

RAPPORT DE PROJET PROJET T'REX

*PALLY Julien - LAUNAY Maria - PAWLAK-BAGORSKI Antoine -
KOUADIO Olivier*



SESAM 2018/2019

RESPONSABLES DE LICENCE : Xavier Mininger & Nathalie
BRISSARD

TUTEUR DU PROJET : PATRICK RUIZ

Table des matières

1. Présentation du projet	3
2. Travail effectué sur le robot.....	4
2.1 Pilotage du robot	4
2.2 CAO carte extension + Support	5
3. Mise en place de la VR et du réseau local	8
3.1 Manette	8
3.1.1 Gestions des trames de la manette	8
3.1.2 Envoi de données en Wi-Fi	8
3.2 Casque.....	10
3.2.1 Le streaming vidéo de la caméra	10
3.2.2 Affichage de la vidéo dans l'environnement VR	11
4.En cours de développement.....	14
4.1 I ² C et Wi-Fi avec la STM32.....	14
4.2 Pilotage du robot par Bluetooth	14
4.3 Carte d'extension STM32	14
4.4 Amélioration de la qualité du streaming	15
Conclusion	15
Tables des Figures	15
Annexe	16

1. Présentation du projet

L'objectif principal du projet est de pouvoir piloter un robot à partir d'un casque de réalité virtuelle.

Le projet a été divisé en plusieurs parties :

- Une partie, contrôle du robot où l'objectif est de le piloter à l'aide d'un microcontrôleur et que celui-ci soit configuré comme un serveur Wi-Fi pour recevoir des données à partir des manettes du casque de réalité virtuelle.
- Une partie visualisation de la caméra 360° dans le casque de réalité virtuelle en y créant un environnement virtuel.

Nous disposons des équipements, ci-dessous :

- STM32 F411RE
- Raspberry Pi 3 B+
- Router Cisco
- Carte Contrôleur T'Rex
- Robot Tank
- Caméra Ricoh Theta 5
- Casque VR HTC Vive + Manettes
- Ordinateur

Afin de mieux comprendre le projet, ci-dessous le schéma synoptique du projet:

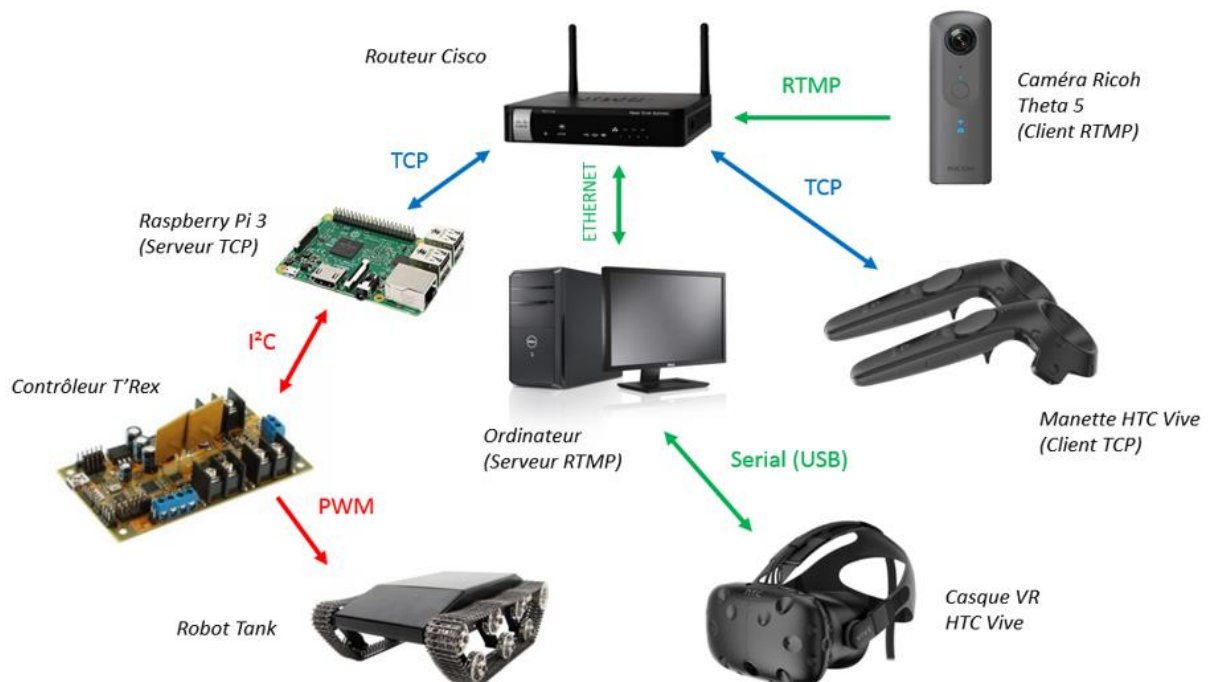


Figure 1 : Schéma synoptique du projet

Le projet a été découpé 2 parties :

- Une partie sur le robot (Julien/Antoine)
- Une partie sur le casque VR et la caméra (Maria/Olivier)

Nous avons séparé par la suite les tâches spécifiques telles que la partie CAO (Antoine), la partie réseau TCP (Julien(Serveur) & Olivier (Client)) et l'environnement virtuel (Maria).

À la fin du projet tout a été regroupé pour finaliser celui-ci.

