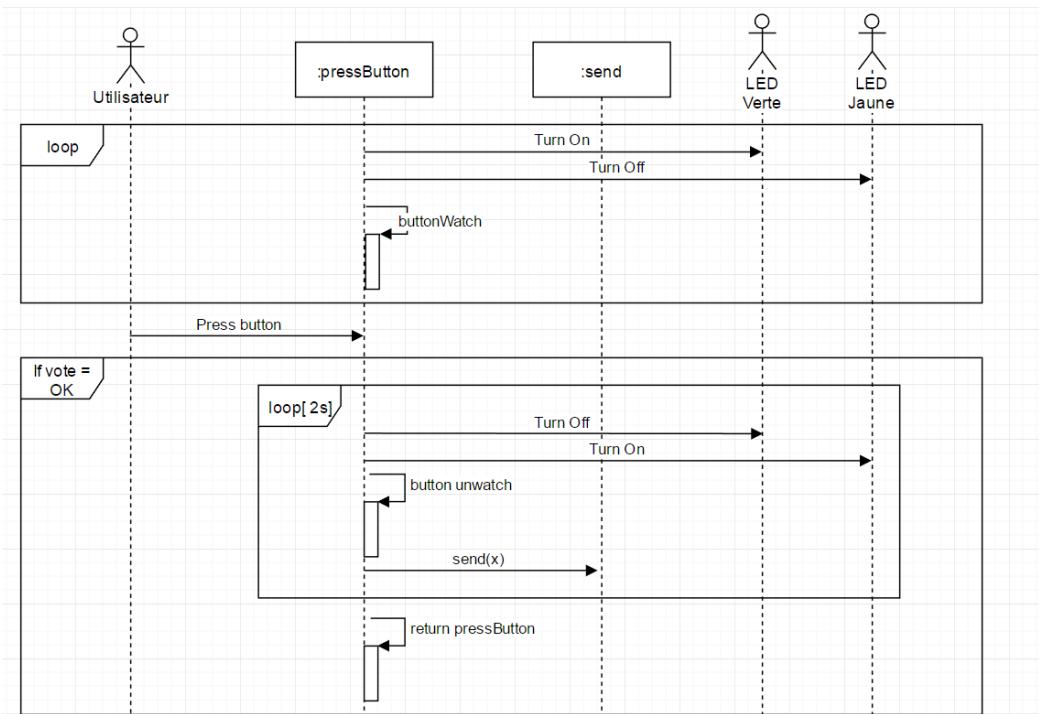


figure 15: Diagramme de séquence : Capter l'appui sur un bouton

Afin de capter l'appui sur un bouton, le programme va constamment écouter sur les deux boutons poussoirs reliés à la RPI. Si le programme capte l'appui sur l'un des deux boutons, il va alors se charger de prévenir l'utilisateur à l'aide des LEDs, puis couper l'écoute sur les deux boutons pour éviter de recevoir plusieurs votes dans les deux secondes qui suivent le premier vote, et enfin d'envoyer le vote. Au bout de deux secondes, la LED verte se rallumera et le vote sera de nouveau possible.

4 - Réalisation de la tâche transmettre le vote

4.1 - Cas d'utilisation :

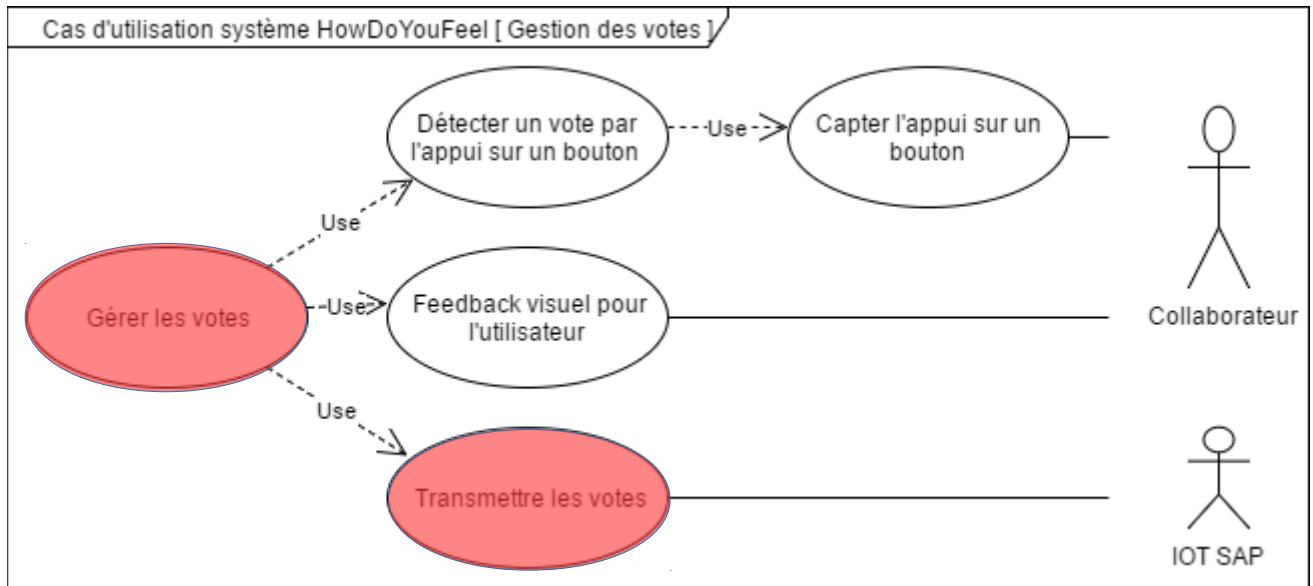


figure 16: Cas d'utilisation : Transmettre le vote

Après avoir capté l'appui sur un bouton, le système doit être en mesure de transmettre le vote vers l'IOT de SAP. Pour cela le système publiera le vote reçu sur un broker. L'IOT, de son côté, se chargera d'aller récupérer les votes sur le broker.

4.2 - Diagramme de déploiement :

