

PROJET DRONE TERRESTRE EN VR



MAXENCE ADNOT — THOMAS FIRION — AURÉLIEN BONNET

RESPONSABLE PROJET : M.RUIZ

RESPONSABLES LICENCE : MME.BRISSARD & M.CLAMENS

Sommaire

Table des matières

Introduction	3
1. General.....	4
A. Interconnexion.....	4
2. Liaison Raspberry Bluetooth	5
A. Changement de transmission.....	5
B. Bluetooth	6
C. Analyse trame	7
D. Code python.....	9
3. Echange de données vidéo	11
A. Streaming vidéo avec une caméra 360	11
B. Création et utilisation de l’environnement VR sous Unreal Engine	14
4. Partie commandes du drone	17
A. Idée d'amélioration	17
B. Conception du programme informatique.....	18
Conclusion	23

Introduction

Le projet à réaliser et à améliorer est un drone terrestre connecter et en réalité virtuel.

Le but du projet est de pouvoir contrôler le drone terrestre grâce aux manettes d'un casque de réalité virtuel, et de pouvoir récupérer le flux vidéo de la caméra 360° qui est sur le drone, tous cela connecter sur un réseau internet.

Lien du projet GitHub

<https://github.com/AurelienBonnet/projetdroneiutcachan>

1. General

A. Interconnexion

Ci-dessous vous pouvez voir un diagramme des éléments du projet ainsi que les protocoles de connexion qu'ils peuvent y avoir entre eux

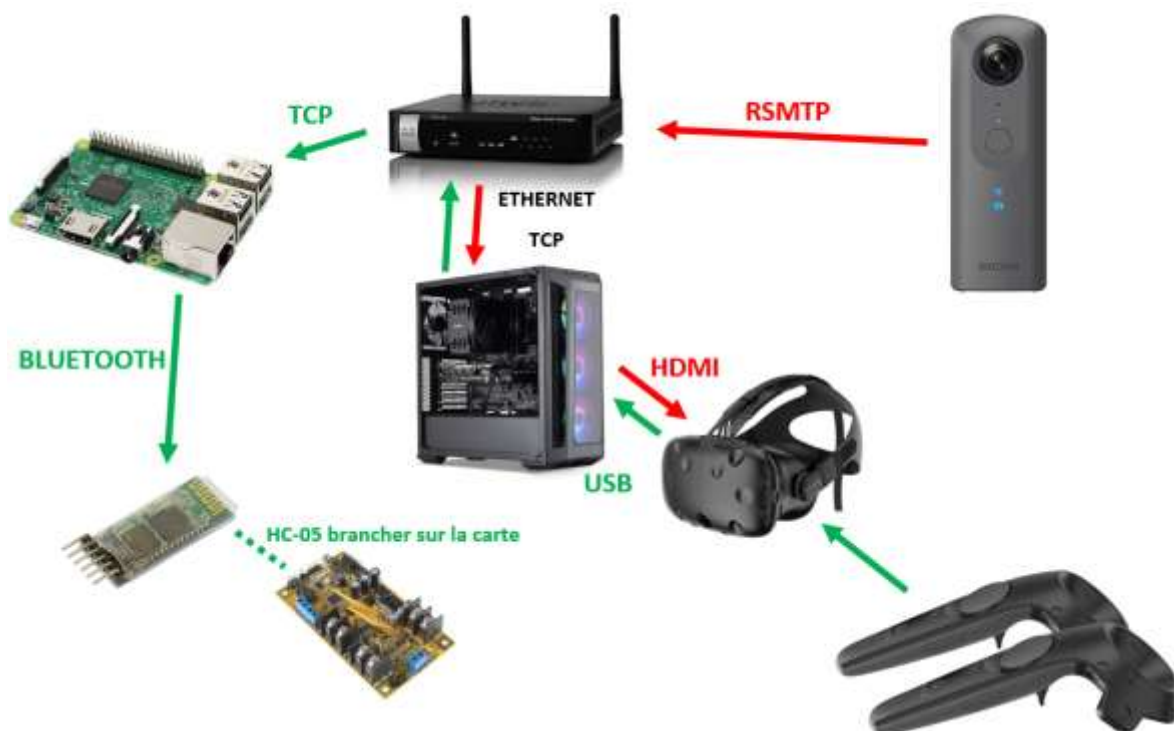


Figure 1 : Diagramme du projet

Pour pouvoir réaliser le projet nous avons configuré un sous-réseau dans le réseau internet de l'IUT grâce à un router Cisco.

SSID : cisco

MDP : projetdrone2021

La raspberry, l'ordinateur et la caméra doivent donc être connectés sur le même sous-réseau pour communiquer.

2. Liaison Raspberry Bluetooth

A. Changement de transmission

Dans un premier temps, les projets précédents utilisaient la communication I2C entre la carte raspberry et la carte de contrôle du drone.

Pour pouvoir améliorer le projet j'ai décidé de changer de communication car les cartes de contrôle du robot sont sensibles, et certaines ont la partie électronique qui prend en charge l'I2C qui est endommagée. Pour pouvoir continuer à utiliser les cartes endommagées je vais donc utiliser le deuxième mode de communication qui est pris en charge par la carte de contrôle qui est le Bluetooth pour pouvoir communiquer avec le robot.

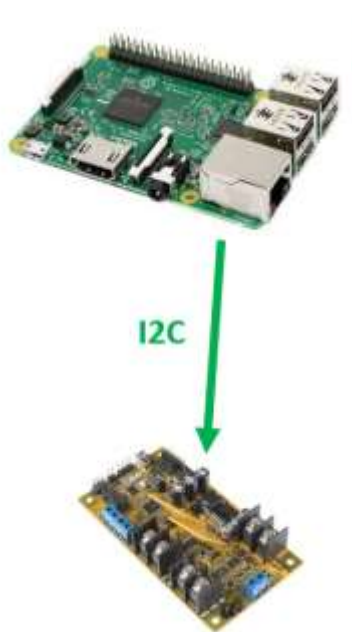


Figure 2: Ancienne communication en I2C



Figure 3: Nouvelle communication en Bluetooth

B. Bluetooth

Pour l'utilisation du Bluetooth nous devons brancher sur la carte du T'REX un module Bluetooth HC-05.