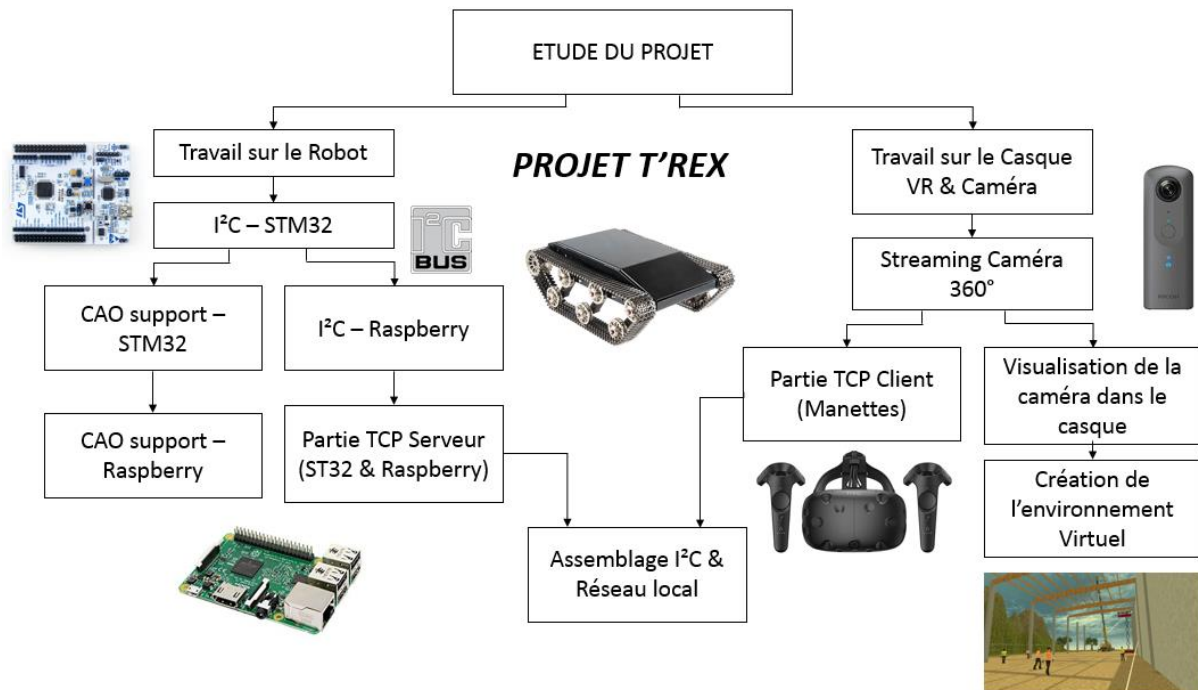


Figure 2 : Algorithme du déroulement du projet

2. Travail effectué sur le robot

2.1 Pilotage du robot

Dans un premier temps, nous avons lu la documentation technique de la carte contrôleur T'Rex RS036, ainsi pour piloter le robot en I²C, nous sommes partis sur la carte Raspberry Pi 3 puisque celle-ci possède les pins SDA et SCL permettant de faire de l'I²C et est capable de se configurer en serveur Wi-Fi. Une fois la carte choisie, la première chose à vérifier, sont les branchements des fils I²C. Une photo qui montre le code couleur pour chaque fil a été suivi tout le long du projet .

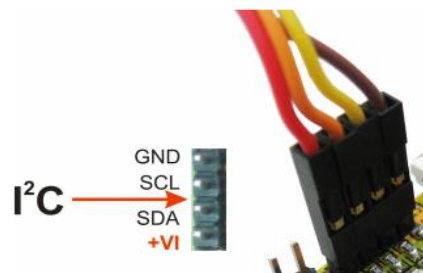


Figure 3 : Code couleur câblage I²C

Une fois cela fait, nous avons pu commencer à mettre en place la liaison I²C. Il faut prendre en compte l'adresse du contrôleur qui est 0x07 et la fréquence de l'I²C qui est de 100kHz.

Pour commander la carte, il faut envoyer un ensemble de données de 27 octets.

Ces 27 octets sont décomposés de la manière suivante :

- Octet 1 : Octet de start, de valeur 0x0F
- Octet 2 : Fréquence PWM : 0x02 par défaut
- Octets 3-4 : Vitesse du moteur gauche
- Octet 5 : Arrêt du moteur gauche
- Octets 6-7 : Vitesse du moteur droite
- Octet 8 : Arrêt du moteur droit
- Octets 9 à 20 : Positions des servomoteurs
- Octet 21 : Valeur de vibration du robot : 0x32 par défaut
- Octets 22 - 23 : Ajustement de la vibration du robot : 0x32 par défaut
- Octets 24 - 25 : Valeur minimale du fonctionnement de la batterie
- Octet 26 : Adresse I²C de la carte contrôleur (A ne pas modifier)
- Octet 27 : Fréquence de l'I²C (0x00 par défaut)

Différents tableaux ont été créés en fonction de l'instruction que l'on demande au robot, en modifiant principalement les octets 3-4 et 6-7 (en rouge dans l'exemple suivant). Sachant que le premier octet n'est pas inclus dans le tableau car il est envoyé à l'aide de la fonction d'écriture, l'exemple suivant est un tableau de données, permettant d'avancer le robot :

avancer = [0x02, 0xFF, 0x01, 0x00, 0xFF, 0x01, 0x00, 0x05, 0xDC, 0x05, 0xDC, 0x05, 0xDC, 0x05, 0xDC, 0x00, 0x00, 0x00, 0x00, 0x00, 0x32, 0x00, 0x32, 0x02, 0x23, 0x07, 0x00]

Pour envoyer ces valeurs, nous avons utilisé les fonctions de la bibliothèque « SMbus », une bibliothèque permettant de faire de l'I²C en Python.

Pour écrire, nous avons utilisé la fonction : **write_i2c_block_data(addr, 0x0F, avancer)**.

Avec 'addr' qui correspond à l'adresse de la carte contrôleur (0x07), 0x0F l'octet de Start (1er Octet) et "avancer" le tableau de données vu ci-dessus.

Pour vérifier que l'esclave (Contrôleur T'Rex) répond au maître (Raspberry), nous utilisons la fonction **read_byte(addr)**. Généralement le "read" doit envoyer des informations liées au robot, mais dans notre cas, nous ne récupérons pas ces données.

Le code complet est disponible en annexe.

2.2 CAO carte extension + Support

Une fois la liaison I²C validée, une étape essentielle pour le projet, était de pouvoir rendre le montage Raspberry Pi / carte contrôleur T'Rex RS036 embarquable sur le robot.

Pour cela il nous fallait une carte ou shield pour la Raspberry Pi. Ce dernier doit permettre une liaison I²C entre les deux cartes. Nous avons décidé de réaliser ce shield Raspberry Pi sous Altium Designer.