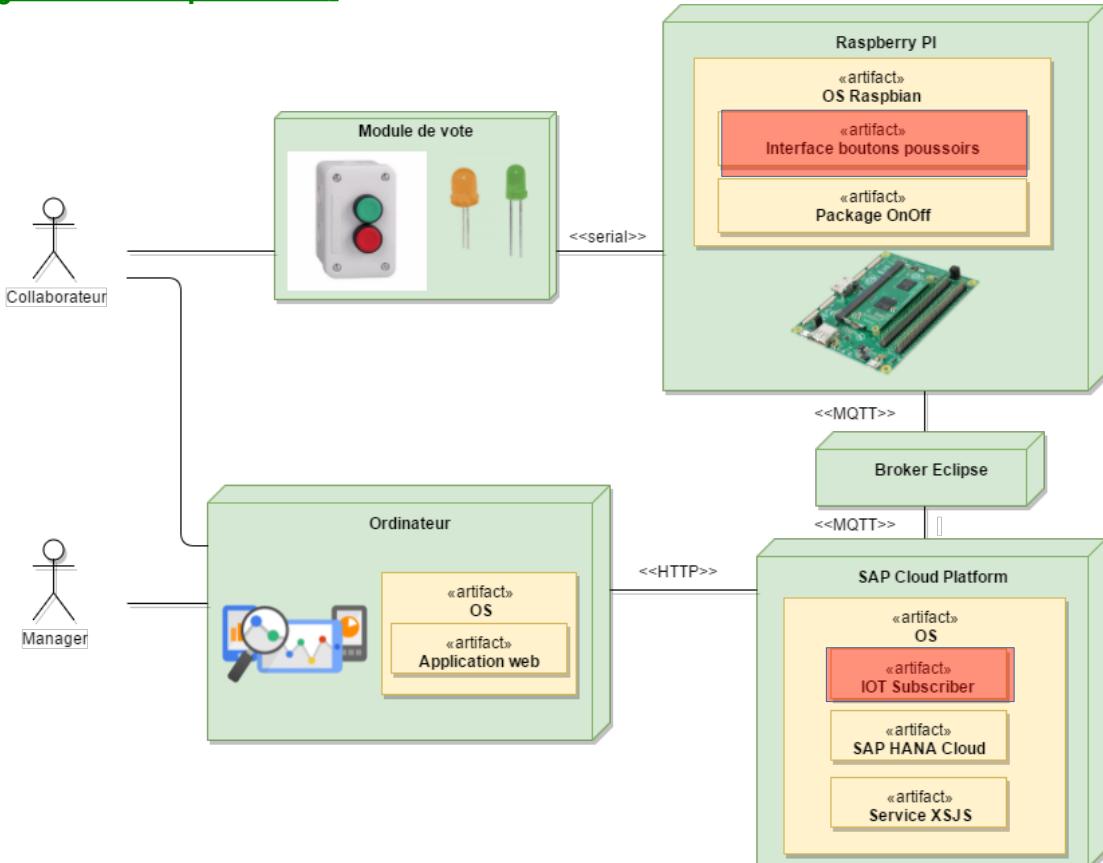


figure 17: Diagramme de déploiement : transmettre le vote

4.3 - Conception détaillée :

Lors de l'envoie d'un vote, ce dernier est publié sur un broker, une connexion vers un broker doit donc être effective.

```
var client = mqtt.connect('tcp://iot.eclipse.org', 1883)
client.on('connect', function() {
```

Figure 18 : Extrait code source : Connexion au Broker d'éclipse

Voici un extrait du code source qui illustre la connexion au broker Eclipse qui à pour adresse `iot.eclipse.org` et pour port 1883. Dans un premier temps on définit le client sur lequel on veut se connecter, puis, dans un second temps, on réalise la connexion.

La transmission du vote se fait de manière très précise et détaillée. Une publication doit être effectuée sous le modèle imposé par l'IOT de SAP et détaillé ci-dessous.

```
client.publish('iot/data/iotmmsp1942378810trial/v1/fbfd00eb-9561-4a6d-91df-dc1177ea7167',
  '{"mode":"async","messageType":"55222a0080cacc74b5f0","messages":[{"Vote":"1","ID_Project":"1"}]}')
```

Élément de la requête	Description
<code>iot/data/iotmmsp1942378810trial/v1/fbfd00eb-9561-4a6d-91df-dc1177ea7167</code>	En rouge : adresse du répertoire de publication au sein du broker (ici broker public d'eclipse), lié au compte SAP p1942378810trial En bleu : adresse du capteur actuel (cette adresse est générée par l'IOT SAP)
<code>"messageType":"55222a0080cacc74b5f0"</code>	En orange : Type de message, généré au moment de la configuration de l'IOT SAP. Il englobe donc les deux messages envoyés par vote.
<code>"messages":[{"Vote":"1","ID_Project":"44000"}]}</code>	En vert: Informations envoyées par vote, avec le détail du Vote (1/0 pour positif/négatif), et le détail de l'ID du projet, permettant de filtrer les résultats pour les différents projets.

4.4 - Diagramme de déploiement :

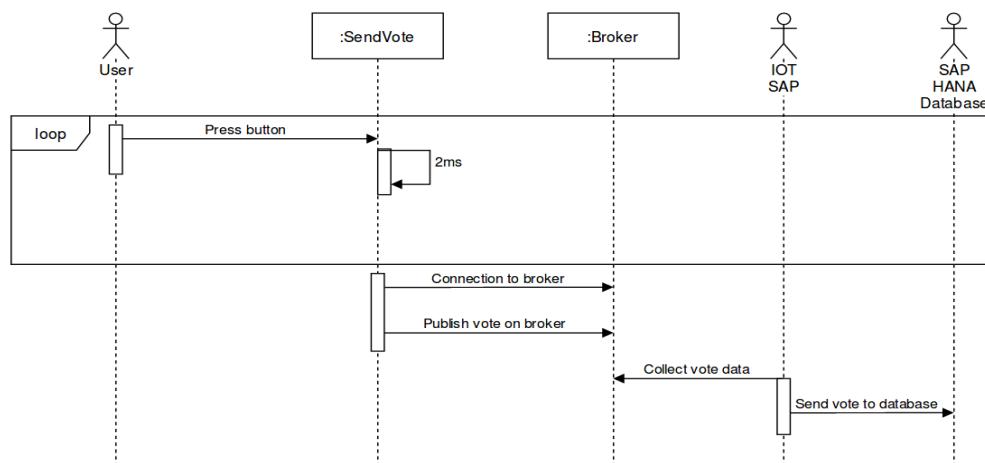


figure 19: Diagramme de séquence : Transmettre le vote

En cas de pression sur un des boutons poussoirs, le programme va envoyer un vote vers la base de données. Pour cela, il va en premier temps se connecter sur le broker (ici le broker public d'éclipse), ensuite il va publier au sein du répertoire donné de ce broker les données à envoyer. De l'autre côté, c'est l'IOT SAP qui est abonné au même répertoire qui va se charger de récupérer les votes envoyés et des les transmettre dans la base de données.

5 - Réalisation du feedback pour l'utilisateur

5.1 - Cas d'utilisation :

