Une image contenant symbole, logo, Graphique, drapeau

Description générée automatiquement

UNIVERSITE DE KINSHASA

FACULTE POLYTECHNIQUE

RH Monkole

[Adresse de courrier]

Fait par

[Attirez votre lecteur avec un résumé attrayant. Il s’agit généralement d’une brève synthèse du document.   
Lorsque vous êtes prêt à ajouter votre contenu, cliquez ici et commencez à taper.]

**AUTOMATISATION D'UNE VOITURE ROBOT POUR OBTENIR UN SYSTEME SEMI-AUTONOME PILOTE PAR UNE INTERFACE CERVEAU MACHINE.**

# EPIGRAPHE

« Les ordinateurs ne sont pas fiables, les hommes le sont encore moins. À l’origine de chaque erreur attribuée à l’ordinateur, on trouve au moins deux erreurs humaines, dont celle qui consiste à accuser l’ordinateur. »

Loi de non-fiabilité de Gibbs

# DEDICACE

Le terme « héros » n’est pas suffisant pour qualifier celui qui, après avoir vendu peau de sa chair, a passé son existence à souhaiter la venue des jours meilleurs dans votre vie, tout en restant un modèle pour vous malgré les innombrables vagues qu’apporte ce monde. A l’amour, à vous mes parents.

# REMERCIEMENTS

Tous les honneurs reviennent à sa Seigneurie. Raison pour laquelle nous remercions le bon Dieu pour la vie et la grâce qu'il nous a accordé dans la réalisation de ce travail.

Nos remerciements sont aussi adressés :

A nos parents IKULA ELOCK Jean, MIMBU MABWADI Patience pour leurs amours et prières, leurs soutiens financiers et moraux.

A nos frères et soeurs KAYONGO Pathy, IKULA Gabriel, SEKO Aridja, très chère IKULA Grace pour leur assistance.

Aux membres de nos familles MIMBU Constant, MALUMA Marie, KAYONGO Adolphine, KAYONGO Emilienne, les IKULA, les MIMBU, les KAYONGO, les BANGABANGA, les NKEYE pour leurs soutiens de près ou de loin.

A tous les professeurs de la faculté polytechnique de l'Université de Kinshasa qui ont contribué à notre formation.

Aux amis et connaissances ainsi qu'à nos condisciples qui nous ont été utiles d'une manière ou d'une autre tout au long du parcours.

# ABSTRACT

Les interfaces cerveau-ordinateur (BCI) sont des systèmes qui permettent de connecter le cerveau humain à un ordinateur ou à tout autre dispositif externe pour établir une communication silencieuse et sans contact entre les humains et les machines. Ces systèmes sont particulièrement utiles pour des applications médicales, notamment pour les personnes gravement handicapées qui ont perdu toute capacité motrice. En utilisant un BCI, un utilisateur peut envoyer des commandes mentales à un dispositif robotique pour effectuer des tâches spécifiques et gagner un certain niveau d'indépendance. L'objectif de ce travail est de développer un BCI couplé à un système robotique intelligent, partiellement autonome, qui utilise la vision par ordinateur et d'autres capteurs pour faciliter l'interaction entre l'utilisateur et le dispositif robotique contrôlé. Le dispositif robotique choisi est une voiture robotique, équipée d'une caméra, d'un capteur ultrasonique, et d'un micro et d'un écouteur pour créer une télé présence. Le BCI utilisé est basée sur l'expression faciale, qui est une modalité naturelle et intuitive pour contrôler le robot. Le modèle d'apprentissage automatique utilisé pour la détection d'objets est basé sur TensorFlow Lite, qui est une version allégée et optimisée de TensorFlow, adaptée aux systèmes embarqués. Le langage de programmation utilisé est le C++, qui offre des performances élevées et une gestion fine des ressources matérielles. Le traitement de la vision par ordinateur est intégré directement sur le dispositif robotique, en utilisant OpenCV et TFlite, ce qui permet de réduire la latence et la dépendance à une connexion réseau. L'IP call est utilisé pour établir une communication audio bidirectionnelle entre l'utilisateur et le robot, ce qui permet de renforcer le sentiment de présence et d'immersion. Les résultats obtenus montrent que le système développé est capable de contrôler le robot de manière efficace et ergonomique, en utilisant les expressions faciales de l'utilisateur et en tenant compte des informations fournies par les capteurs. Le système présente également un potentiel d'application pour les personnes dans le besoin, notamment les paraplégiques, qui pourraient utiliser le robot comme un moyen de transport et d'exploration, tout en restant en contact avec leur environnement.

# AVANT-PROPOS

La révolution scientifique, telle la quête du saint Graal, a engendré un éveil majeur. La technologie, arme secrète de la science, a propulsé l'humanité à travers le temps et l'espace.

Malgré sa noblesse, la cause scientifique a véritablement fait progresser notre société en résolvant divers problèmes du quotidien. Son influence s'étendant à tous les domaines de la vie, la science devient indispensable pour atteindre le summum du bien-être.

Au 21e siècle, la technologie, résultat de la symbiose entre divers domaines scientifiques, a permis le développement d'Interfaces Cerveau-Machine (ICM) à la fois ludiques, résolutives, et eschatologiques.

Ce travail démontre la puissance du BCI en tant qu'outil d'assistance dans les tâches quotidiennes des hommes, notamment pour ceux ayant perdu l'usage de certains membres. Il s'agit d'une démonstration de faisabilité, un "proof of concept", confirmant la capacité du BCI à maintenir le contrôle sur un équipement capable de s'auto-gérer dans des situations spécifiques.