Ensuite, une application a été créée par le constructeur (l'APK se trouve dans le dossier du projet)



Figure 4: Application T'REX

Quand un appareil est connecté au HC-05 une led verte clignote à côté de la led bleu.



Figure 5: Bluetooth HC-05 sur carte T'REX

Maintenant que nous avons compris le principe nous allons récupérer les fichiers java de l'APK pour connaître les valeurs qui sont envoyées pour manipuler le drone.

Pour cela j'ai utilisé le site suivant «<http://www.javadecompilers.com/apk> » pour pouvoir récupérer les fichiers java et analyser le code.

Une fois la décompilation effectuée par le site suivre le chemin suivant pour retrouver le fichier principal java (TREXV2.3.apk → sources → com → robot → trex → MainActivity.java)

C. Analyse trame

En analysant le code j'ai compris que pour chaque moteur du drone une valeur en 0 et 255 est envoyer.

Pour visualiser cela, j'installe sur un ordinateur le logiciel « REALTERM » (le fichier .EXE est dans le dossier du projet)

Puis je branche le module Bluetooth HC-05 sur un LPC qui va permettre de le brancher sur un port USB de l'ordinateur.

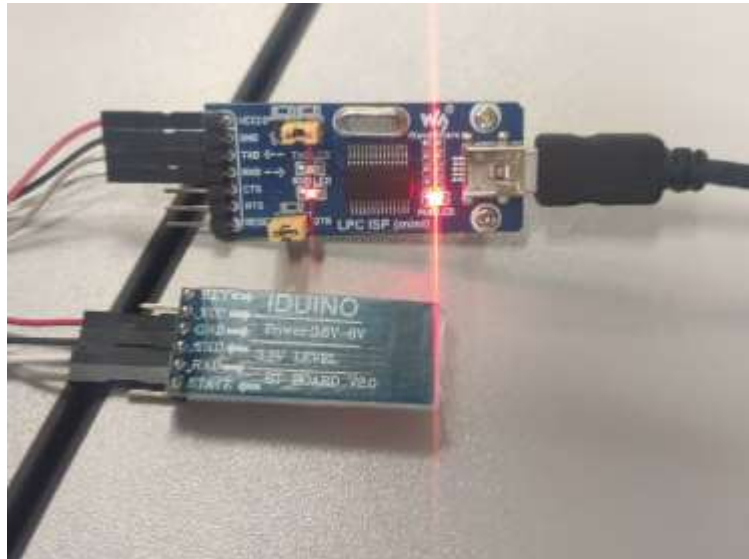


Figure 6: Bluetooth HC-05 & LPC ISP

Je configure le logiciel REALTERM tel que :

-onglet "display"

Display as --> hex

-onglet "port"

Baud --> 9600

Port --> (port com ou le Bluetooth est brancher)

Le bouton "open" doit  tre enfoncer

Puis cliquez sur "Charge"

Maintenant vous pouvez connecter votre smartphone au module HC-05 et ouvrir l'application est jouer avec les deux slides.

Vous recevrez dans le logiciel la trame que l'application envoie.

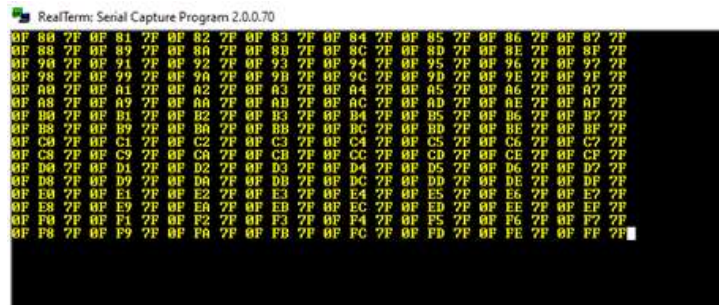


Figure 7: Trame application

Nous pouvons voir que la trame est : [0F, A, B]

A= valeur entre 0 et 255 chenille gauche

B= valeur entre 0 et 255 chenille droite

Voici l'interface de l'application, deux sliders, un pour la chenille gauche et l'autre pour la droite



Figure 8: Explication valeurs application

Nous pouvons en déduire que : entre les valeurs 126 et 0 le drone ira en arrière et pour les valeurs entre 128 et 255 le drone ira en avant

	AVANCER	STOP	RECULER
CHENILLE GAUCHE	128-255	127	126-0
CHENILLE DROITE	128-255	127	126-0

Figure 9: Tableau valeurs trame

D. Code python

Maintenant nous allons pouvoir recréer deux codes python

Le premier sera la réception des données envoyées par les manettes.

Le deuxième permettra d'attendre et de simuler les données des manettes.

Car pour rappel les manettes envoient des informations à la Raspberry puis elle envoie cette information sous la forme de la trame vue précédemment au Bluetooth sur la carte T'REX.

Le code de simulation des manettes se lance dans : anaconda → Jupiter → new → python 3 et collez le code (Le code est dans le dossier du projet simulation_manettes_py)

```
#----- A simple TCP client program in Python using send() function -----  
  
import socket  
clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM);  
clientSocket.connect(("10.10.10.104",54000));  
i=0  
while True:  
    data = '{:04d}'.format(i);  
    print(data)  
    clientSocket.send(data.encode());  
    i=i+1  
    if(i==10000):  
        i=0
```

Figure 10: Code simulation manettes

Le code qui permet la réception du code ci-dessus est lancé dans la Raspberry

(Le code est dans le dossier du projet
Reception_manettes.py)

Ce code récupère les informations envoyées sur le port 54000 du réseau auquel est connectée la Raspberry.