Une liaison I²C a besoin de résistance de PULL UP pour fonctionner. Ces résistances permettent, lorsque aucun périphérique n'impose quelque chose sur le bus, que ce dernier soit à 1 (5V). pour nos prototypes, nous avons choisi des résistances de 4.7K qui est une valeur de résistance dans la moyenne pour ce genre de liaison. Une donnée importante à prendre en compte est que la carte Raspberry Pi possède déjà des résistances de Pull UP en interne. en rajouter ne sert donc à rien et peut même provoquer des dysfonctionnements car cela pourrait avoir un effet pont diviseur qui serait dommageable pour le système.

Le shield Raspberry Pi réalisé est donc le suivant :

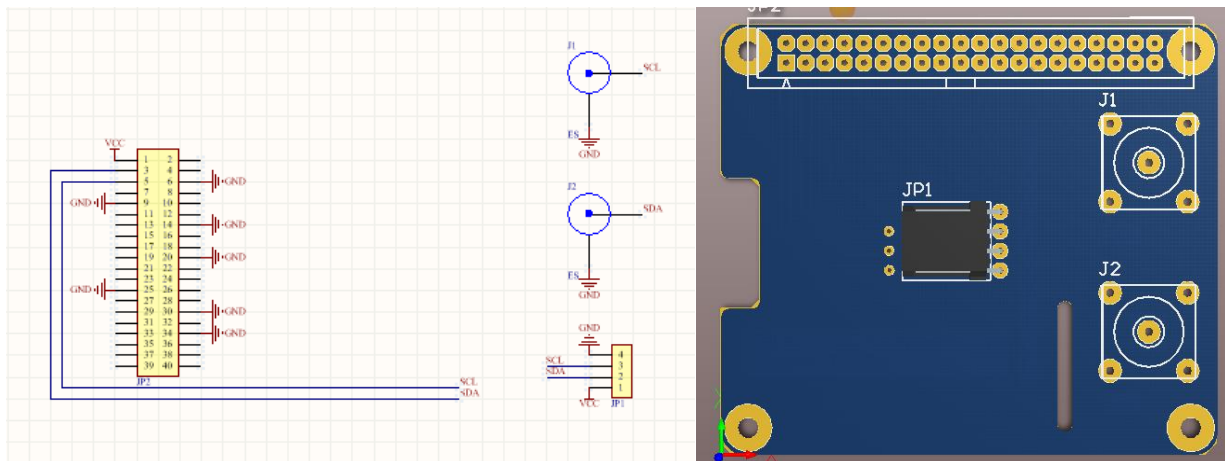


Figure 4 : Schéma de câblage et PCB de la carte d'extension Raspberry

J'ai choisi d'ajouter deux connecteurs BNC afin de pouvoir, à l'avenir, observer les trames de données envoyer à la carte contrôleur T'Rex RS036 sans avoir besoin de débrancher ou de placer des sondes.

Une fois le shield soudé et tester à l'écart, il fut connecté à la carte Raspberry Pi

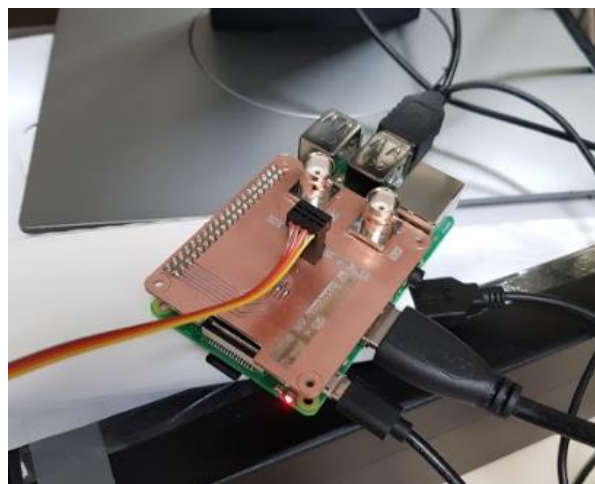


Figure 5 : Photo de la carte d'extension Raspberry

Une seconde partie importante était de créer une pièce pouvant accueillir la carte Raspberry Pi (l'isolant ainsi de l'ossature métallique du drone) mais également pouvant permettre l'installation de la caméra à vision 360°. La pièce 3D a été conçue sous SolidWorks.

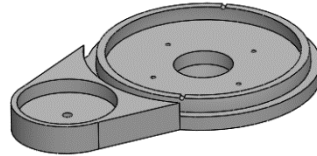


Figure 6 : Vue 3D du support

La pièce est dans la même lignée que la précédente qui était un dôme, mais a été redimensionnée et redessinée pour mettre en valeur la carte électronique et permettre d'accueillir la caméra tout en restant dans le gabarit du robot. Afin de désigner au mieux cette nouvelle pièce, j'ai décidé de modéliser l'ensemble du système (extérieur et intérieur) dans lequel elle allait s'intégrer, ainsi je pourrais valider totalement cette dernière avant de l'envoyer en production. cela nous donne ceci:

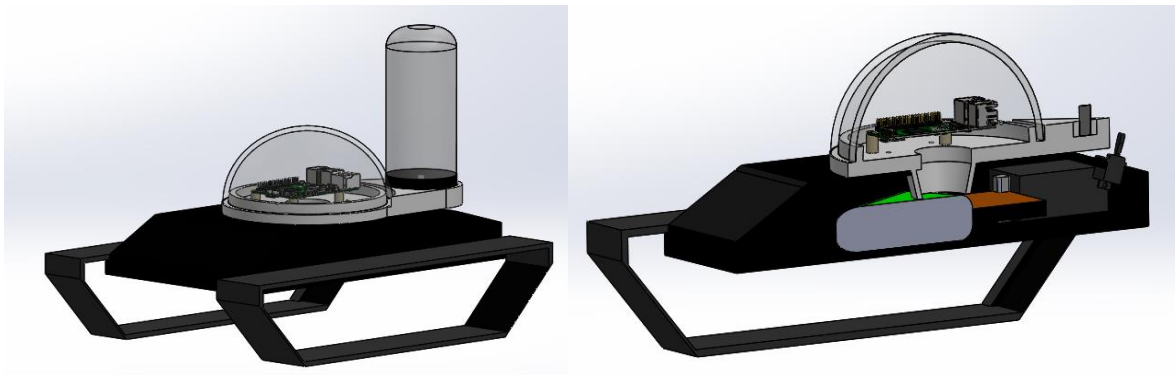


Figure 7 : Vues 3D du robot T'Rex avec le support

Une fois toutes les contraintes validées, la pièce fut produite et montée sur le système validant définitivement la conception



Figure 8 : Photo du robot avec le support de la carte Raspberry

3. Mise en place de la VR et du réseau local

3.1 Manette

3.1.1 Gestions des trames de la manette

Comme nous l'avons dit précédemment, l'objectif final est de contrôler le robot avec les manettes du casque virtuel. Pour ce faire, je devais concevoir un programme qui me permettrait de récupérer les valeurs envoyées par les manettes ou de détecter qu'un bouton de la manette a été pressé.