Notre engagement se focalise sur la résolution de problèmes tout en favorisant l'ergonomie pour les utilisateurs équipés d'un BCI non-invasif, pilotant ainsi un robot en voie d'intelligence croissante.

Avec une confiance profonde dans le processus évolutif, nous aspirons à ce que ce projet serve de fondement pour un avenir meilleur. Nous espérons que l'utilisation des technologies ne nuit pas à l'homme dans le besoin, mais contribue plutôt à son bien-être, l'aidant à retrouver un équilibre dans un environnement sain avec un corps sain.

# INTRODUCTION GENERALE

## PROBLEMATIQUE

La vulnérabilité des personnes handicapées ne nous laisse pas indifférents, étant au courant des difficultés auxquelles ils font face dans la réalisation de telle ou telle autre tâche de leur vie de chaque jour. Ces personnes ont besoin des solutions technologiques adaptées à leur situation, qui leurs permettent de réaliser des activités quotidiennes et de communiquer avec leur entourage. Les axes actuels de la science nous ouvrent un nouveau monde où les applications de la technologie de l'ingénierie servent aussi aux avancées technologiques des biomédicaux. Parmi les technologies émergentes, les BCI offrent de nouvelles perspectives pour développer des solutions médicales basées sur la robotique.

Les Interfaces Cerveau-Ordinateur (BCI) cherchent à établir une communication sans contact entre humains et machines. Elles permettent à un utilisateur d'envoyer des commandes à un ordinateur ou à un dispositif externe en utilisant uniquement son activité cérébrale, mesurée par des capteurs électroencéphalographiques (EEG). Les BCI offrent ainsi une nouvelle modalité d'interaction humain-machine, qui peut être utile dans de nombreux domaines, notamment médical, éducatif, ludique, artistique, etc.[[1]](#footnote-1)

Nous nous focalisons sur les BCI non-invasifs, qui présentent l’avantage d’être plus accessibles, plus adaptés, notamment avec le paradigme asynchrone, et moins coûteux en termes financiers et de préservation des fonctions vitales de l’homme. Parmi les applications potentielles des BCI, l'une des plus importantes est le besoin d'assistance pour les personnes gravement handicapées, en particulier les paraplégiques. Ces personnes souffrent d'une perte totale ou partielle de la capacité motrice, ce qui les empêche de réaliser des activités quotidiennes et de communiquer avec leur entourage.

Les BCI peuvent leur offrir une alternative pour contrôler des technologies d'assistance, telles que des fauteuils roulants, des prothèses, des ordinateurs, des robots[[2]](#footnote-2). Cependant, le développement d'un BCI couplé à un système robotique intelligent pour faciliter l'interaction pose de nombreux défis, tant du point de vue technique que du point de vue de l'expérience utilisateur. Il faut en effet concevoir un système qui soit fiable, robuste, adaptatif, ergonomique, et qui respecte les besoins et les préférences de l'utilisateur. Il faut également prendre en compte les contraintes liées à l'environnement, à la connexion, à la sécurité, à l'éthique, etc.

## INTERET DU SUJET

L’intérêt du sujet réside dans la question de savoir comment utiliser les interfaces cerveau-machine (BCI) pour contrôler un système robotique intelligent, partiellement autonome, qui puisse assister les personnes handicapées dans leur mobilité et leur interaction avec leur environnement. En effet, prendre indéfiniment contrôle d’un objet s’avère une tâche épuisante au fur et à mesure que nous sommes obligés d’attribuer toute notre attention et énergie sur la chose sans perdre de vue tout ce qui peut arriver dans l’environnement où nous manifestons ce contrôle. Il est donc nécessaire de concevoir un système qui soit capable de s’adapter aux situations changeantes, de prendre des décisions autonomes, et de communiquer avec l’utilisateur.

Notre travail, intitulé AUTOMATISATION D’UNE VOITURE ROBOT POUR OBTENIR UN SYSTEME SEMI-AUTONOME PILOTE PAR UNE INTERFACE CERVEAU MACHINE, est l’étude d’un proof of concept sur les ajouts des capacités sur le robot contrôlé, comme l’a montré le travail de Pierre Sedi[[3]](#footnote-3) sur lequel nous nous basons, qui pourra être habile et apte de prendre des décisions face à des circonstances événementielles. Nous nous inspirons de son approche qui utilise l’expression faciale comme modalité de contrôle du robot, offrant une méthode naturelle et intuitive. Nous cherchons à améliorer son système en utilisant la vision par ordinateur et d’autres capteurs pour doter le robot d’une intelligence artificielle, qui lui permette de naviguer de manière semi-autonome, de détecter les obstacles, et de réagir en conséquence.

Le travail se veut être la réponse à un cahier de charge, que nous découvrirons au chapitre 2, qui nous a été donné pour ouvrir un champ d’application sur les travaux des BCI asynchrones, insistant sur la facilité et l’ergonomie de l’utilisateur.

## DELIMITATION DU SUJET

Notre travail s’inscrit dans le domaine des interfaces cerveau-machine (BCI) couplées à des systèmes robotiques intelligents, qui visent à offrir une solution d’assistance aux personnes handicapées. Nous nous intéressons plus particulièrement au smart video car, une voiture robotique contrôlée par un BCI basé sur l’expression faciale, qui utilise la vision par ordinateur et d’autres capteurs pour naviguer de manière semi-autonome[[4]](#footnote-4).

Après avoir exposé l’intérêt de notre travail, nous allons nous limiter dans l'étude expérimentale des comportements que nous attribuons au robot, sa réactivité lorsqu’il reçoit des commandes, sa fluidité dans le rendu visuel des images qu’il capte avec ses capteurs, son efficacité face à la perte de connexion avec l’utilisateur Les défis majeurs du travail se trouvent dans la gestion des ressources que nous offre le RPI 4, qui est l’élément central, la partie intelligente, le cerveau du robot.