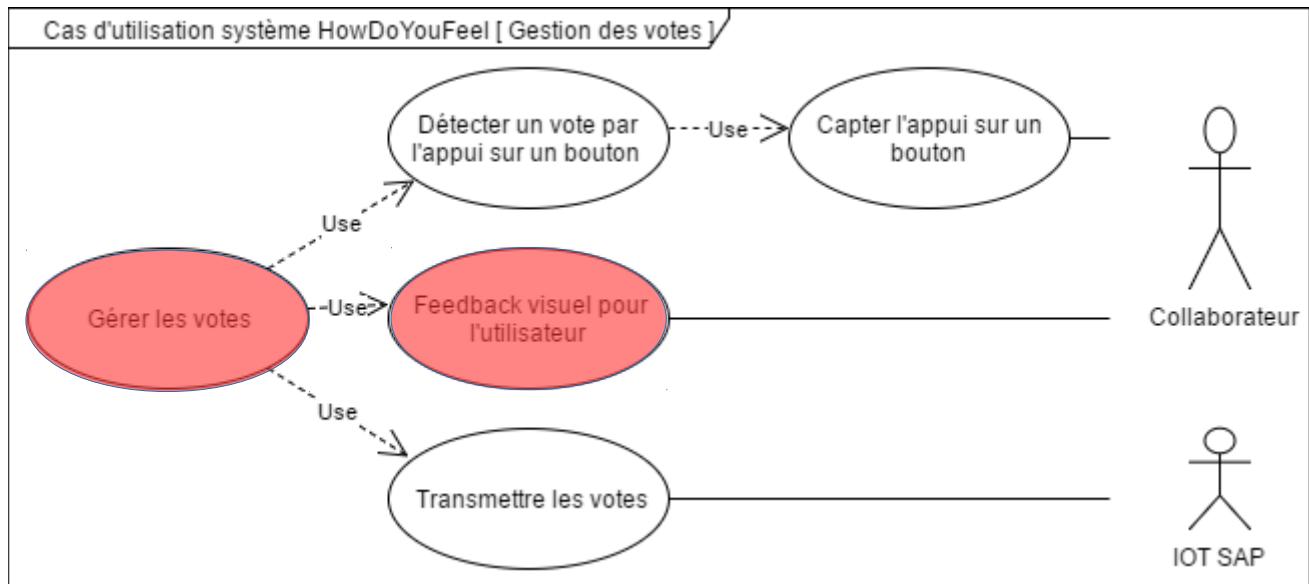


figure 20: Diagramme d'utilisation : feedback pour l'utilisateur

5.2 - Diagramme de déploiement :

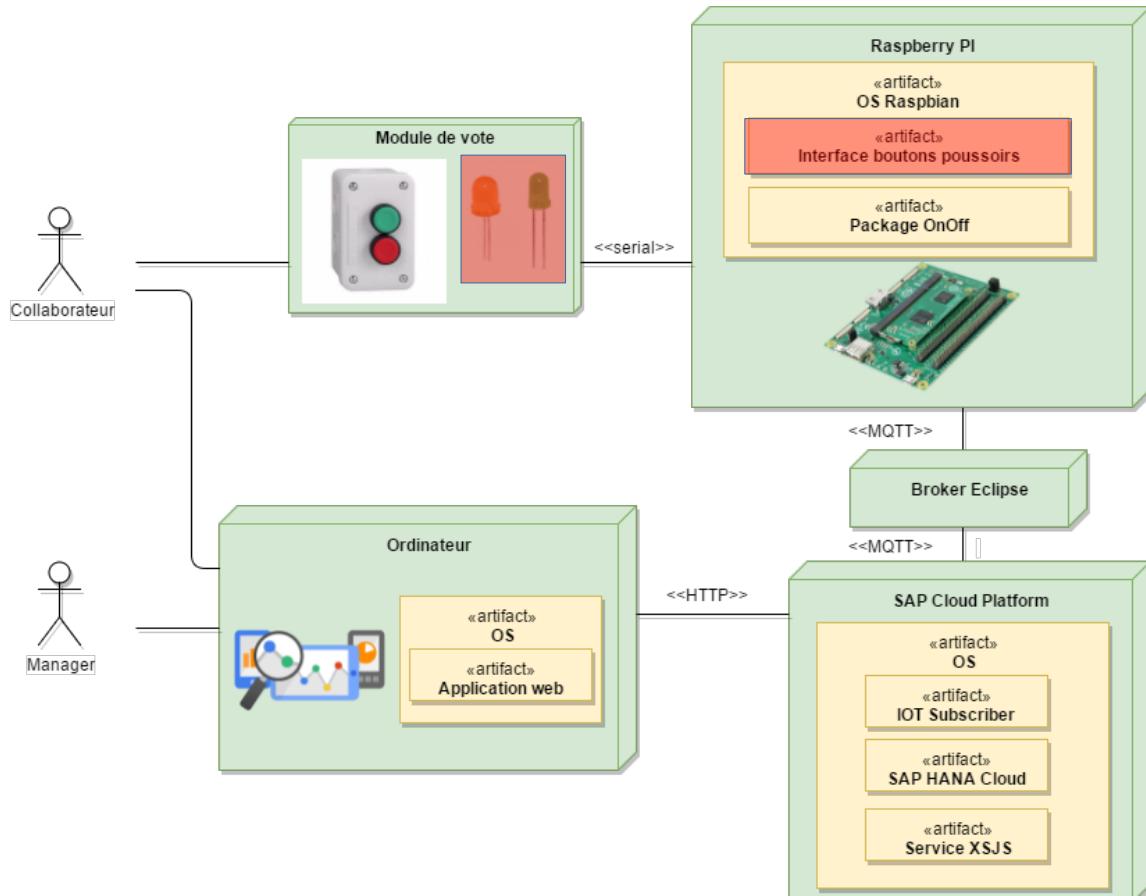


figure 21: Diagramme de déploiement: feedback pour l'utilisateur

5.3 - Conception détaillée :

Pour que les résultats des votes ne soient pas faussés, l'utilisateur ne pourra pas voter plusieurs fois à la suite. Il y aura donc un blocage de réalisé au niveau logiciel. Mais le plus important est de signaler à l'utilisateur quand il a la possibilité de réaliser son vote. Pour cela, 2 LEDs de couleur jaune et verte ont été mises en œuvre afin de réaliser un retour visuel. Il y a donc deux états possibles :

Couleur LED :	Etat :	Etat :	Vote :
LED Verte	Allumée	Éteinte	OK
LED Jaune	Éteinte	Allumée	KO

Au niveau logiciel, ce retour utilisateur utilise le même package « OnOff » que pour l'appui sur le bouton.

```
ledgreen = new Gpio(8, 'out');           //Initialisation de la LED rouge sur GPIO 8
ledred = new Gpio(9, 'out');             //Initialisation de la LED jaune sur GPIO 9
```

Après l'appel du package, les LEDs sont initialisées sur les ports GPIO 8 et 9 et sont configurées comme des sorties, « out ».

Suite à l'initialisation des LEDs, la LED Verte est automatiquement allumée en début de programme pour indiquer que le vote est possible, la LED jaune est éteinte.

```
ledred.writeSync(0);
ledgreen.writeSync(1);
```

Enfin, lors de l'appui sur un bouton poussoir, la LED jaune s'allume tandis que la LED verte s'éteint le temps que les boutons poussoirs soient bloqués. Le vote est donc impossible et l'utilisateur en est prévenu. Au bout des 2 secondes de blocage, le programme retour à son état initial, l'état des LED, y compris, le vote est de nouveau possible.

```
buttonNo.watch(function(err, value){           //Attente d'un changement d'état sur le bouton
    ledgreen.writeSync(0);                     //LED Verte éteinte
    ledred.writeSync(1);                      //LED Jaune allumé
    buttonNo.unwatch();                      //Bouton non plus sous écoute
    buttonYes.unwatch();                     //Bouton oui plus sous écoute
    sendVote.send(0);                        //Envoie de vote
    setTimeout(function() {
        return sendVote.pressButton();       //retour à l'état initial
    }, 2000);
}
```

5.4 - Diagramme de séquence :