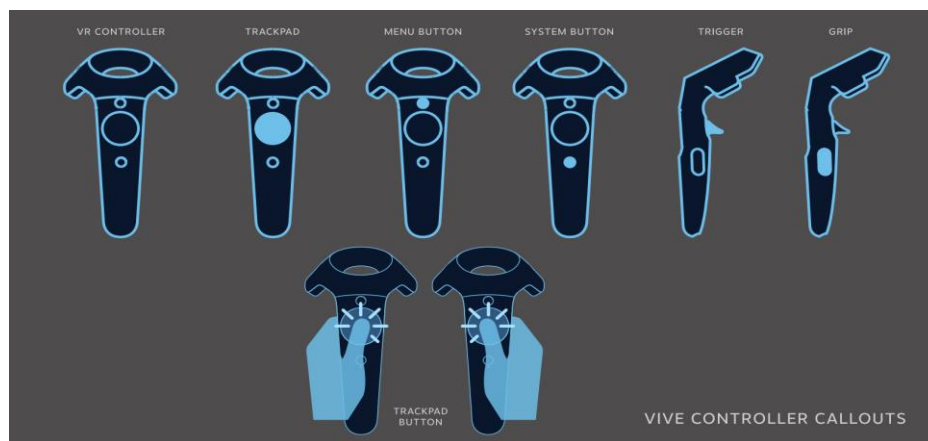


Figure 9 : Présentation des touches des manettes HTC Vive

À l'aide de la bibliothèque open source "OpenVR" et au projet GitHub "OpenVR-Quick-Start", j'ai pu concevoir un programme sous Visual studio en C++ qui me permet de détecter et d'afficher le bouton appuyé.

(<https://github.com/osudr/CassieVrControls/wiki/OpenVR-Quick-Start>)

Une fois le boîtier du système virtuel initialisé, le programme est alors dans une boucle infinie où se trouve un switch case. Il va nous permettre de détecter rapidement si une touche des manettes a été pressé ou relâché. Lorsque c'est le cas, il va d'abord déterminer si la touche enclenchée se trouve sur la manette droite ou gauche. Puis il va afficher dans le terminal, la manette en question ainsi que le nom de la touche enclenchée. et vice-versa lorsqu'on relâche la touche (voir annexe)

3.1.2 Envoi de données en Wi-Fi

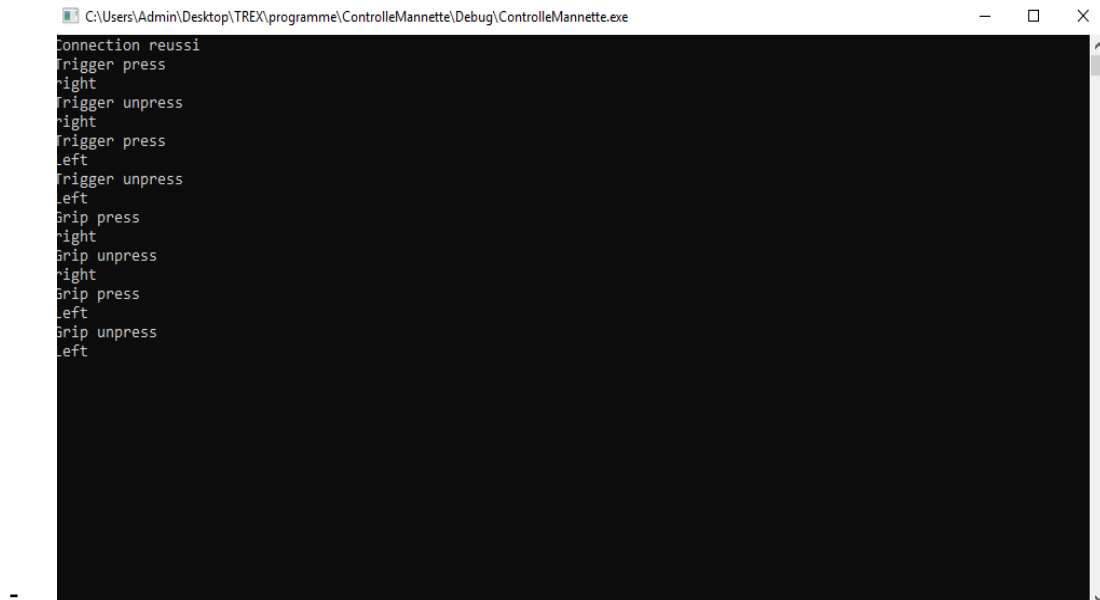
Nous nous sommes concentrés ensuite sur l'envoi de données en Wi-Fi. Le but est d'envoyer des ordres à la Raspberry pi uniquement lorsqu'on maintient une touche appuyée. Pour ce faire, nous avons implémenté un client TCP au programme des manettes. Sur la Raspberry, nous avons programmé un serveur TCP en Python qui recevra les commandes du client.

Nous avons choisi cette solution car c'est un procédé que nous maîtrisons.