Nous balayons les notions apprises en systèmes embarqués, traitement des signaux, réseau, analyse des circuits, programmation, génie logiciel, systèmes à multiprocesseur et machines parallèles, machine learning, operating system, computer design, électroniques, mesures électriques, etc.

## OBJECTIFS

Le flux d'information enregistré à partir de la problématique, de l’intérêt, et de la délimitation du sujet, nous pousse à définir les objectifs principaux de notre travail, qui sont :

- Concevoir un système robotique intelligent contrôlé par un BCI qui, utilisant le paradigme des expressions faciales pour extraire des signaux EEG de l’individu, permette à l’utilisateur de piloter à distance une voiture robotique, en tenant compte des informations fournies par les capteurs.

- Implémenter les comportements du robot en approche des obstacles, en situation d’embourbement, ou en cas de perte de connexion.

- Évaluer la performance du système et la satisfaction des utilisateurs à partir des expériences menées au laboratoire.

## PLAN DU TRAVAIL

Nous avons subdivisé notre travail en quatre (4) chapitres qui sont :

1. Généralités sur le cerveau humain et le BCI
2. Conception du système
3. Outils et méthodologies
4. Expérimentations et résultats

C’est par là que nous mettons fin à l’introduction générale, nous vous invitons à passer du bon temps dans la lecture de la suite du travail.

# CHAPITRE 1 : GENERALITES SUR LE CERVEAU HUMAIN ET LE BCI

## LE CERVEAU HUMAIN

Cet organe miracle ne sait nullement cesser d’étonner l’homme moderne en dépit de son éventail de connaissance qui prend le large au fil des années. Notre humble tâche consiste à avoir une idée commune sur le concept du cerveau humain partant, bien évidemment, des définitions attribuées à cet organe dans d’autres domaines de la science. Nous en décrivons certains :

**Philosophiquement**, le cerveau est l’instrument qui nous permet de percevoir, de réfléchir et d’agir. Il est le moteur qui donne un sens à notre existence.

**Sociologiquement**, le cerveau est le maestro de notre organisme, le dirigeant tout en se dirigeant lui-même. Il est à l’origine de nos comportements et, par conséquent, de nos interactions avec les autres membres de la société.

**Scientifiquement**, le cerveau est un domaine clé car il recèle encore de nombreux mystères, que ce soit dans son développement, son fonctionnement normal et pathologique, ou ses capacités d’adaptation. Il est crucial aujourd’hui de déchiffrer les mécanismes qui sous-tendent nos capacités intellectuelles, nos émotions et les comportements moteurs qui en découlent.

En tant qu’ingénieur, ces perspectives sont essentielles pour comprendre comment nous pouvons concevoir des technologies, comme les interfaces cerveau-machine, qui peuvent interagir efficacement et de manière significative avec ces aspects complexes du cerveau humain.

## QUID LE CERVEAU HUMAIN ? CONSTITUTION

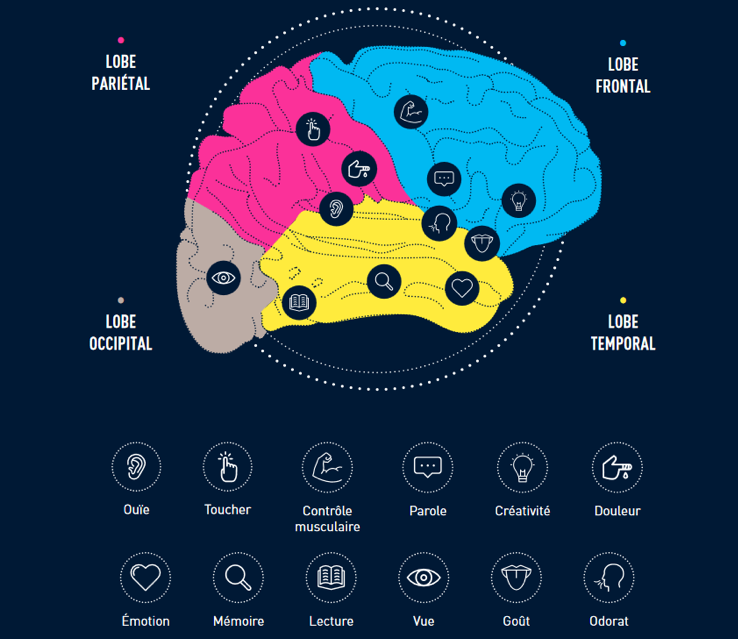
Au cœur de notre être, se trouve un orchestre complexe et harmonieux, le système nerveux central, composé du cerveau et de la moelle épinière. Il est le maître d’œuvre qui intègre les informations, commande nos mouvements et assure nos fonctions cognitives.

Le cerveau, pesant environ 1,3 kg et composé à 75% d’eau, est l’organe le plus protégé de notre corps. Il est enveloppé par trois couches protectrices, les méninges, et baigne dans le liquide céphalo-rachidien, qui amortit les chocs. Malgré sa petite taille, il consomme 15 à 20% de l’énergie produite par notre corps, principalement sous forme de glucose. Un réseau dense de vaisseaux sanguins le traverse, assurant un apport constant en oxygène.

Le cerveau est divisé en deux hémisphères, droit et gauche, reliés par le corps calleux. Chaque hémisphère est composé de plusieurs lobes : le lobe frontal, siège du raisonnement, du langage et de la coordination motrice volontaire ; le lobe pariétal, qui nous permet de prendre conscience de notre corps et de l’espace environnant ; le lobe occipital, qui intègre les messages visuels ; le lobe temporal, centre de l’audition, de la mémoire et des émotions ; le lobe limbique, qui traite les informations relatives aux émotions, aux affects et à la mémoire ; et le lobe de l’insula, qui nous permet de ressentir la douleur, les odeurs et le goût.