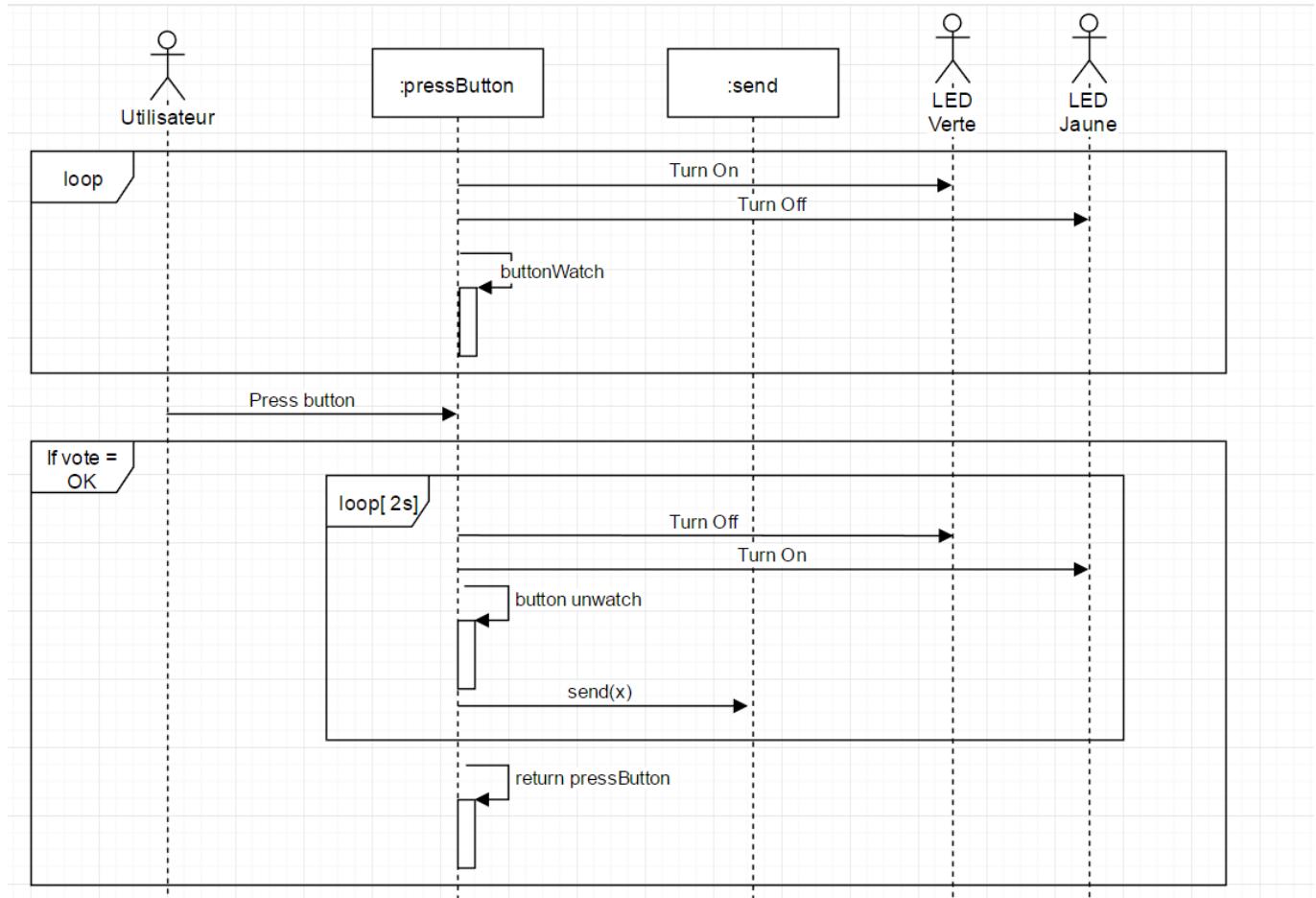


figure 22: Diagramme de séquence : FeedBack Utilisateur

Pour que le vote soit le plus simple possible, un retour pour l'utilisateur est nécessaire. C'est donc via deux LEDs de couleur jaune et verte que le feedback sera effectué. Lorsque le vote est possible, la LED de couleur verte est allumée, la jaune est éteinte. Pendant les deux secondes suivant le vote, de nouveaux votes ne peuvent pas être envoyés. L'utilisateur est donc prévenu à l'aide de la LED verte éteinte et de la LED jaune qui s'allume.

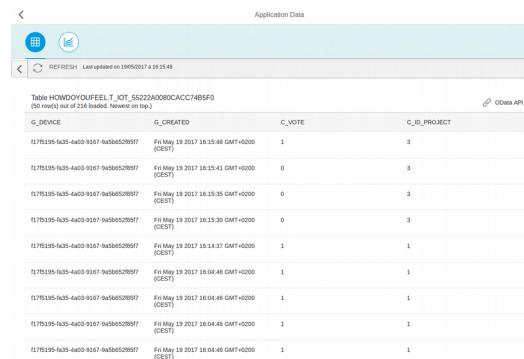
TEST UNITAIRES

6 - Plans des tests unitaires

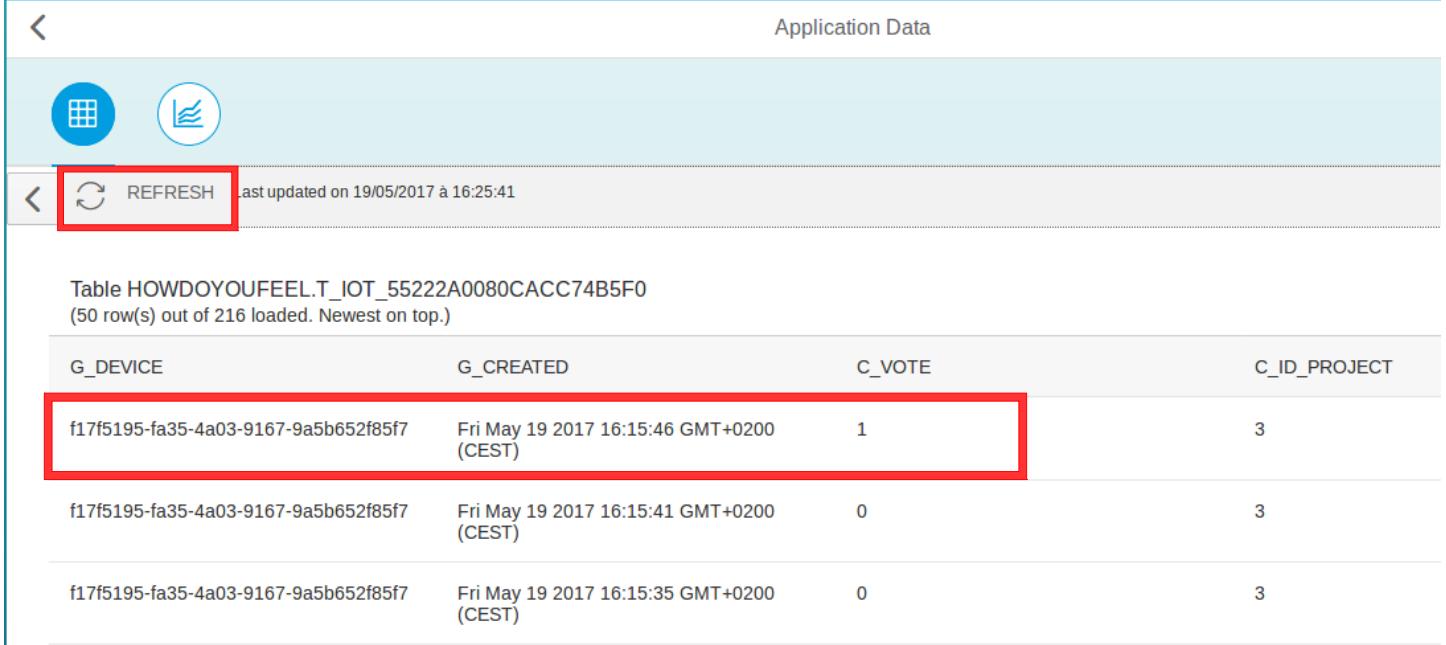
- Test Global
- Test d'appui sur le bouton poussoir
- Test de transmission du vote
- Test du feedback utilisateur