Nous avons ensuite associé des commandes au bouton. les commandes vont de 0 à 4:

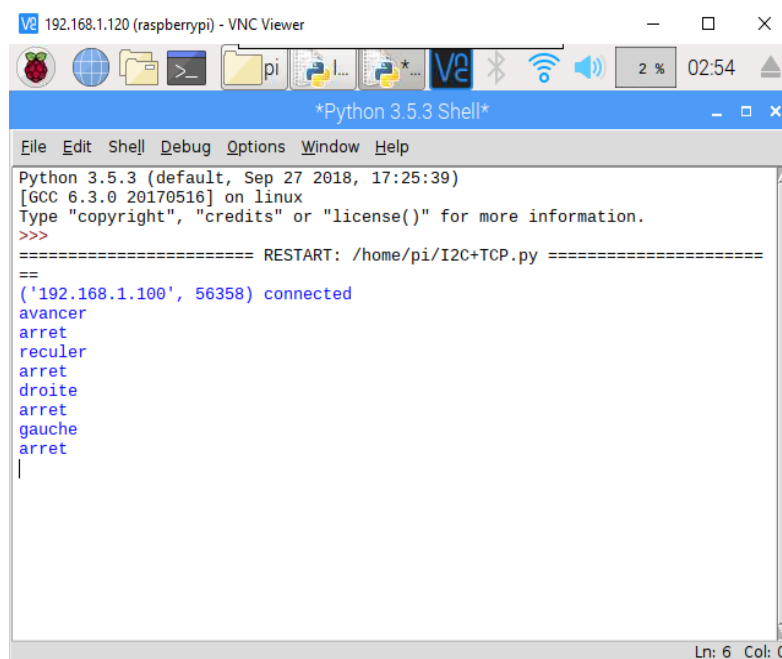
- 0 : arrêter (pas d'appui sur bouton)
- 1 : marche avant (bouton trigger droit)
- 2 : marche arrière (bouton trigger gauche)
- 3 : tourner à gauche (bouton grip gauche)
- 4 : tourner à droite (bouton grip droit)

(voir annexe)



```

C:\Users\Admin\Desktop\T'REX\programme\ControleMannette\Debug\ControleMannette.exe
connection reussi
trigger press
right
trigger unpress
right
trigger press
left
trigger unpress
left
grip press
right
grip unpress
right
grip press
left
grip unpress
left
  
```



```

192.168.1.120 (raspberrypi) - VNC Viewer
*Python 3.5.3 Shell*
File Edit Shell Debug Options Window Help
Python 3.5.3 (default, Sep 27 2018, 17:25:39)
[GCC 6.3.0 20170516] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/pi/I2C+TCP.py =====
==
('192.168.1.100', 56358) connected
avancer
arret
reculer
droite
arret
gauche
arret
  
```

Figure 10 : Photos du terminal du programme client (Manettes) et du programme serveur (Raspberry Pi)

3.2 Casque

3.2.1 Le streaming vidéo de la caméra

Pour récupérer le flux vidéo de la caméra, nous devons d'abord connecter la caméra au réseau local que nous avons mis en place avec un routeur Cisco. (voir annexe).

Comme serveur de streaming, nous avons utilisé dans un premier temps YouTube. Il suffit de créer un live sur celui-ci avec un compte Gmail (attention : si on n'a jamais fait de live YouTube avant, il y a une attente de 24h avant de pouvoir en créer un). Une fois ce live créé, YouTube nous renseigne une URL qui correspond à l'adresse de notre live. Nous récupérons cette URL ainsi que le "stream name/key" afin de les renseigner dans l'interface web de la caméra (voir annexe). Cela nous permet d'indiquer à la caméra où envoyer le flux vidéo.

Après plusieurs tests et plusieurs paramètres, nous avons réussi à récupérer le flux vidéo à l'aide de YouTube. Mais il y avait un délai d'environ 30 secondes.



Figure 11 : Photo du live YouTube avec la caméra 360°

Néanmoins, notre objectif était d'avoir un serveur de streaming en local et de lire le flux vidéo à l'aide de VLC. Nous utilisons ce 'media player' car c'est un des plus connus et simple à utiliser qui nous permet de lire une vidéo en streaming avec le protocole RTMP (Real Time Messaging Protocol), protocole utilisé par la caméra. De plus, il existe un plugin VLC pour Unreal Engine 4 (4.20.3) qui nous permettra donc d'afficher le flux vidéo dans l'environnement virtuel. L'utilisation de YouTube nous a permis de comprendre, dans les grandes lignes, comment procéder pour récupérer le flux vidéo.

Après plusieurs recherches, nous avons trouvé un projet open source qui nous permet d'installer un serveur de streaming en local qui dispose du protocole RTMP. Il s'agit de NGINX (voir annexe). Une fois le serveur en marche, on renseigne cette fois-ci dans l'interface web de la caméra l'URL correspondant à notre serveur (exemple : `rtmp//ip_du_Pc`). De plus on peut également régler la résolution et la vitesse de transmission des données. Nous avons réglé la résolution à 2K (1920x960) et la vitesse à 3 Mps. Ce réglage nous permet d'avoir un retour vidéo fluide. Cependant, il existe toujours un décalage de 4 secondes entre l'envoi et la réception de la vidéo, ce qui n'est pas optimal en vue du pilotage du robot en temps réel avec le casque virtuel. De plus, la vidéo n'est pas de bonne qualité.