Enfin, le cervelet, qui contrôle notre équilibre et la coordination de nos mouvements, et le tronc cérébral, qui sert de point de passage entre les hémisphères cérébraux et la moelle épinière, complètent ce tableau.

Cette narration met en lumière la complexité et la beauté de notre cerveau, un organe qui, malgré les progrès de la science, conserve encore de nombreux mystères à dévoiler.



Le cerveau humain[[5]](#footnote-5)

Figure 1. Le cerveau humain et la répartition des fonctions par lobe[[6]](#footnote-6)

## INTERET DU CERVEAU DANS LE PRESENT TRAVAIL

Le cerveau, une merveille de la nature, est un réseau complexe de 100 milliards de neurones, ces cellules nerveuses qui sont les unités de base de notre système nerveux. Ces neurones sont reliés par des axones, des prolongements qui sont protégés par une gaine appelée myéline. Cette myéline est produite par des cellules spéciales appelées oligodendrocytes, qui font partie d’un groupe de cellules connues sous le nom de cellules gliales, tout aussi nombreuses que les neurones.

Le cerveau est structuré en différentes régions, dont le cortex, ou substance grise, qui est la couche externe du cerveau et contient les corps cellulaires des neurones. En dessous du cortex se trouve la substance blanche, qui est composée des axones des neurones, enveloppés dans leur gaine de myéline. Le cerveau abrite également quatre cavités appelées ventricules cérébraux, où circule le liquide cérébrospinal. Au centre du cerveau, les noyaux gris centraux, également connus sous le nom de ganglions de la base, jouent un rôle crucial dans le contrôle du comportement et de l’apprentissage.

Le cerveau est un expert en communication. Les neurones communiquent entre eux par des signaux électriques, appelés influx nerveux. Ces signaux se propagent le long des axones jusqu’à la terminaison synaptique, où ils déclenchent la libération de substances chimiques appelées neurotransmetteurs. Ces neurotransmetteurs peuvent activer ou inhiber un autre neurone, permettant ainsi à l’influx nerveux de continuer son chemin.

Il existe plusieurs types de neurotransmetteurs, certains excitateurs comme le glutamate, d’autres inhibiteurs comme le GABA. Parmi les plus connus, on trouve la dopamine, la sérotonine, l’histamine et l’acétylcholine. Les neurones qui produisent la dopamine, situés dans une région profonde du cerveau appelée la substance noire, sont essentiels pour le contrôle du mouvement.

Ces connaissances sur le fonctionnement du cerveau sont fondamentales pour le développement des interfaces cerveau-machine (ICM). Les ICM sont des dispositifs qui permettent une communication directe entre le cerveau et un système externe, comme une voiture robotisée. En captant et en interprétant les signaux électriques du cerveau, les ICM peuvent traduire les intentions de l’utilisateur en commandes pour le système externe, permettant ainsi un contrôle direct par la pensée. C’est un domaine de recherche passionnant et prometteur, qui pourrait révolutionner notre interaction avec la technologie.

# CHAPITRE 2 : CONCEPTION DU SYSTEME

## INTRODUCTION

Parler du système semi-autonome englobant à la fois le BCI, censé donner des commandes strictes à un robot, et la piCar-V, capable de se prendre seul en charge face à des situations étudiées dans ce travail, est une vision répondant à un cahier de charge qui nous a été confié après avoir vu les résultats de la totale direction menée seule par Cerebro.

Les objectifs de ce chapitre seront alors tirés directement du cahier de charge, tel est le socle qui nous mène à l’élaboration et à la réalisation de cette œuvre qualifiée de proof of concept pour l’ingénierie du futur.

## OBJECTIFS

Mettre en exergue d'une manière analytique et méthodique la capacité de la gestion BMI à faire une auto-analyse (debugging) des pannes/difficultés quand le robot est bloqué, à travers aussi un stress test.

Fiabilité (reliability) du contrôle BMI :  Quels sont les problèmes de connexion et action quand le robot se trouve à la limite de son rayon d'action.

Voulant répondre à l’implémentation de ces 2 paradigmes tout en se basant sur les moyens à notre disposition et qui pourraient matcher parfaitement avec le système BCI existant, nous nous sommes donnés des réponses concluant à la réalisation d’un système semi-autonome, qui selon nous, offrirait beaucoup d’élégance, de simplicité et d’ergonomie aux utilisateurs plutôt que de se baser à 1000% à l’utilisation unique du casque.

## CONCEPTION DU SYSTEME

Les règles de conception des systèmes automatiques se subdivisent en 2 grandes parties qui sont : la partie opérative et la partie commande.

Dans ce chapitre nous définissons la partie opérative comme étant l’ensemble des acteurs majeurs à la constitution de notre système. Ce sont des entités qui entre en contact les uns des autres afin de réaliser cette symbiose recherchée.

La partie commande désigne l’ensemble des éléments logiciels qui déterminent le comportement du système, les modèles, et les objets qui nous permettent de bien suivre la réalisation du système. Dans cette section, nous allons présenter la conception de notre système robot autonome avec BCI en détaillant ces deux parties.

### PARTIE OPERATIVE

La partie opérative de notre système se compose de deux sous-systèmes principaux : le BCI et le robot.

Le BCI est l’interface qui permet de traduire l’activité cérébrale de la personne qui conduit le robot en commandes pour le robot. Le robot est le système qui reçoit les commandes du BCI et qui les exécute en utilisant ses propres capacités. Les composantes de chaque sous-système sont dans le prochain chapitre qui leur a été dédié.