7 - Test de la gestion des votes

7.1 - Procédure de test :

• Id.	<ul style="list-style-type: none"> • Etape à réaliser • Description Sommaire 	• Procédure de test
		• Résultats attendus
• U1.0	ssh root@wrpi5.local	Etablir la connexion ssh vers la RPI
		Connexion à la rpi
• U1.1	Lancement du test vote.js	<p>Lancement du programme de gestion des votes se trouvant de le répertoire vote</p> <ul style="list-style-type: none"> - cd /home/pi/vote - node vote.js <p>Le programme se lance</p>
• U1.2	Se rendre sur l'interface de l'IOT permettant de voir les votes et analyser les votes	<p>https://iotmmmsp1942378810trial.hanatrial.ondemand.com/com.sap.iotservices.mms/#/appdata/HOWDOYOUFEEL/T_IOT_55222A0080CACC74B5F0</p>  <p>Regarder la date du dernier vote envoyé</p>
• U1.3	Envoyer un vote positif ou négatif	Appuyer sur un des boutons poussoir pour envoyer un vote et relever l'heure de cet envoie
		<pre>root@wrpi5:/home/pi/vote# node vote.js Vote 0 send</pre> <p>La console confirme l'envoie du vote.</p>
• U1.4	Retourner sur l'interface de l'IOT et actualiser les votes, comparer avec l'heure d'envoie	<p>https://iotmmmsp1942378810trial.hanatrial.ondemand.com/com.sap.iotservices.mms/#/appdata/HOWDOYOUFEEL/T_IOT_55222A0080CACC74B5F0</p> <p>CF U1.4 IOT</p>

7.1.1 - U1.4 IOT :



Application Data

Table HOWDOYOUFEEL.T_IOT_55222A0080CACC74B5F0
(50 row(s) out of 216 loaded. Newest on top.)

G_DEVICE	G_CREATED	C_VOTE	C_ID_PROJECT
f17f5195-fa35-4a03-9167-9a5b652f85f7	Fri May 19 2017 16:15:46 GMT+0200 (CEST)	1	3
f17f5195-fa35-4a03-9167-9a5b652f85f7	Fri May 19 2017 16:15:41 GMT+0200 (CEST)	0	3
f17f5195-fa35-4a03-9167-9a5b652f85f7	Fri May 19 2017 16:15:35 GMT+0200 (CEST)	0	3

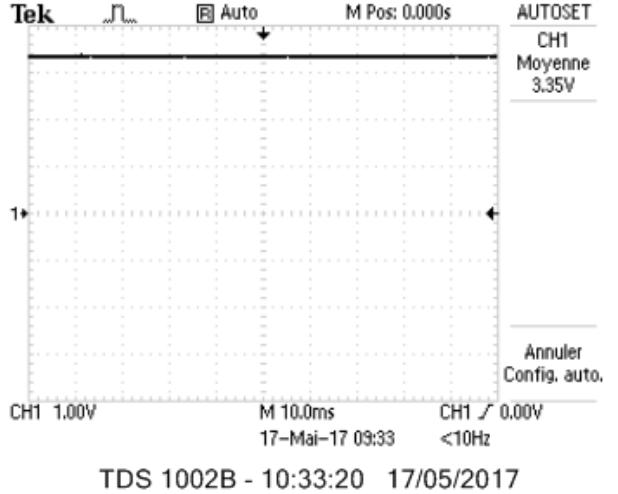
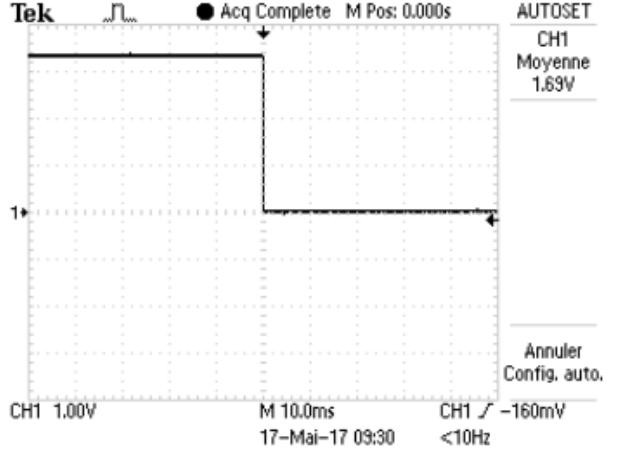
Dans un premier temps il faut cliquer sur l'icone refresh. Ensuite nous comparerons l'heure d'envoi du vote à son heure de réception par l'IOT.

7.2 - Rapport d'exécution :

• Id.	OK	!KO	Observations
• U1.0	✓		Résultats OK
• U1.1	✓		Résultats OK
• U1.2	✓		Résultats OK
• U1.3	✓		Résultats OK
• U1.4	✓		Résultats OK

8 - Test de l'appui sur le bouton poussoir

8.1 - Procédure de test :

• Id.	• Etapes à réaliser • Description Sommaire	• Procédure de test • Résultats attendus
• U2.0	ssh root@wrpi5.local	Établir la connexion ssh vers la RPI Connexion à la rpi
• U2.1	cd /home/pi node testAppui.js	Lancement du programme de test d'appui Le programme se lance et attend l'appui d'un bouton
• U2.2	Branchemet et initialisation de l'oscilloscope	Branchemet en dérivation de l'oscilloscope aux bornes du bouton poussoir et de la résistance et initialisation à l'aide du bouton « auto set » de l'oscilloscope. Une tension d'environ 3,3 volts est affichée sur l'oscilloscope.  TDS 1002B - 10:33:20 17/05/2017
• U2.3	Appui sur le bouton poussoir	Appui sur le bouton sur lequel est branché l'oscilloscope La tension chute à environ 1,69 volts lors de l'appui sur un bouton. Le programme de test renvoi « Un appui a été effectué ».  TDS 1002B - 10:30:49 17/05/2017