Comme les valeurs pour les deux moteurs sont entre 0 et 255 nous avons dû pour le moteur droit faire un +300 -300

Quand la valeur reçue est supérieure à 300 alors on sait que c'est la valeur du moteur droit

Puis on envoie la trame reconstituée sous la même forme que celle de l'application.

```
import time  
import socket  
import serial  
  
bluetoothSerial = serial.Serial("/dev/rfcomm0",baudrate=9600)  
  
socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
socket.bind(('',54000))  
socket.listen(5)  
client, address = socket.accept()  
print("{} connected".format( address ))  
  
y=500  
gauche=0  
droite=0  
  
while (1==1):  
  
    try:  
  
        msg = client.recv(4)  
        reponse=int(msg)  
  
        if(reponse >= 300):  
            droite=reponse-300  
        if(reponse <= -300):  
            gauche=reponse  
        print(msg)  
  
        trame = [15,int(gauche),int(droite)]  
        bluetoothSerial.write(trame)  
  
        #print(reponse)
```

Le fichier tuto.txt sous forme de tutoriel est dans le dossier pour permettre une mise en place du projet plus rapide

La carte de contrôle T'REX a plusieurs modes, il est possible de mettre la carte en mode Bluetooth en compilant un code dedans grâce à arduino (ce qui est dit dans la documentation de la carte).

Problème, impossible de trouver comment faire et de trouver ce code, après avoir demander a des fournisseur le contact des constructeurs qui reste sans réponse

Si un constructeur me répond alors j'enverrai un mail à M. RUIZ pour lui communiquer une potentiel solution

3. Echange de données vidéo

A. Streaming vidéo avec une caméra 360

Afin de récupérer le flux vidéo de la caméra afin de l'afficher dans le casque, il faut tout d'abord connecter la caméra vidéo au sous réseau géré par le routeur Cisco.

Dans un premier temps il faut allumer la caméra :

- Si la caméra est reset il faut donc rester appuyer sur la bouton avec le symbole Wi-Fi afin de permettre à la caméra de s'auto connecter au routeur et appuyer sur le bouton WPS du routeur sur la page du Cisco (Wireless->WPS) afin de laisser ouvert la connexion.



Figure 11: Caméra bouton wifi



Figure 12: Cisco bouton WPS

- Si la caméra et le routeur ne sont pas reset il suffit d'allumer la caméra et un voyant Wi-Fi vers d'affichera.



Figure 13: Caméra afficheur Wi-Fi actif

Pour vérifier la bonne connexion de l'appareil il suffit d'aller sur la page du Cisco et de regarder sur la caméra Theta est bien affiché (Networking->LAN->DHCP Leased Client).



Figure 14: Cisco DHCP IP

Afin de vérifier que le flux il faut utiliser un serveur de streaming, j'ai donc utilisé le plus simple c'est à dire YouTube. Avec un compte Google, il suffit de d'aller sur YouTube et de créer un live en cliquant sur le bouton "Passer en direct" (Attention : pour un compte n'ayant jamais utilisé la fonction live prendra 24h pour s'activer) Arrivé sur la page de configuration il faut récupérer l'URL du flux, la clé de Stream et enfin activer la fonction "Vidéo à 360 degrés".