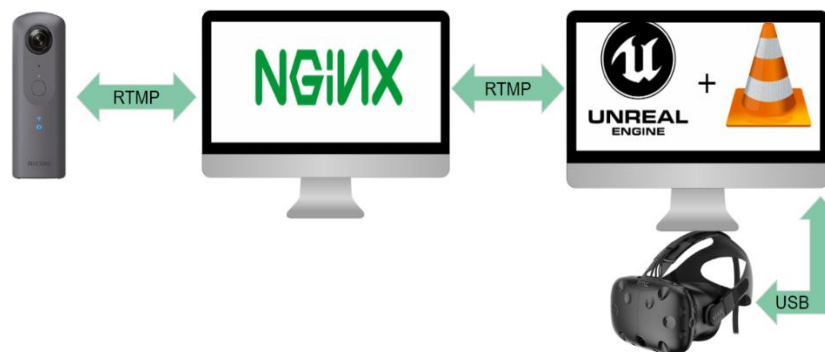


Figure 12 : Protocole de streaming

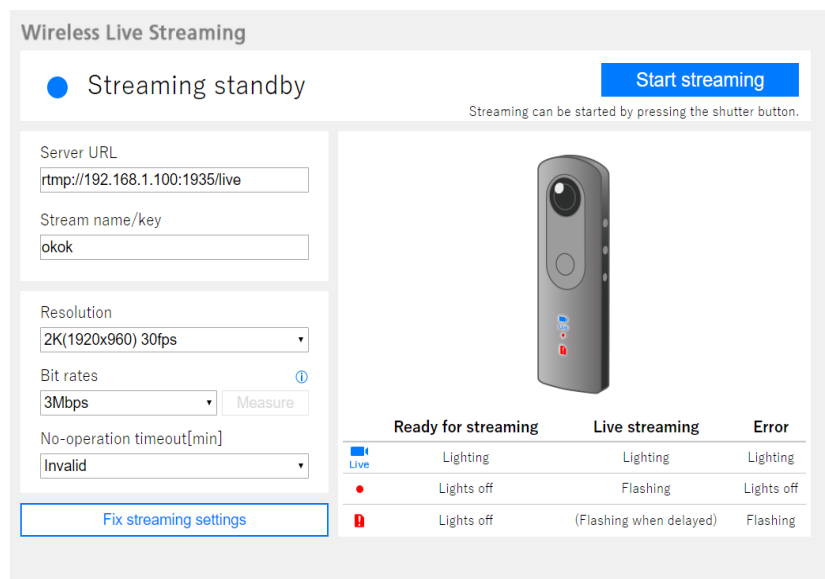


Figure 13 : Interface graphique du plugin de la caméra.

3.2.2 Affichage de la vidéo dans l'environnement VR

Une fois la vidéo récupérée, il reste à la mettre dans le casque de réalité virtuelle. Pour cela, on utilise le logiciel Unreal Engine 4 et un casque HTC Vive. Pour prendre en main la partie vidéo de ce logiciel, on essaie tout d'abord de mettre une vidéo 180° enregistrée sur l'ordinateur. À partir d'un projet de base, on peut créer un plan et associer la vidéo à l'aide des Blueprints (voir annexe). Lors de la création de ce projet, le choix nous est laissé de lire une vidéo à partir d'une URL ou en mettant directement la vidéo dans le projet. Dans notre cas c'est à partir d'une URL puisque la vidéo est en direct. L'URL à utiliser est celle donnée par le server. Attention à ne pas oublier de rajouter la "key" du stream.

La deuxième étape est de lire une vidéo 360°. Pour cela on ne peut pas prendre un plan car cela aplatit la vidéo et elle apparaît devant nos yeux et non pas autour de nous. la meilleure des solutions est de créer une "matière" à partir de la vidéo et de recouvrir les parois internes d'une sphère de cette matière. Alors on pourra se placer au centre de la sphère et se retrouver au centre de la vidéo. On pourra alors se croire dans le monde de la vidéo. Les parois externes de la sphère ne doivent pas laisser passer la lumière externe. Elles doivent être recouvertes d'une matière opaque. À l'intérieur de la sphère, il ne faut pas oublier de placer un "point light" pour que la vidéo soit visible.

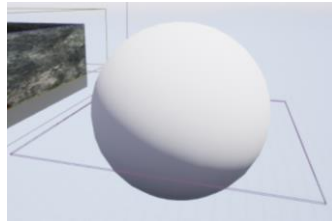


Figure 14 : Sphère pour la vidéo 360°

Enfin, la dernière étape est la mise en place de la vidéo en direct de la caméra du robot. Unreal Engine 4 possède plusieurs logiciels de lecture vidéo comme média player. Cependant aucun ne peut lire les vidéos de type rtmp. Or la vidéo à lire est de type rtmp.

Server URL	<input type="text" value="rtmp://192.168.1.100:1935/live"/>
Stream name/key	<input type="text" value="okok"/>

Figure 15 : URL et clé de la vidéo

VLC média player peut lire ce type de vidéo. N'étant pas directement intégré à Unreal Engine 4, il faut télécharger le plugin. Cependant, toutes les versions d'Unreal Engine 4 ne sont pas compatibles. La version finale choisie est donc 4.20.3. Afin d'avoir un point de départ et de retour, nous avons choisi de créer une map de départ. Cette map doit avoir un point de téléportation vers la sphère et un moyen de retour de la sphère vers cette map.



Figure 16 : Map de départ

Pour créer cette map, il faut tout d'abord réaliser l'environnement. Il faut ensuite gérer les déplacements sur la map en elle-même et les téléportations. Nous avons choisi d'établir un point de téléportation. Il suffit de toucher ce point avec les mains représentant les manettes du HTC Vive. Alors on se retrouve à l'intérieur de la sphère. Il est nécessaire de placer un plan dans la sphère pour que lors de la téléportation le personnage soit bloqué dans la sphère et reste sur le plan.



Figure 17 : Zone de départ

Les déplacements se font par téléportation. Il ne faut pas oublier de délimiter la zone de déplacement en créant un “NavMeshBoundVolume”. Enfin il reste à associer un bouton de la manette à un point de la map afin que l’on puisse revenir à cet endroit quand on veut. C’est une sorte de bouton retour. Pour cela on utilise aussi une téléportation. L’association se fait dans le blueprint.