Le robot

### PARTIE COMMANDE

La partie commande est celle qui apporte l’âme dans notre conception, elle décrit le fonctionnement intelligent du système et elle détermine en avance quels seraient les paramètres à prendre en compte quand on parlera des expérimentations et des résultats que nous souhaiterons présenter.

Nous voulons concevoir un système qui illumine les aspects suivants subdivisés en code :

**Premier code (Capteur ultrasonique et traitement d'image sur Raspberry Pi)**

- Entité : Raspberry Pi et capteur ultrasonique HC-SR04.

- Tâches : Mesure de distance, détection d'obstacles, et streaming vidéo. Utilise TensorFlow Lite pour le traitement d'image et la détection d'objets avec OpenCV.

- Apport en métriques :

a. Précision : Amélioration grâce à la détection d'obstacles et la classification d'objets en temps réel.

b. Réactivité : Évaluation basée sur le temps de réponse du système à un obstacle détecté.

c. Efficacité : Mesurée par la capacité à naviguer sans collisions en utilisant les données ultrasoniques et visuelles.

**Deuxième code (Serveur HTTP pour le contrôle du PiCar)**

- Entité : Serveur HTTP sur Raspberry Pi.

- Tâches : Réception des commandes de direction et de mouvement via HTTP, et contrôle du PiCar-V en conséquence.

- Apport en métriques :

a. Fiabilité : Basée sur la réussite des commandes envoyées au PiCar-V sans erreurs.

b. Latence : Temps de réponse du serveur à une commande HTTP.

c. Sécurité : Implique la robustesse du serveur à gérer les demandes sans crasher.

**Troisième code (Interface Cerveau-Ordinateur pour le contrôle du PiCar)**

- Entité : Interface Cerveau-Ordinateur (BCI) utilisant OpenViBE.

- Tâches : Traduction des signaux EEG en commandes de contrôle pour le PiCar-V via le serveur HTTP.

- Apport en métriques :

a. Précision : Exactitude de la classification des signaux EEG en commandes spécifiques.

b. Rapidité : Délai entre la pensée de l'utilisateur et la réaction du PiCar-V.

c. Expérience utilisateur : Mesurée par la facilité d'utilisation et la satisfaction de l'utilisateur en contrôlant le PiCar-V avec le mental.

Ces résumés montrent comment chaque code contribue à l'ensemble du projet en termes de métriques spécifiques, en indiquant clairement les responsabilités des différentes entités (capteurs, serveur, et interface BCI) et leur impact sur la performance globale du système.

### DIAGRAMMES

Pour illustrer la conception de notre système robot autonome avec BCI, nous allons présenter deux types de diagrammes : un diagramme de blocs fonctionnels, qui montre les relations entre les différents composants du système, les flux de données et de signaux, et les fonctions réalisées par chaque bloc ; et un diagramme de séquence, qui montre les interactions entre les différents composants du système, les événements et les messages, et le déroulement temporel du système. Ces diagrammes sont basés sur les éléments de conception que nous avons décrits précédemment.

* Le diagramme de blocs fonctionnels : Le diagramme de blocs fonctionnels de notre système est le suivant:

st=>start: Début

e=>end: Fin

bci=>operation: BCI

eeg=>inputoutput: EEG

cmd=>inputoutput: Commandes

rob=>operation: Robot

cam=>operation: Caméra

us=>operation: Capteur ultrason

mic=>operation: Microphone

spk=>operation: Haut-parleur

mbv=>operation: Modèle MobileNetV1

ctrl=>operation: Contrôleur neuronal

mot=>operation: Moteurs

dir=>operation: Direction

env=>inputoutput: Environnement

scr=>inputoutput: Écran

hum=>inputoutput: Humain

st->eeg->bci->cmd->rob

rob->cam->scr

rob->us

rob->mic->spk->hum

rob->mbv->ctrl->mot->env

rob->ctrl->dir->env

env->eeg

env->rob

env->hum

hum->mic

scr->eeg

mot->e

dir->e

spk->e

Ce diagramme montre que le BCI reçoit les signaux EEG de la personne qui conduit le robot, et les transforme en commandes pour le robot. Le robot reçoit les commandes du BCI, et les exécute en utilisant les technologies suivantes:

* Caméra : Permet de capturer les images de l’environnement et de les transmettre à l’écran de la personne qui conduit le robot.
* Capteur ultrason : Permet de mesurer la distance aux obstacles et de modifier la trajectoire du robot en cas de besoin.
* Microphone : Permet de capter les sons de l’environnement et de les transmettre au haut

# CHAPITRE 3 : OUTILS ET METHODOLOGIES

## INTRODUCTION

Après avoir conçu notre système, nous nous basons sur les outils à prendre en compte pour nous permettre de nous conformer à la réalisation des objectifs arrêtés pour la réalisation de notre cahier de charge.

Ce chapitre parle de tous les constituants de notre système avec ses 2 parties majeures, la partie BCI et la partie voiture robot, et puis des méthodologies utilisées pour avoir un système robuste. Le chapitre contiendra donc deux sections qui, en intégrant les parties du système, détaillera tout ce qui est lié à l’aspect hardware et software du système conçu.

## OUTILS

Sachant que nous énumérons chaque élément du système, nous allons parler de prime à bord du hardware pour chuter au software.