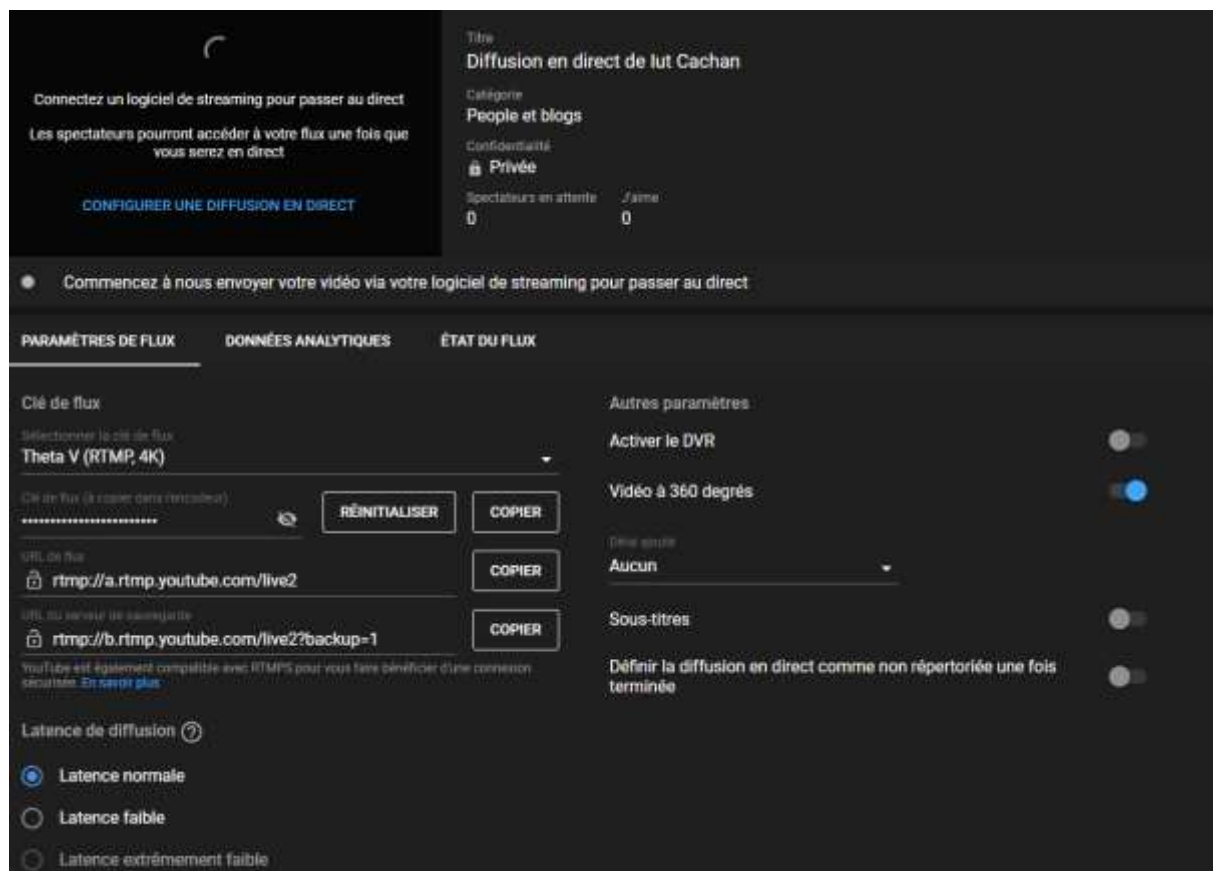


Figure 15: YouTube configuration de live stream

Il faut maintenant lancer le live, pour cela il suffit de se connecter sur la page web server de la caméra avec l'adresse IP fournie et le port du web server (8888) (l'IP est trouvable plus haut dans le Cisco), ici l'adresse est 10.10.10.105:8888. Attention la page web server de la caméra est accessible uniquement si l'appareil est en mode Live, pour l'activer il faut rester appuyer 5 sec sur le bouton Mode. Il faut donc mettre l'url du Stream et la clé dans les cases correspondantes et enfin lancer le live. (Je conseil d'appuyer sur le bouton mesure afin d'avoir le bit rates le plus adapté et le plus rapide)

Wireless Live Streaming

Streaming standby

Start streaming

Streaming can be started by pressing the shutter button.

Server URL

rtmp://a.rtmp.youtube.com/live2

Stream name/key

7v86-3wk4-6me7-6598-19b1

Resolution

4K(3840x1920) 30fps

Bit rates


AUTO (23.52 Mbps)

Measure

No-operation timeout[min]

Invalid

Fix streaming settings






	Ready for streaming	Live streaming	Error
 Live	Lighting	Lighting	Lighting
	Lights off	Flashing	Lights off
	Lights off	(Flashing when delayed)	Flashing

Figure 16: Configuration web server Theta V

Pour la suite il faut au préalable récupérer le lien vidéo du Stream avec le format m3u8, pour ce faire il suffit d'aller sur le live en tant que spectateur, clic droit sur la page puis "Afficher le code source de la page" ou Ctrl+U, et sur la nouvelle page effectuer CTRL+F avec le ".m3u8" et enfin copier l'adresse https qui précède la recherche. Exemple: https://manifest.googlevideo.com/api/manifest/hls_variant/expire/1624549880/ei/mFXUYOq6FqP7xN8PkPWm6Al/ip/129.175.237.130/id/5qap5aO4i9A.1/source/yt_live_broadcast/requiressl/yes/hfr/1/playlist_duration/30/manifest_duration/30/maxh/2160/maudio/1/vprv/1/go/1/nvgoi/1/keepalive/yes/fexp/24001373%2C24007246/dover/11/itag/0/playlist_type/DVR/sparams/expire%2Cei%2Cip%2Cid%2Csource%2Crequiressl%2Chfr%2Cplaylist_duration%2Cmanifest_duration%2Cmaxh%2Cmaudio%2Cvprv%2Cgo%2Citag%2Cplaylist_type/sig/AOq0QJ8wRQlhAOoPqTGgwirtRNMWjh63X9cURFW1TPCIkFD4gGB6RH5AiAGsEiDlorO_eddHYOOpQU82hMy-W3GsJUEfH8Z8s8LqA%3D%3D/file/index.m3u8

B. Création et utilisation de l'environnement VR sous Unreal Engine

Afin de créer l'environnement VR permettant de visualiser le Stream YouTube à l'intérieur du casque HTC Vive, j'ai utilisé le moteur graphique de jeu vidéo développé par Epic Games, Unreal engine et le plugin Stream VR pour permettre la connexion entre Unreal Engine et l'HTC Vive.

Il faut avant tout posséder VLC et le plugin VLC pour Unreal Engine qui permettra la diffusion du Live sur le support 3D sélectionné. Si le plugin VLC n'est pas déjà installé il faut télécharger le zip et l'extraire dans un dossier "Plugins", préalablement créé dans les fichiers du projet, ci-joint une vidéo expliquant la mise en place d'un Stream sur une surface 3D : <https://www.youtube.com/watch?v=nNNzUf3zNjM>

Il est temps de créer l'environnement VR, pour ceci il suffit de créer un nouveau projet. Ensuite il faut créer 3 éléments : le Media Source, le Media Player et une surface 3D sur lequel je vais afficher le média (ici une sphère).