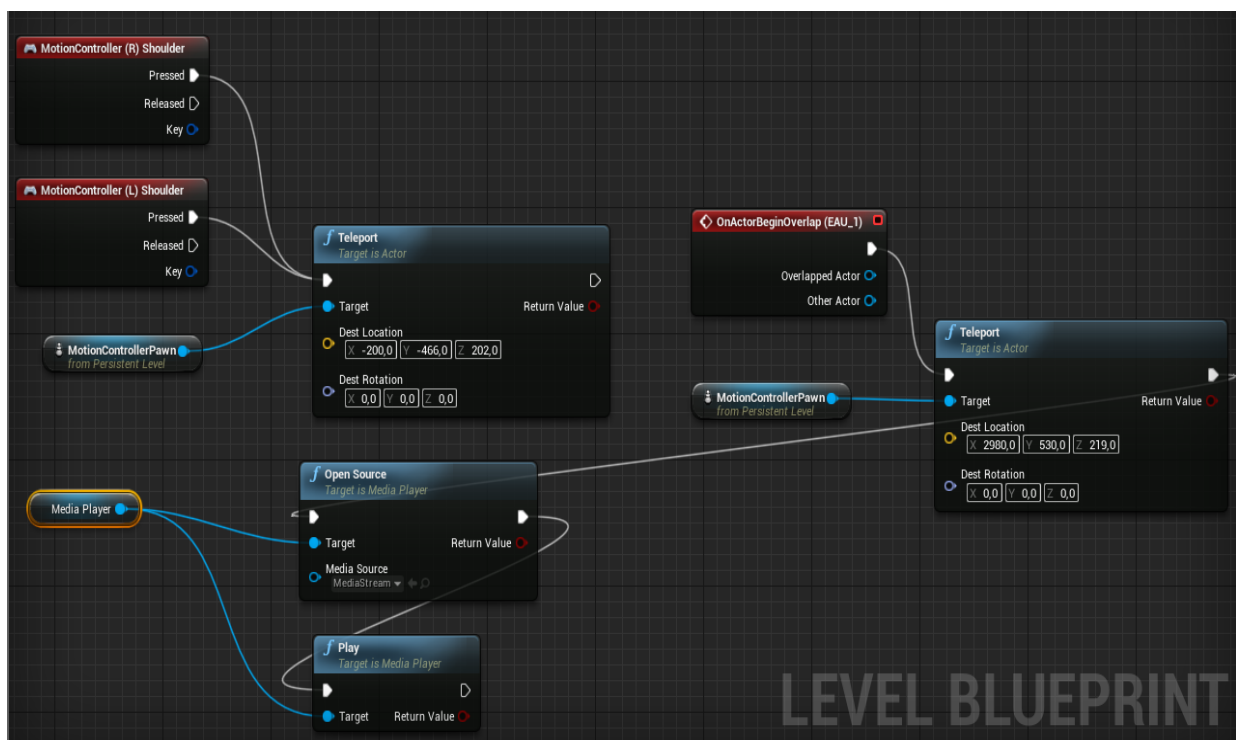


Figure 18 : Blueprint

4. En cours de développement

4.1 I²C et Wi-Fi avec la STM32

Lors du projet, nous avons commencé par travailler avec la carte STM32 NUCLEO-F411RE. La liaison I²C a été réalisé comme sur la Raspberry. Cependant, la partie sur le serveur Wi-Fi est à développer par la suite. En effet, la carte d'extension IDW01M1 possède un module Wi-Fi "SPWF01SA", que l'on branche sur la carte STM32. Nous pouvons initialiser, envoyer et recevoir des données au module Wi-Fi grâce aux différentes commandes 'AT' et communiquer avec un ordinateur (client) dans le réseau local.

Cependant le traitement des données reçues est encore en cours de développement. Le buffer ou le registre permettant de récupérer cette valeur n'a pas été encore identifié. Il est donc impossible pour le moment d'envoyer une trame I²C défini en fonction de la valeur reçu.

4.2 Pilotage du robot par Bluetooth

Une option permettant de contrôler le robot par Bluetooth est disponible sur la carte contrôleur. Nous avons téléchargé l'APK de l'application Android pour l'installer sur un téléphone. Le module Bluetooth 'HC-05' a était branché sur les pins de la carte.

Nous avons essayé de connecter le téléphone avec le module mais en vain.

En effet dans la datasheet, il est précisé que le module Bluetooth est censé s'initialiser au démarrage en modifiant le nom et le code d'accès du module. Cependant, vu que nous utilisons un module de la section IT2R, celui-ci était déjà programmé à l'avance, donc impossible à modifier. Nous avons laissé de côté le Bluetooth pour passer à l'I²C.

4.3 Carte d'extension STM32

Pour ce qui est de la conception de la carte. Contrairement à la Raspberry Pi, la carte STM32 NUCLEO-F411RE ne possède pas de résistance de Pull Up. D'autre part la carte à réaliser ne se trouvait pas sur la cible mais sous cette dernière. Cela nous imposait donc une architecture différente:

Dans un premier temps j'ai décidé d'avoir deux connecteurs pour la nappe I²C. un premier sous la carte qui permettra de connecter la carte STM32 NUCLEO-F411RE à la carte contrôleur T'Rex RS036. Le second, placé sur le haut, servira à nos tests car il sera plus facilement accessible.

Une autre modification de l'architecture importante est que la taille plus imposante de cette carte rendait compliqué l'ajout du connecteur micro USB nécessaire à l'alimentation de la carte STM32 NUCLEO-F411RE. L'idée fut donc d'alimenter cette dernière avec la batterie 6V. De là, nous avons plusieurs possibilités, soit abaisser le courant à 5V ou 3.3V pour alimenter la carte (à l'aide de Traco), soit augmenter le courant d'entrée à 12V.

En raison de la difficulté à trouver un Traco convenant aux dimensions voulues, nous avons opté pour la seconde solution. La carte a été produite et tester (sans la partie alimentation), validé mais en raison du passage sous Raspberry Pi ne fut pas conservée.

4.4 Amélioration de la qualité du streaming

Comme nous l'avons dit précédemment, il existe un décalage de 4 secondes lorsqu'on réceptionne le flux vidéo de la caméra. De plus, dans le casque virtuel, la vidéo n'est pas de bonne qualité. Nous pensons que le problème vient du plugin que nous utilisons. Ce plugin encode le flux vidéo avant de l'envoyer ce qui provoque une perte de temps. Ce que faudrait faire, c'est développer notre propre plugin à l'aide du SDK Android disponible pour ne pas utiliser l'encoder et donc réduire le temps de latence. Ensuite nous avons des doutes sur l'efficacité de notre réseau. Le problème peut venir également de notre router Cisco ou du serveur de streaming que nous utilisons (NGINX).

Conclusion

Ce projet, nous a permis de réaliser un travail de groupe dans des conditions de projet en entreprise. Nous avons pu mettre en œuvre les différentes notions vues durant la licence, mais en apprendre de nouvelles, comme par exemple l'utilisation d'Unreal Engine pour le casque VR.