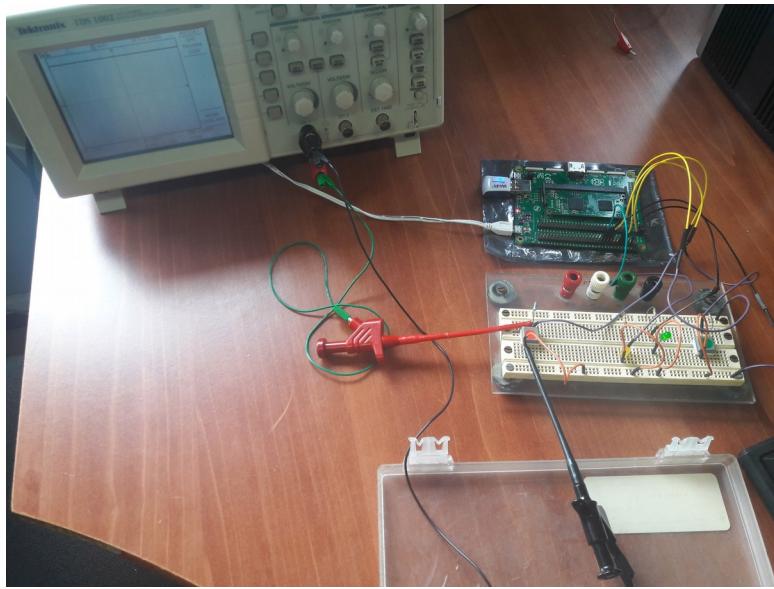


figure 23: Branchement de l'oscilloscope

8.2 - Code source du test :

```
1  var testAppui = {
2      pressButton: function(){
3
4          var Gpio = require('onoff').Gpio,           //Appel du package "Onoff"
5              buttonNo = new Gpio(7, 'in', 'rising'); //Initialisation du bouton "no"
6              buttonYes = new Gpio(0, 'in', 'rising'); //Initialisation du bouton "yes"
7
8          buttonNo.watch(function(err, value){ //Ecoute et attente d'événements sur le bouton "no"
9              buttonNo.unwatch(); //Mise en pause de toute les écoutes
10             buttonYes.unwatch(); //
11             console.log('Un appui a été effectué');
12             setTimeout(function() {
13                 return testAppui.pressButton();
14             }, 1000);
15         });
16
17         buttonYes.watch(function(err, value){ //Ecoute et attente d'événements sur le bouton "yes"
18             buttonNo.unwatch(); //Mise en pause de toute les écoutes
19             buttonYes.unwatch();//
20             console.log('Un appui a été effectué');
21             setTimeout(function() {
22                 return testAppui.pressButton();
23             }, 1000);
24         });
25     },
26
27
28     init: function() {
29         testAppui.pressButton();
30     }
31
32 }
33
34 testAppui.init();
```

8.3 - Rapport d'exécution :

• Id.	OK	!KO	Observations
• U2.0	V		Résultats OK
• U2.1	V		Résultats OK
• U2.2	V		Résultats OK
• U2.3	V		Résultats OK

9 - Test de la transmission du vote

9.1 - Procédure de test :

Pour réaliser ce test une condition initiale est nécessaire :

- Un PC portable configuré en mode « monitor ». Le mode monitor permet à un ordinateur équipé d'une carte réseau Wi-Fi d'écouter tout le trafic d'un réseau sans fil. Il s'agira donc d'écouter le trafic allant vers l'AP Wifi pour retrouver les paquets de vote envoyés.

Pour réaliser le passage en mode monitor, il faut réaliser ces commandes en mode administrateur :

```
# service network-manager stop
# ip link set dev wlp2s0 down
# iwconfig wlp2s0 mode monitor
# ip link set dev wlp2s0 up
```

• Id.	<ul style="list-style-type: none"> • Etape à réaliser • Description Sommaire 	• Procédure de test
		• Résultats attendus
• U3.0	ssh root@wrpi5.local (sur un ordinateur du réseau)	<p>Établir la connexion ssh vers la RPI</p> <p>Connexion à la rpi</p>
• U3.1	Lancement du test	<p>Lancement du programme testTransmission.js se trouvant dans le répertoire /Document/projet5/test</p> <ul style="list-style-type: none"> - cd /home/pi - node testTransmission.js <p>Le programme se lance</p>
• U3.2	Lancer wireshark sur le PC Portable configuré en mode monitor	<p>Lancer l'application wireshark en mode administrateur depuis une console à l'aide de la commande « sudo wireshark ». Sélectionner l'interface wlp2s0.</p> <p>L'application se lance et l'ont aperçoit tout les différents paquets qui transitingent sur les réseaux Wifi alentours</p>
• U3.3	Appliquer le filtre part adresse IP pour ne récupérer que les paquets intéressants	<p>Appliquer le filtre pour ne voir que les paquets ayant pour source l'adresse MAC de la carte raspberry (00:c1:41:33:0b:0a)</p> <ul style="list-style-type: none"> - wlan.sa == 00:c1:41:33:0b:0a <p>CF Image : Wireshark U2.3</p>
• U3.4	Envoyer un vote depuis le programme testTransmission.js	<p>Envoyer un vote en appuyant sur le bouton vert</p> <p>Un vote positif est envoyé et la console renvoi :</p> <div style="border: 1px solid black; padding: 2px;">Vote 1 send</div> <p>Sur wireshark, un paquet de poids 318 octets est transmis et correspond à l'envoie du vote :</p> <p>CF Image : Wireshark U2.4</p>

9.1.1 - U2.3 : Wireshark :

No.	Time	Source	Destination	Protocol	Length	Info
1891	8.139967816	00:c1:41:33:0b:0a	CiscoInc_77:d1:10	802.11	64	Null function (No data), SN=434, FN=0, Flags=.....TC
3105	21.674072595	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	148	QoS Data, SN=1726, FN=0, Flags=p.....TC
3116	21.899767352	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	146	QoS Data, SN=1727, FN=0, Flags=p.R..TC
3732	26.988314837	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	148	QoS Data, SN=1728, FN=0, Flags=p.....TC
3741	27.159842133	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	146	QoS Data, SN=1729, FN=0, Flags=p.....TC
4680	37.851410504	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	148	QoS Data, SN=1730, FN=0, Flags=p.....TC
4698	38.019807159	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	146	QoS Data, SN=1731, FN=0, Flags=p.....TC
8098	68.140013911	00:c1:41:33:0b:0a	CiscoInc_77:d1:10	802.11	64	Null function (No data), SN=435, FN=0, Flags=.....TC
9432	81.713457096	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	148	QoS Data, SN=1732, FN=0, Flags=p.....TC
9441	81.852534619	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	146	QoS Data, SN=1733, FN=0, Flags=p.....TC
9909	86.992980008	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	148	QoS Data, SN=1734, FN=0, Flags=p.....TC
9909	87.128648598	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	146	QoS Data, SN=1735, FN=0, Flags=p.....TC
10932	97.863761060	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	148	QoS Data, SN=1736, FN=0, Flags=p.....TC
10947	97.997808312	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	146	QoS Data, SN=1737, FN=0, Flags=p.....TC
14147	128.140026377	00:c1:41:33:0b:0a	CiscoInc_77:d1:10	802.11	64	Null function (No data), SN=436, FN=0, Flags=.....TC

9.1.2 - U2.4 : Wireshark :