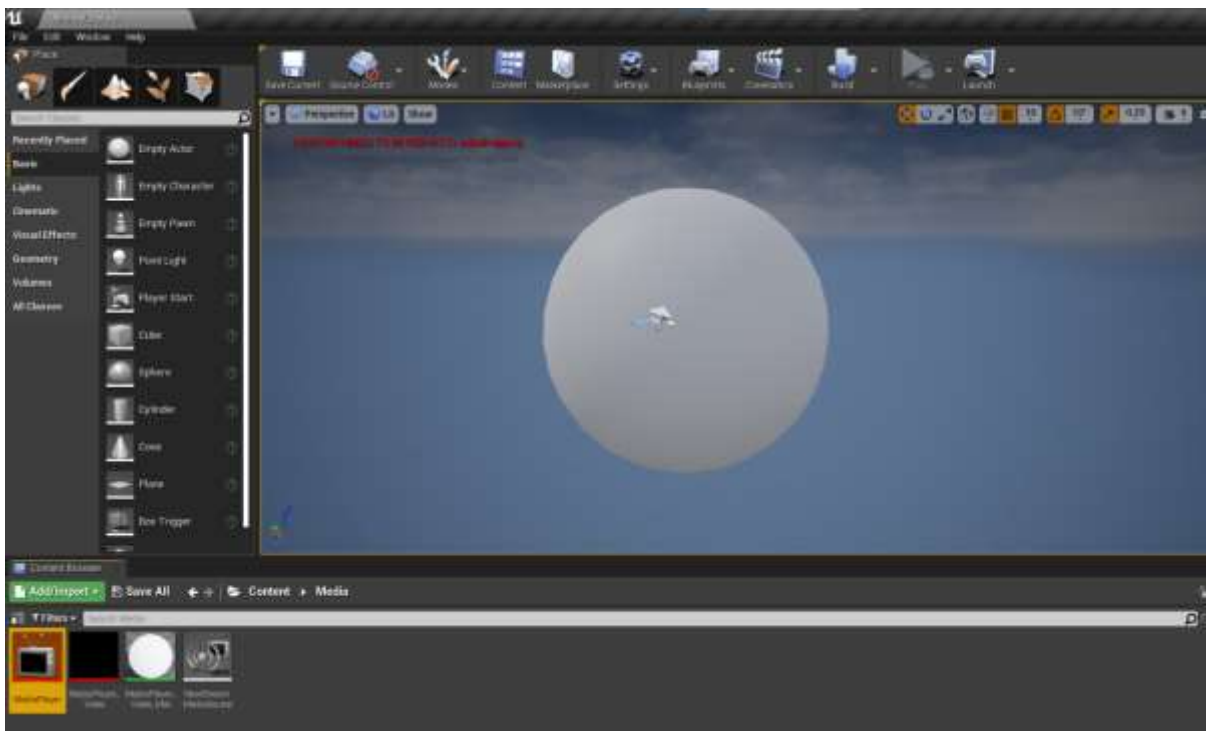


Figure 17: Environnement VR

Afin de configurer Media Source il faut double cliquer sur l'asset, une fenêtre s'ouvre, ensuite sélectionné dans le « player overrides » -> Windows (32 ou 64 bits) le plugin VlcMedia (préalablement installé) et dans la section Stream il faut placer le lien du live .m3u8 qui a été copier plus tôt.



Figure 18: Environnement Unreal

Maintenant il faut s'occuper du Media Player qui permet de lire le Live et d'afficher la vidéo. Pour lancer la vidéo il suffit tous simplement de double cliqué sur la source ou de lancé la lecture avec le symbole approprié.

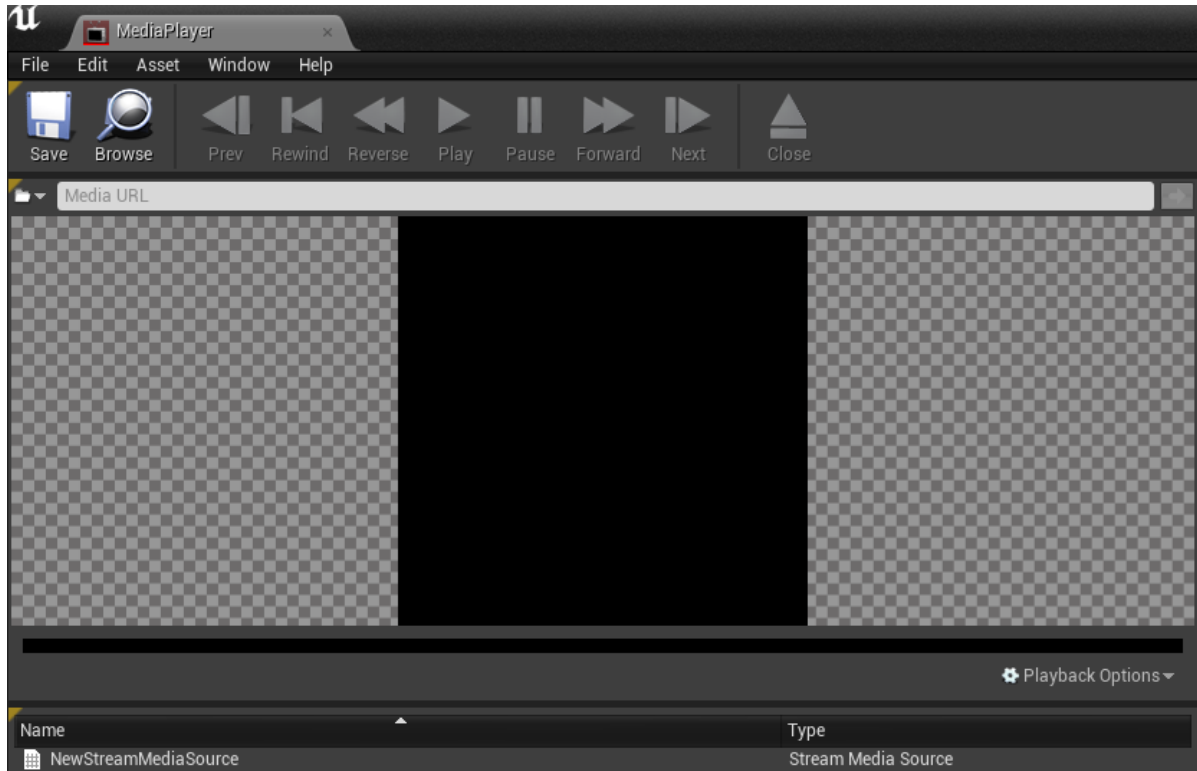


Figure 19: Environnement Unreal 2

Enfin la sphère, donc après avoir créé une sphère on ouvre la configuration afin de pouvoir modifier sa texture et ses paramètres, ici j'ai choisi de mettre les textures du média Player sur la sphère et de paramétrer la sphère pour que les 2 cotés (intérieur et extérieur) puissent diffuser l'image. Remarque si l'image à l'intérieur est à l'envers il suffit de changer les dimensions de la sphere en négatif (passer de 1 à -1 par exemple).