Il reste toutefois, certaines parties qui ont été mises de côté, comme la partie avec la carte STM32 (Wi-Fi), le contrôle du robot par Bluetooth et la fluidité du streaming de la caméra, qui pourront être développées dans les années suivantes.

Tables des Figures

Figure 1 : Schéma synoptique du projet.....	3
Figure 2 : Algorithme du déroulement du projet	4
Figure 3 : Code couleur câblage I ² C.....	4
Figure 4 : Schéma de câblage et PCB de la carte d'extension Raspberry	6
Figure 5 : Photo de la carte d'extension Raspberry	6
Figure 6 : Vue 3D du support	7
Figure 7 : Vues 3D du robot T'Rex avec le support	7
Figure 8 : Photo du robot avec le support de la carte Raspberry	7
Figure 9 : Présentation des touches des manettes HTC Vive.....	8
Figure 10 : Photos du terminal du programme client (Manettes) et du programme serveur (Raspberry Pi).....	9
Figure 11 : Photo du live YouTube avec la caméra 360°	10
Figure 12 : Protocole de streaming	11
Figure 13 : Interface graphique du plugin de la caméra.....	11
Figure 14 : Sphère pour la vidéo 360°	12
Figure 15 : URL et clé de la vidéo	12
Figure 16 : Map de départ.....	12
Figure 17 : Zone de départ.....	13
Figure 18 : Blueprint.....	13

Annexe

https://github.com/JulienPally/TREX_Projet

Code d'accès au routeur Cisco (Page Web)

https://github.com/JulienPally/TREX_Projet/blob/master/En%20cours/Code%20Wifi%20IDW01M1/Code%20Acces%20Cisco.txt

Code d'accès au point d'accès Wi-Fi du Cisco

https://github.com/JulienPally/TREX_Projet/blob/master/En%20cours/Code%20Wifi%20IDW01M1/Mot%20de%20passe%20-%20Connexion%20au%20routeur.txt

Code client TCP - Gestion des manettes VR - Visual Studio

https://github.com/JulienPally/TREX_Projet/blob/master/Code%20Gestion%20des%20Manettes%20%2B%20Client%20TCP/ContrôleMannette/ContrôleMannette.cpp

Code Serveur TCP + Commande I²C (Raspberry) - Python

https://github.com/JulienPally/TREX_Projet/blob/master/Code%20T'Rex%20I%C%20%2B%200Serveur%20-%20Raspberry/Programme%20-%20I%C%20%2B%20Connexion%20au%20serveur%20-%20Raspberry.py

Comment ajouter la bibliothèque OpenVR à un projet Visual studio :

- Il faut télécharger le répertoire de la librairie qui projet disponible sur GitHub (<https://github.com/ValveSoftware/openvr>).
- Puis il faut ajouter les dossiers [controller](#), [callouts](#), [headers](#), [lib](#) et [src](#) au dossier ou se situe le fichier main.cpp du projet.
- Et enfin il faut mettre le fichier “ [openvr_api.dll](#) ” à côté du main.cpp également.

(pour voir notre exemple :

https://github.com/JulienPally/TREX_Projet/tree/master/Code%20Gestion%20des%20Manettes%20%2B%20Client%20TCP/ContrôleMannette)

Theta Ricoh :

- Connection à la caméra à l'aide de l'appli mobile :
 - Allumer la caméra (sans la connecter à un ordinateur)
 - Appuyer sur le bouton Wi-Fi jusqu'à ce que le voyant Wi-Fi devienne bleu
 - Aller dans la liste des Wi-Fi disponibles dans les paramètres de votre smartphone. Normalement, le SSID (“Theta....”) est affiché.
 - On se connecte à la caméra en indiquant les chiffres en dessous de la caméra (juste après “YL”, rentrer tous les chiffres à la suite)

Source : https://support.theta360.com/fr/manual/v/content/prepare/prepare_06.html

- Connexion de la caméra au routeur
 - Se connecter au Wi-Fi de la caméra via son smartphone
 - Ouvrir l'application mobile "THETA V"
 - Aller dans Paramètre/mode client Wireless LAN/Point d'accès/ajouter un nouveau point d'accès.
 - Connecter la caméra au routeur en appuyant sur le bouton Wi-Fi jusqu'à ce que le voyant Wi-Fi devienne vert

Source : <https://community.theta360.guide/t/theta-v-client-mode-configuration-guide/2565>

Utilisation du plugin "Wireless live streaming" :

- Pour installer le plugin, il faut installer l'application Windows sur le pc puis aller sur le site officiel de Ricoh Theta pour télécharger le plugin et suivre les instructions indiquées lors de la procédure

Source : <http://theta360.guide/wireless-plugin-guide/>

Installation et utilisation de NGINX (serveur de streaming en local) :

- Télécharger le dossier "Nginx Gryphon" (<http://nginx-win.ecsds.eu/download/>)
 - Aller dans le répertoire "Conf" puis modifier le fichier "nginx.conf" en ajoutant les lignes de code correspondant au serveur de streaming :
- ```

- rtmp {
- server {
- listen 1935;
- chunk_size 4096;
-
- application live {
- live on;
- record off;
- }
- }
- }

```
- (source : <https://obsproject.com/forum/resources/how-to-set-up-your-own-private-rtmp-server-using-nginx.50/>)
  - Pour lancer le serveur, exécuter l'application "nginx.exe" en tant qu'administrateur. Pour vérifier si le serveur est en marche, ouvrir le gestionnaire de tâches.

Lire une vidéo en streaming sur Unreal Engine 4 (4.20.3) :

<https://docs.unrealengine.com/en-US/Engine/MediaFramework/HowTo/StreamMediaSource/index.html>

Créer un projet pour une lecture de vidéo 180° téléchargée sur ordinateur

<https://docs.unrealengine.com/en-US/Engine/MediaFramework/HowTo/FileMediaSource/index.html>

Lien vers projet VR : [https://drive.google.com/drive/folders/1wKuK6QkVGvGE8Xo\\_AT-5coKN8l1Bthmj?usp=sharing](https://drive.google.com/drive/folders/1wKuK6QkVGvGE8Xo_AT-5coKN8l1Bthmj?usp=sharing)