### HARDWARE

1. EMOTIV EPOC+

EMOTIV EPOC+ L'EMOTIV EPOC+ est un dispositif de Brain-Computer Interface (BCI) avancé développé par la société EMOTIV. Il s'agit d'un appareil portable non invasif conçu pour mesurer l'activité électrique du cerveau (électroencéphalographie ou EEG) en temps réel. Il est équipé de 14 électrodes disposées selon le système international 10-20, qui couvrent les zones corticales impliquées dans la réalisation de tâches motrices imaginées. Ces électrodes captent les signaux électriques du cerveau et permettent de détecter les différentes activités cérébrales.

L'appareil est conçu pour être confortable à porter, avec un design ergonomique qui s'adapte à la forme de la tête de l'utilisateur. Cela permet une utilisation prolongée sans inconfort. Il utilise la technologie sans fil pour se connecter à un ordinateur ou à d'autres périphériques compatibles. Cela offre une liberté de mouvement à l'utilisateur et permet une utilisation pratique dans divers environnements.

EMOTIV fournit une gamme d'applications logicielles qui permettent de traiter et d'analyser les données EEG capturées par l'EMOTIV EPOC+.

Ces applications permettent notamment de visualiser les signaux cérébraux, de détecter des modèles d'activité et de contrôler des appareils externes en utilisant la pensée.



Casque Emoiv Epoc+

1. SUNFOUNDER SMART VIDEO CAR

C’est un kit de voiture robotique qui peut être contrôlé à distance par un Raspberry Pi. Il dispose d’une caméra, de quatre roues motrices, d’un capteur à ultrasons et d’un servomoteur.

[Il peut réaliser des fonctions telles que l’évitement d’obstacles, le streaming vidéo en temps réel et la reconnaissance d’objets](https://www.maxicours.com/se/cours/fonctionnement-des-composants-d-un-ensemble/). Raison pour laquelle nous l’avons reconduit dans notre travail comme robot principal où s’assoit toute la structure réalisée.

Sans trop détaillé les éléments constituant notre robot, comme l’a déjà fait notre ami Pierre dans son travail de mémoire, l’image suivante montre l’entière composition du piCar-V.



Equipement SunFounder Picar-V

1. RASPBERRY PI 4 MODEL B

Le RPI 4, comme nous aimons bien l’appeler, est un mini-ordinateur de la taille d’une carte de crédit qui peut exécuter divers systèmes d’exploitation. Il dispose d’un processeur quadricœur, de 2 Go de mémoire vive, de quatre ports USB (2 USB en noir : USB 2.0 qui permettent un transfert de données limité à 480 Mbps. 2 USB en bleu : USB 3.1 qui permettent de transférer les données à une vitesse de 10 Gbps, essentiels pour les données en temps réel), d’un port Ethernet, de 2 ports micro HDMI, d’un port audio, d’un lecteur de carte microSD et d’une connectivité sans fil. Il p ut être utilisé comme un ordinateur de bureau, un serveur, un centre multimédia ou une plateforme de développement.



Raspberry Pi 4

1. ACCESSOIRES

Ici nous avons voulu spécifier quelques éléments, quoique non cités dans les composants majeurs du système, qui jouent un rôle crucial dans le fonctionnement du système.

Ces éléments qui complètent le hardware, comprennent les batteries, les chargeurs, les câbles, l’adaptateur et les supports. Les batteries fournissent l’énergie nécessaire au fonctionnement du hardware. Les câbles permettent de connecter les différents éléments du hardware. Les adaptateurs permettent de convertir les signaux ou les tensions. Les supports permettent de fixer ou de positionner le hardware.

1. WEBCAM EXTERNE

Externe car ne faisant pas partie de l’arsenal prévu pour le picar-V, nous avons porté le choix sur un dispositif qui nous permettra d’avoir un micro intégré dans la caméra afin de bien jouer le rôle de webcam dans notre application sur l’IP call entre le robot et l’utilisateur. Le modèle utilisé est presqu’identique à celui présenté à la figure et à prix beaucoup plus bas.

C’est une caméra externe qui peut être connectée au Raspberry Pi via un port USB. Elle permet de capturer des images et des vidéos en haute définition. Elle dispose d’un microphone intégré, d’un autofocus, d’un zoom numérique et d’une correction automatique de l’éclairage, et l’annulaire s’allume au contact.



Webcam

1. BAFFLE POUR SORTIE AUDIO RPI 4

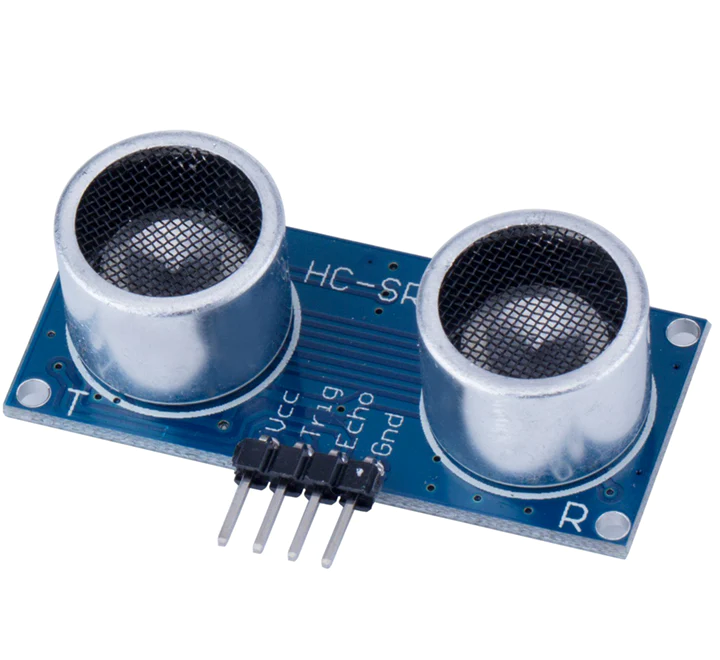
C’est un haut-parleur qui peut être connecté au Raspberry Pi via un port audio. Il permet de diffuser du son en stéréo. Il dispose d’un contrôle du volume, d’une prise casque et d’une entrée auxiliaire.