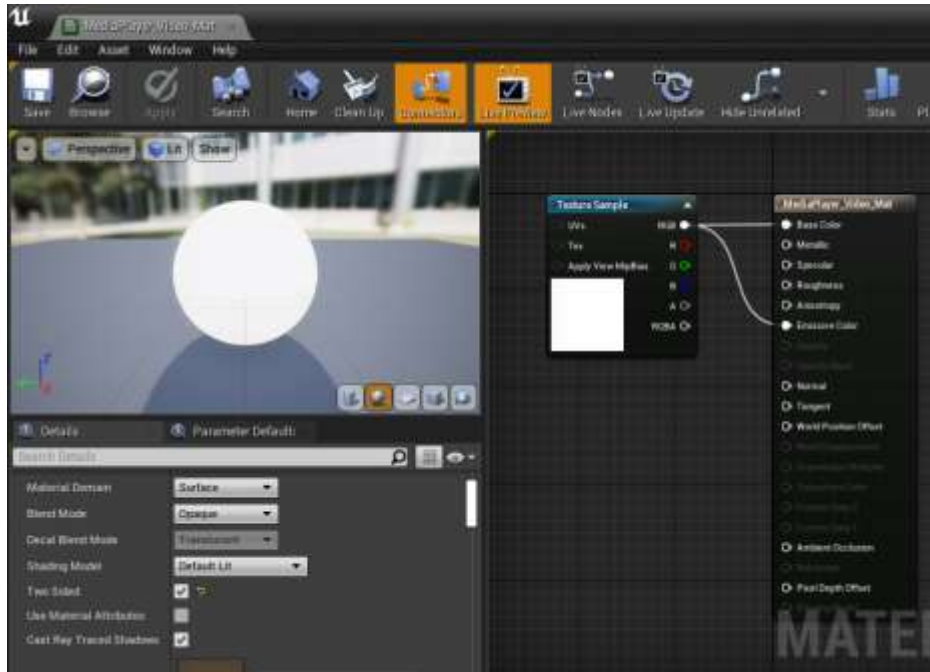


Figure 20: Environnement Unreal 3

Pour terminer et avoir le Stream qui s'affiche au lancement d'Unreal Engine il suffit de programmer via Blueprint (sur Unreal Engine) des conditions et évènement. J'ai utilisé l'évènement BeginPlay pour la connecté à l'Open Source. Et enfin il faut connecter le Media Player en target afin que l'évènement lance la target sélectionné.

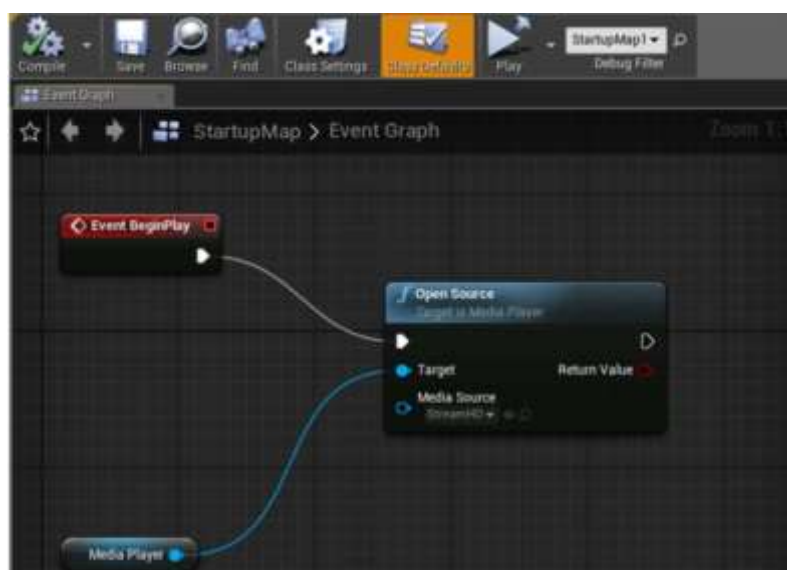


Figure 21: Environnement Unreal 4

4. Partie commandes du drone

A. Idée d'amélioration

Jusqu'à présent, il était possible de piloter le drone avec les gâchettes (Trigger) des manettes de l'HTC VIVE. Cependant, cela n'avait pas d'impact sur la variation de vitesse des moteurs : le drone avançait à la vitesse maximale constamment.

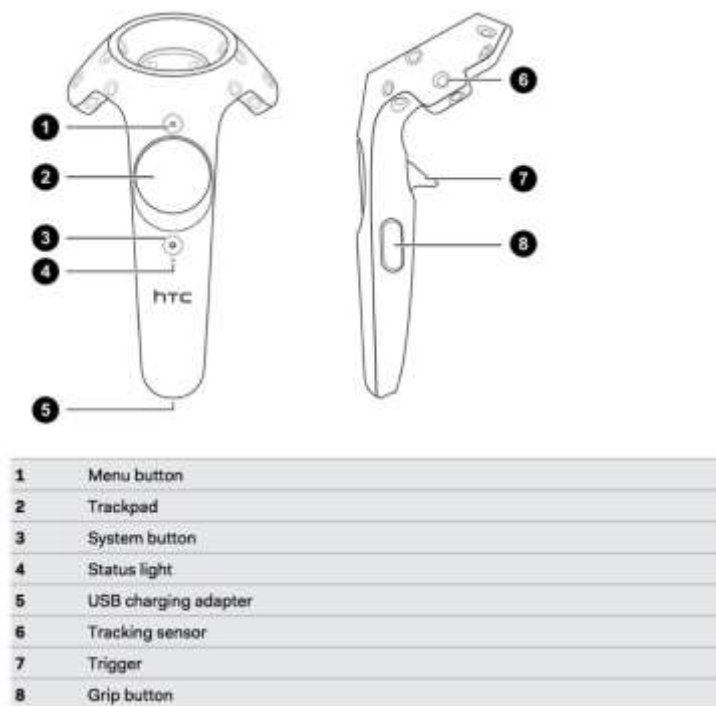


Figure 22: Différentes touches de l'HTC VIVE

L'objectif de notre projet étant de proposer des améliorations par rapport aux différents travaux réalisés jusqu'à présent, j'ai donc décidé d'utiliser les triggers des manettes afin d'avancer le drone tout en jouant sur la vitesse de déplacement.