No.	Time	Source	Destination	Protocol	Length	Info
864	6.748576048	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	146	QoS Data, SN=2021, FN=0, Flags=p.R..TC
980	7.749124607	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	148	QoS Data, SN=2022, FN=0, Flags=p.....TC
992	7.886512708	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	146	QoS Data, SN=2023, FN=0, Flags=p.....TC
1130	8.994119104	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	148	QoS Data, SN=2024, FN=0, Flags=p.....TC
1165	9.135413054	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	146	QoS Data, SN=2025, FN=0, Flags=p.....TC
1624	11.615436085	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	148	QoS Data, SN=2026, FN=0, Flags=p.....TC
1632	11.751109721	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	146	QoS Data, SN=2027, FN=0, Flags=p.....TC
1821	13.206806812	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	148	QoS Data, SN=2028, FN=0, Flags=p.....TC
1832	13.347446762	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	146	QoS Data, SN=2029, FN=0, Flags=p.....TC
2674	19.144150236	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	148	QoS Data, SN=2030, FN=0, Flags=p.....TC
2710	19.281221769	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	146	QoS Data, SN=2031, FN=0, Flags=p.....TC
3002	22.379986057	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	155	QoS Data, SN=2032, FN=0, Flags=p.....TC
3003	22.38003625	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	155	QoS Data, SN=2032, FN=0, Flags=p.R..TC
3009	22.383263600	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	155	QoS Data, SN=2033, FN=0, Flags=p.....TC
3010	22.383278408	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	155	QoS Data, SN=2033, FN=0, Flags=p.R..TC
3015	22.390172448	00:c1:41:33:0b:0a	HewlettP_ed:66:a0	802.11	198	QoS Data, SN=2034, FN=0, Flags=p.....TC
3019	22.392015450	00:c1:41:33:0b:0a	HewlettP_ed:66:a0	802.11	182	QoS Data, SN=2035, FN=0, Flags=p.....TC
3020	22.392028979	00:c1:41:33:0b:0a	HewlettP_ed:66:a0	802.11	182	QoS Data, SN=2035, FN=0, Flags=p.R..TC
3021	22.392034089	00:c1:41:33:0b:0a	HewlettP_ed:66:a0	802.11	182	QoS Data, SN=2035, FN=0, Flags=p.R..TC
3025	22.395201515	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	154	QoS Data, SN=2036, FN=0, Flags=p.....TC
3026	22.395217380	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	154	QoS Data, SN=2036, FN=0, Flags=p.R..TC
3027	22.395222704	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	154	QoS Data, SN=2036, FN=0, Flags=p.R..TC
3038	22.530628616	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	146	QoS Data, SN=2037, FN=0, Flags=p.....TC
3039	22.530644753	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	146	QoS Data, SN=2037, FN=0, Flags=p.R..TC
3040	22.531977551	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	175	QoS Data, SN=2038, FN=0, Flags=p.....TC
3048	22.667382864	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	146	QoS Data, SN=2039, FN=0, Flags=p.....TC
3050	22.673615079	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	318	QoS Data, SN=2040, FN=0, Flags=p.....TC
3171	24.258721877	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	148	QoS Data, SN=2041, FN=0, Flags=p.....TC
3172	24.258740052	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	148	QoS Data, SN=2041, FN=0, Flags=p.R..TC
3178	24.394143969	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	146	QoS Data, SN=2042, FN=0, Flags=p.....TC
3179	24.394161265	00:c1:41:33:0b:0a	CiscoInc_f4:aa:46	802.11	146	QoS Data, SN=2042, FN=0, Flags=p.R..TC

9.2 - Code source du test :

```

1  var stdin = process.stdin;
2
3  var testTransmission = {
4      pressButton: function() {
5
6          var Gpio = require('onoff').Gpio,           //Appel du package "Onoff"
7              buttonYes = new Gpio(0, 'in', 'rising'); //Initialisation du bouton "yes"
8
9          buttonYes.watch(function(err, value){ //Ecoute et attente d'événements sur le bouton "yes"
10             buttonYes.unwatch(); //Mise en pause de l'écoute sur le bouton
11             testTransmission.send(1); // Envoie d'un vote positif
12             setTimeout(function() {
13                 return testTransmission.pressButton();
14             }, 1000);
15         });
16     },
17
18     send: function(value) {
19         var mqtt = require('mqtt') //Appel du package "mqtt"
20         var client = mqtt.connect('tcp://iot.eclipse.org', 1883) //Connexion au broker public d'eclipse
21         client.on('connect', function() {
22             client.publish('iot/data/iotmmsp1942378810trial/v1/f17f5195-fa35-4a03-9167-9a5b652f85f7',
23             '{"mode":"async","messageType":"55222a0080cacc74b5f0","messages":[{"Vote":"' + value + '","ID_Project":"1"}]}')
24         });
25
26         console.log('Vote ' + value + ' envoyé');
27     },
28
29     init: function() {
30         testTransmission.pressButton();
31     }
32 }
33
34 testTransmission.init();
35

```

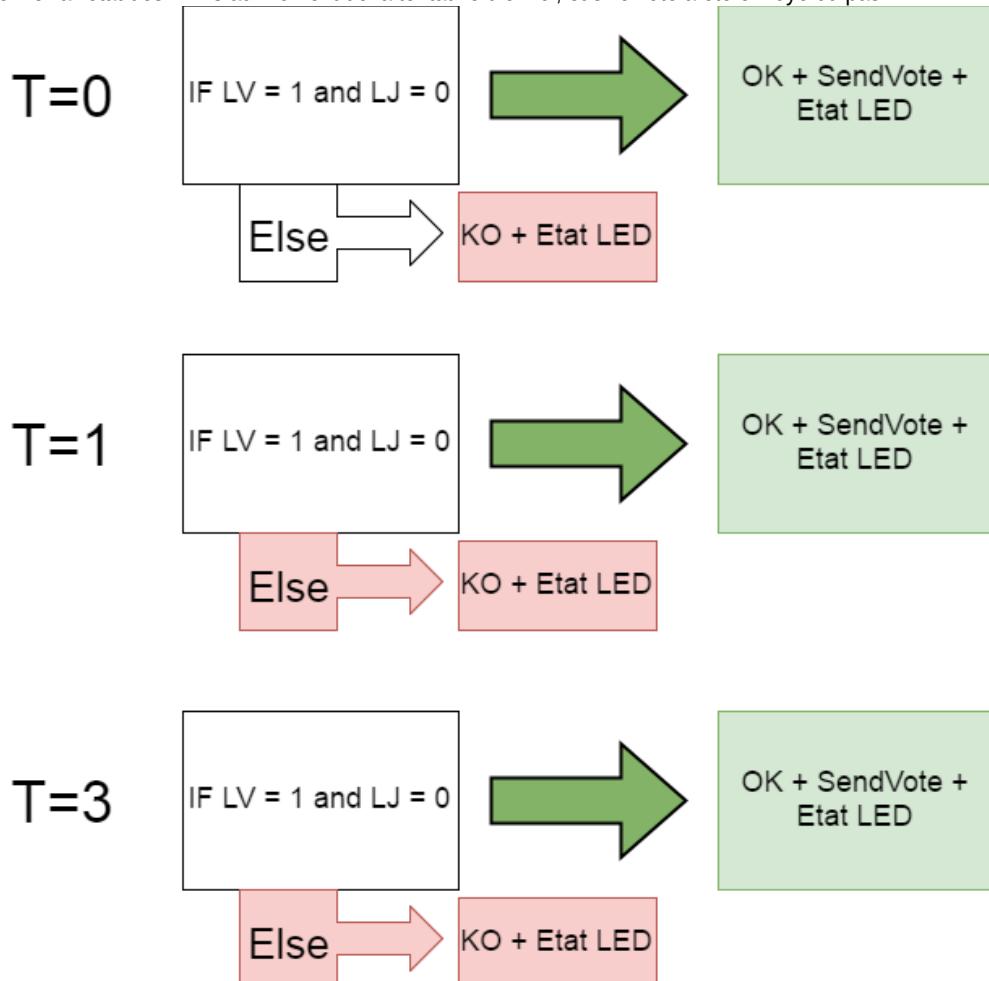
9.3 - Rapport d'exécution :