Baffle

1. HC-SR04

C’est un capteur à ultrasons qui permet de mesurer la distance entre le capteur et un objet. Il émet un signal sonore à haute fréquence et mesure le temps que met l’écho à revenir. Il peut détecter des objets situés entre 2 cm et 4 m.



Ultrason

1. ESTIMATION COUT

Nous rappelons que la plupart des composants physiques utilisés dans notre système sont identiques à ceux utilisés dans le travail intitulé « Conception d’une interface cerveau-machine temps-réel asynchrone pour le pilotage à distance des systèmes robotiques à l’aide du Casque Emotiv EPOC+ ».

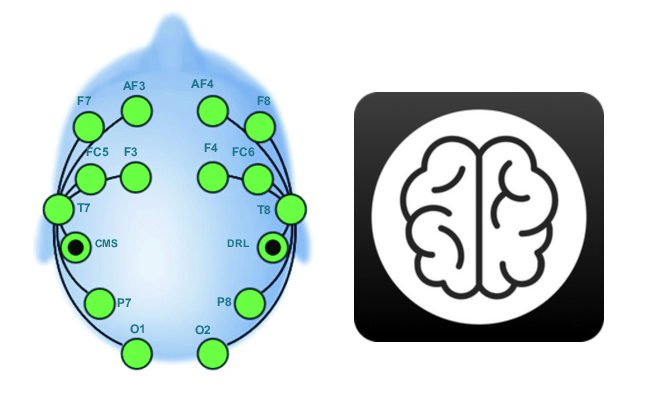
D’où le coût n’a pas beaucoup bougé dans ce tableau :

|  |  |  |  |
| --- | --- | --- | --- |
| Matériel | Quantité | Prix unitaire ($) | Prix total ($) |
| Emotiv EPOC+ | 1 | 699 | 699 |
| SunFounder Smart Video Car | 1 | 100 | 100 |
| Raspberry Pi 4B | 1 | 45 | 45 |
| Batterie 18650 8000 mAh 4.2V | 4 | 2 | 8 |
| Chargeur pour batterie 18650 | 1 | 4 | 4 |
| Carte microSD 32G + adaptateur | 1 | 4 | 4 |
| Ultrasonic HC-SR04 | 1 | 3 | 3 |
| Webcam externe | 1 | 10 | 10 |
| Baffle | 1 | 5 | 5 |
| Total | 12 |  | 878 |

### 3.2.2. SOFTWARE

1. EMOTIV APPS

Ce sont des applications qui permettent de configurer, de visualiser et d’analyser les données du casque EMOTIV EPOC+. Elles comprennent EMOTIV App, EMOTIV Xavier, EMOTIV Insight et EMOTIV BCI. Elles sont écrites en C#, Java, Python et JavaScript.



Vue des électrodes dans Emotiv Launcher

Emotiv Launcher est l’application que nous utilisons au switch on du casque, elle nous permet de vérifier l’état du casque sur la tête de l’utilisateur, savoir si le contact est bon et si le pourcentage des signaux EEG lus est acceptable.

Fig. Qualité de contact et des signaux émis du casque Emotiv EPOC+

Fig. Icône Emotiv Launcher

1. OPENVIBE

C’est une plateforme logicielle qui permet de concevoir, de tester et d’utiliser des interfaces cerveau-ordinateur (BCI). Elle permet de traiter les signaux EEG, de les classifier, suivant un épochage, de les filtrer et de les convertir en commandes. Elle est écrite en C++ et utilise des bibliothèques comme Qt, OpenGL et OpenAL.



OpenVibe

1. CYKIT

C’est un outil qui permet de connecter le casque EMOTIV EPOC+ au Raspberry Pi via un port USB. Il permet de recevoir les données brutes du casque et de les transmettre à OPENVIBE via un protocole TCP/IP. Il est écrit en Python et utilise des bibliothèques comme PyUSB, PyCrypto et Socket.

1. INTERPRÉTEUR PYTHON

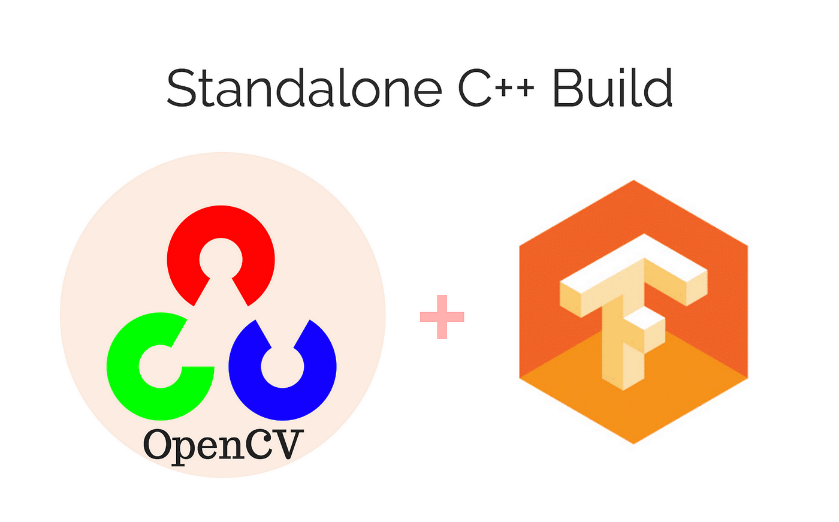
C’est un programme qui permet d’exécuter des scripts écrits en Python. Il permet de réaliser des tâches de programmation de haut niveau, comme le calcul, la manipulation de données, le contrôle du hardware, etc. Il est écrit en C et utilise des bibliothèques comme NumPy, SciPy, Matplotlib, etc.