Afin de concevoir cette amélioration, nous allons donc utiliser le programme de détection des touches réalisé auparavant de manière à récupérer une valeur analogique en fonction de l'appui sur le trigger. Puis, envoyer cette valeur à la Raspberry pour que la variation de vitesse s'effectue au niveau des moteurs du drone terrestre.

B. Conception du programme informatique

Le programme informatique créé repose sur trois parties :

1. Initialisation du client TCP et serveur :

Le protocole de communication utilisé pour pouvoir échanger les données entre l'ordinateur où se trouve le programme et la Raspberry est le TCP / IP. Ce protocole de communication permet d'encapsuler des données sous forme d'un message et va le transmettre sous forme de segments à un serveur.

Afin de créer cette liaison dans le programme, nous allons utiliser la bibliothèque winsock.h . Avec cette bibliothèque, nous allons pouvoir renseigner l'adresse IP du serveur, le port d'écoute et créer les sockets pour l'envoi des données.

Le programme va fonctionner de la manière suivante :

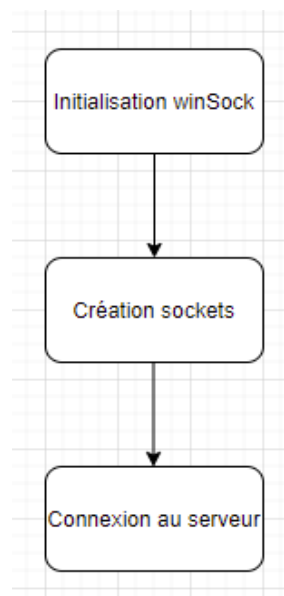


Figure 23: Algorithme code

```

void main()
{
    //int controllerIndex; //The index of the controllers[] array that corresponds with the controller that had a buttonEvent
    EVRInitError eError = VRInitError_None;
    vr_pointer = VR_Init(&eError, VRApplication_Background);

    std::string ipAddress = "10.10.10.104"; // IP Address of the server (10.10.10.104)
    int port = 54000; // Listening port # on the server

    // Initialize Winsock
    WSADATA data;
    WORD ver = MAKEWORD(2, 2);
    int wsResult = WSASStartup(ver, &data);
    if (wsResult != 0)
    {
        std::cout << "Can't start Winsock, Err #" << wsResult << std::endl;
        return;
    }

    // Create socket
    SOCKET sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock == INVALID_SOCKET)
    {
        std::cout << "Can't create socket, Err #" << WSAGetLastError() << std::endl;
        WSACleanup();
        return;
    }

    // Fill in a hint structure
    sockaddr_in hint;
    hint.sin_family = AF_INET;
    hint.sin_port = htons(port);
    inet_pton(AF_INET, ipAddress.c_str(), &hint.sin_addr);

    // Connect to server
    int connResult = connect(sock, (sockaddr*)&hint, sizeof(hint));
    if (connResult == SOCKET_ERROR)
    {
        std::cout << "Can't connect to server, Err #" << WSAGetLastError() << std::endl;
        closesocket(sock);
        WSACleanup();
        return;
    }

    if (eError != VRInitError_None)
    {
        vr_pointer = NULL;
        printf("Unable to init VR runtime: %s\n",
            VR_GetVRInitErrorAsEnglishDescription(eError));
        exit(EXIT_FAILURE);
    }
}

```

Figure 24: Partie du programme correspondant

2. Détection de l’appui sur le Trigger de la manette :

Cette partie du programme va avoir pour rôle de détecter un appui sur le trigger de la manette et récupérer les valeurs analogiques. Pour se faire, nous allons utiliser la bibliothèque OpenVR.h : cela va permettre aux différentes applications d’accéder au matériel VR (ici l’équipement HTC VIVE).

Après avoir ciblé le trigger de la manette, la valeur retournée par la gâchette est comprise entre 0 et 1. Cependant, il va être nécessaire de convertir cette valeur pour pouvoir différencier les manettes. Mais aussi pour que la valeur envoyée au serveur soit adaptée, de manière à pouvoir utiliser le bluetooth de la Raspberry. Nous avons décidé d’utiliser la plage de conversion suivante :

Equipement de l’HTC	Valeurs associées
Manette Droite	1 → 255
Manette Gauche	300 → 554

[illegible]

```
graph TD; A[Détection appui sur manette] --> B[Conversion valeur et attribution à une variable]; B --> C[Conversion valeur en chaîne de caractères]; C --> D[Attribution socket et affichage valeur];
```

20


```

case k_Button_Steering_Trigger:
switch (event.eventType) {
case VKEvent_ButtonPress:

std::cout << "Trigger press!\n";
//message = "Trigger press";
printf("Vidu!\n", event.eventType);

if (role == TrackedControllerRole_Invalid) {
std::cout << "erreur!\n";
//message << "erreur";
}
else if (role == TrackedControllerRole_LeftHand) {
std::cout << "Left!\n";
vr_pointer->GetControllerStateWithPose(trackingInversedStanding, TrackedControllerRole_LeftHand, &controllerState, sizeof(controllerState), &trackedDevicePose);
int t = 1;
while (controllerState.raxis[t].x != 0) {
//nouveau
vr_pointer->GetControllerStateWithPose(trackingInversedStanding, TrackedControllerRole_LeftHand, &controllerState, sizeof(controllerState), &trackedDevicePose);
t = 1;
Left_AnalogValue = 300 + (controllerState.raxis[t].x * 250);
valToSend = Left_AnalogValue;
char Msg[10];
sprintf_s(Msg, "Send", valToSend);
printf("Vidu!\n", Msg);
int sendResult = send(socket, Msg, strlen(Msg), 0);
}
}
else if (role == TrackedControllerRole_RightHand) {
std::cout << "Right!\n";
vr_pointer->GetControllerStateWithPose(trackingInversedStanding, TrackedControllerRole_RightHand, &controllerState, sizeof(controllerState), &trackedDevicePose);
int t = 1;
while (controllerState.raxis[t].x != 0) {
//nouveau
vr_pointer->GetControllerStateWithPose(trackingInversedStanding, TrackedControllerRole_RightHand, &controllerState, sizeof(controllerState), &trackedDevicePose);
t = 1;
Right_AnalogValue = 1 + (controllerState.raxis[t].x * 250);
valToSend = Right_AnalogValue;
char Msg[10];
sprintf_s(Msg, "Send", valToSend);
printf("Vidu!\n", Msg);
int sendResult = send(socket, Msg, strlen(Msg), 0);
}
}
}
}