• Id.	OK	!KO	Observations
• U3.0	V		Résultat OK
• U3.1	V		Résultat OK
• U3.2	V		Résultat OK
• U3.3	V		Résultat OK
• U3.4	V		Résultat OK

10 - Test du feedback utilisateur

10.1 - Description du test :

Ce test est constitué de 3 parties qui vont se suivre. En effet, lorsque qu'un vote est envoyé, un délai de 2 secondes doit être effectué avant l'envoi d'un autre vote. Ces 2 secondes sont caractérisées par l'allumage de la LED jaune. Ce test va avoir pour but d'envoyer 3 votes à 3 moments différents, à savoir T=0 (premier vote), T=1 (une seconde après) et T=3 (trois secondes après). Chacun de ces trois tests renverra l'état des LEDs au moment de la tentative d'envoi, et si le vote a été envoyé ou pas.



Ici LV = LED Verte et LJ = LED Jaune

Les résultats attendus sont donc :

T=0	OK
T=1	KO
T=3	OK

10.2 - Procédure de test :

• Id.	• Etapes à réaliser • Description Sommaire	• Procédure de test • Résultats attendus
• U4.0	ssh root@wrpi5.local	<p>Établir la connexion ssh vers la RPI</p> <p>Prompt rpi affiché</p>
• U4.1	cd /home/pi node testFeedback.js	<p>Se rendre dans le répertoire du test et le lancer</p> <p>Le programme se lance et va réaliser 3 tentatives d'envoi : Un premier envoi de vote qui doit réussir : OK + Etat des LEDs + Envoi du vote</p> <p>Un second envoi de vote instantané qui doit échouer : KO + Etat des LEDs + vote non envoyée</p> <p>Un troisième après 3 secondes de délais qui doit réussir : OK + Etat des LEDs + Envoi du vote</p> <pre>root@wrpi5:/home/pi# node testFeedback.js Test 1 : Premier test OK Etat LED verte 1 Etat LED jaune 0 ***** Vote 0 send ***** Test 2 : 1 seconde après KO Etat LED verte 0 Etat LED jaune 1 ***** Vote not send ***** Test 3 : après 3s OK Etat LED verte 1 Etat LED jaune 0 ***** Vote 0 send *****</pre>

10.3 - Code source du test :

```

1 var SendVote = require('./SendVote.js');
2 var testFeedback = {
3     testPressButton: function() {
4         console.log("Test 1 : Premier test");
5         SendVote.readButtons();
6         if(SendVote.ledgreen.readSync() == 1 && SendVote.ledred.readSync() == 0) {
7             console.log("OK");
8             console.log("Etat LED verte " + SendVote.ledgreen.readSync());
9             console.log("Etat LED jaune " + SendVote.ledred.readSync());
10            console.log("*****");
11            SendVote.pressButton(0);
12            console.log("*****");
13        } else {
14            console.log("KO");
15            console.log("Etat LED verte " + SendVote.ledgreen.readSync());
16            console.log("Etat LED jaune " + SendVote.ledred.readSync());
17            console.log("*****");
18            console.log("Vote not send");
19            console.log("*****");
20        }
21
22
23
24     setTimeout(function() {
25         return testFeedback.testPressButtonAfter1S();
26     }, 1000);
27 },
28
29
30     testPressButtonAfter1S: function() {
31         console.log("Test 2 : 1 seconde après");
32         if(SendVote.ledgreen.readSync() == 1 && SendVote.ledred.readSync() == 0) {
33             console.log("OK");
34             console.log("Etat LED verte " + SendVote.ledgreen.readSync());
35             console.log("Etat LED jaune " + SendVote.ledred.readSync());
36             console.log("*****");
37             SendVote.pressButton(0);
38             console.log("*****");
39        } else {
40            console.log("KO");
41            console.log("Etat LED verte " + SendVote.ledgreen.readSync());
42            console.log("Etat LED jaune " + SendVote.ledred.readSync());
43            console.log("*****");
44            console.log("Vote not send");
45            console.log("*****");
46        }
47
48
49     setTimeout(function() {
50         return testFeedback.testPressButtonAfter3S();
51     }, 3000);
52 },
53
54     testPressButtonAfter3S: function() {
55         console.log("Test 3 : après 3s");
56         if(SendVote.ledgreen.readSync() == 1 && SendVote.ledred.readSync() == 0) {
57             console.log("OK");
58             console.log("Etat LED verte " + SendVote.ledgreen.readSync());
59             console.log("Etat LED jaune " + SendVote.ledred.readSync());
60             console.log("*****");
61             SendVote.pressButton(0);
62             console.log("*****");
63        } else {
64            console.log("KO");
65            console.log("Etat LED verte " + SendVote.ledgreen.readSync());
66            console.log("Etat LED jaune " + SendVote.ledred.readSync());
67            console.log("*****");
68            console.log("Vote not send");
69            console.log("*****");
70        }
71    },
72
73    init: function() {
74        testFeedback.testPressButton();
75    }
76}
77

```

10.4 - Rapport d'exécution :

• Id.	OK	!KO	Observations
• U4.0	V		Résultats OK
• U4.1	V		Résultats OK

11 - Bilan de la réalisation personnelle

11.1 - Statut des fonctions à charges :

Application embarquée	Bouton Pousoir	Choisir un système à Boutons Pousoirs	Terminé
	Bouton Pousoir	Mettre en œuvre le système à Boutons Pousoirs	Terminé
	Bouton Pousoir	DéTECTer les événements du système à Boutons Pousoirs	Terminé
	Bouton Pousoir	Filtrer les événements du système à Boutons Pousoirs	Terminé
	Bouton Pousoir	Stocker localement l'événement du système à Boutons Pousoirs	
	Communiquer	Choisir le protocole de communication avec le Cloud	Terminé
	Communiquer	Mettre en œuvre une communication avec le Cloud	Terminé
	Communiquer	Communiquer les votes au Cloud	Terminé
	Date	Dater le vote	Effectué par l'IOT
	Gérer	Ordonnancer les votes	Terminé

11.2 - Conclusion :

11.2.1 - Points négatifs

Compte trial restreint et SAP difficile à prendre en main au vu du nombre de services.
Difficulté et manque de documentation pour la configuration de l'IOT de SAP.

11.2.2 - Points Positifs

Richesse de la gestion et du travail de groupe
De nombreuses découvertes notamment autour de SAP et de l'IOT.
Application des méthodes agiles
Découverte d'un nouveau type de raspberry