```

Figure 27: Code commandes manettes

3. Création du socket :

Enfin, la dernière partie du programme consiste à convertir la valeur créée par l'appui sur le trigger en chaîne de caractères. Nous allons donc utiliser la commande « sprintf » pour pouvoir réaliser cette conversion. Lorsque cette conversion est terminée, nous allons intégrer cette chaîne de caractère au socket puis l'envoyer.

```

if (Message.size() > 0) // Make sure the user has typed in something
{
/*commande d origine
int sendResult = send(socket, Message.c_str(), Message.size() + 1, 0);*/
//NOUVEAU
char Msg[10];
sprintf_s(Msg, "%04d", valToSend);
int sendResult = send(socket, Msg, strlen(Msg), 0);
}

```

Figure 28: Code envoi des valeurs

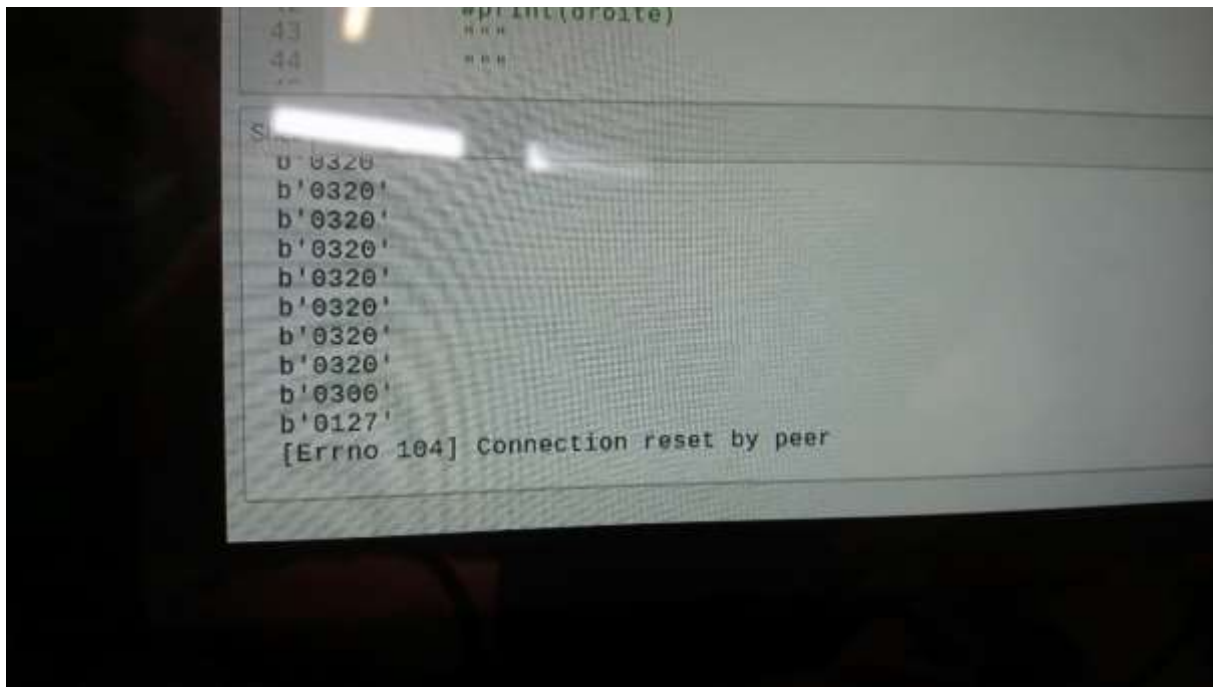


Figure 29: Terminal réception du Raspberry

Sur l'image ci-dessus, nous pouvons voir que la Raspberry réceptionne bien les différentes valeurs en fonction de l'appui (variation en 320 à 127). La partie de ce projet est donc opérationnelle.

Conclusion

Le projet reste à améliorer au niveau de la caméra en réduisant la latence de Steam, puis au niveau du Bluetooth, pouvoir récupérer un code pour utiliser ce Bluetooth ou recrée une carte de contrôle t'rex de toute pièces.

Figure 1 : Diagramme du projet	4
Figure 2: Ancienne communication en I2C	5
Figure 3: Nouvelle communication en Bluetooth	5
Figure 4: Application T'REX.....	6
Figure 5: Bluetooth HC-05 sur carte T'REX.....	6
Figure 6: Bluetooth HC-05 & LCP ISP	7
Figure 7: Trame application.....	8
Figure 8: Explication valeurs application.....	8
Figure 9: Tableau valeurs trame.....	8
Figure 10: Code simulation manettes	9
Figure 11: Caméra bouton wifi.....	11
Figure 12:Cisco bouton WPS	11
Figure 13:Caméra afficheur Wi-Fi actif.....	11
Figure 14: Cisco DHCP IP	12
Figure 15: YouTube configuration de live stream	12
Figure 16: Configuration web server Theta V.....	13
Figure 17: Environnement VR	14
Figure 18: Environnement Unreal	15
Figure 19: Environnement Unreal 2	15
Figure 20: Environnement Unreal 3	16
Figure 21: Environnement Unreal 4	16
Figure 22:Différentes touches de l'HTC VIVE	17
Figure 23: Algorithme code	18
Figure 24: Partie du programme correspondant	19
Figure 25: Affichage terminal commande manettes.....	20
Figure 26: Algorithme 2	20
Figure 27: Code commandes manettes.....	21
Figure 28: Code envoi des valeurs.....	21
Figure 29: Terminal réception du Raspberry.....	22