1. LIBRAIRIE SMART VIDEO CAR KIT

C’est une bibliothèque qui permet de contrôler le kit de voiture robotique SunFounder Smart Video Car. Elle permet de gérer les mouvements des roues, la rotation du servomoteur, la capture de la caméra, etc. Elle est écrite en Python et utilise des bibliothèques comme RPi.GPIO, Picamera, etc.

1. OPENCV

C’est une bibliothèque qui permet de réaliser des opérations de vision par ordinateur. Elle permet de traiter les images et les vidéos, de détecter des objets, des visages, des gestes, etc. Elle est écrite en C++ et utilise des bibliothèques comme Numpy, Scipy, Matplotlib, etc.



Trio C++, OpenCV et TensorflowLite

1. TENSORFLOW LITE

C’est un framework qui permet de déployer des modèles d’apprentissage automatique sur des appareils à faible puissance, comme le Raspberry Pi. Il permet de réaliser des tâches de reconnaissance d’images, de classification, de détection, etc. Il est écrit en C++ et utilise des bibliothèques comme Eigen, Gemmlowp, etc.

1. MOBILENETV1

C’est un modèle d’apprentissage automatique qui permet de réaliser de la reconnaissance d’objets sur des images. Il utilise une architecture de réseau de neurones convolutifs optimisée pour les appareils mobiles. Il peut reconnaître plus de 1000 classes d’objets différents. Il est écrit en Python et utilise des bibliothèques comme TensorFlow, Keras, etc.

1. C++

C’est un langage de programmation qui permet de réaliser des applications de bas niveau, comme le contrôle du hardware, le traitement des signaux, l’optimisation des performances, etc. Il utilise des bibliothèques comme STL, Boost, Qt, etc.

1. SHELL

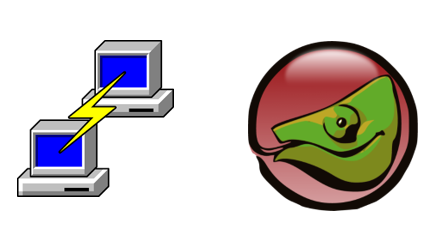
C’est un langage de script qui permet de réaliser des tâches de gestion du système, comme le lancement de programmes, la manipulation de fichiers, la communication entre processus, etc. Il utilise des commandes comme ls, cd, ps, etc.

1. WIRINGPI

C’est une bibliothèque écrite en C conçue pour faciliter l'accès aux GPIO (General Purpose Input/Output) du Raspberry Pi. Elle permet aux développeurs d'écrire des scripts et des programmes en C (et autres langages compatibles) qui interagissent directement avec les broches d'entrée et de sortie du Raspberry Pi, facilitant le contrôle de composants hardware tels que des LEDs, des moteurs, des capteurs, etc. WiringPi imite la fonctionnalité et la syntaxe de la bibliothèque Arduino Wiring, rendant la transition des projets Arduino vers Raspberry Pi plus intuitive pour les développeurs.

1. PUTTY

PuTTY est un émulateur de terminal open-source et gratuit, permettant une connexion sécurisée via SSH (Secure Shell) à des systèmes distants, tels que le Raspberry Pi. Il supporte également les connexions via Telnet, rlogin et les connexions série. Cet outil est crucial pour la configuration initiale, le débogage et la maintenance à distance du Raspberry Pi, offrant aux développeurs un moyen efficace de gérer leur système sans nécessiter un accès physique direct.



Putty et K-meleon Browser

1. K-MELEON

K-Meleon est un navigateur Web léger et personnalisable pour Windows, basé sur le moteur de rendu Gecko développé par Mozilla. Bien qu'il ne soit pas directement utilisé sur le Raspberry Pi, il peut être utilisé sur des systèmes de contrôle ou de monitoring fonctionnant sous Windows, pour accéder à des interfaces web du système, telles que des dashboards de contrôle ou des interfaces de programmation pour le système BCI et la voiture robot. Sa légèreté en fait un choix idéal pour des systèmes aux ressources limitées, où l'efficacité et la rapidité sont essentielles.

Le choix est penché sur K-meleon comme browser sur ce projet à cause de sa légèreté, sa faible utilisation de la RAM et sa faible consommation des ressources du processeur.

### INVENTAIRE ENERGETIQUE DES COMPOSANTS PHYSIQUES

L'évaluation de la consommation énergétique de chaque composant matériel est essentielle pour garantir l'autonomie et l'efficacité du système dans son ensemble. Cette section présente une analyse de la consommation énergétique des principaux composants matériels utilisés dans notre système.

A. EMOTIV EPOC+

* Consommation Énergétique: Environ 1,4 W
* Source d'Alimentation: Batterie rechargeable intégrée
* Autonomie: Jusqu'à 12 heures en utilisation continue

L'EMOTIV EPOC+ est optimisé pour une consommation énergétique faible, ce qui est essentiel pour les sessions prolongées d'utilisation. Sa batterie rechargeable garantit une utilisation continue sans nécessiter de changements fréquents ou de recharges, ce qui est crucial pour des applications mobiles ou à distance.

B. Raspberry Pi 4 Model B

* Consommation Énergétique: Environ 6 W (peut varier selon l'utilisation)
* Source d'Alimentation : Adaptateur secteur de 5 V et 3 A
* Autonomie : Non applicable (alimenté par une source d'énergie externe)

Bien que le Raspberry Pi 4 soit plus puissant que ses prédécesseurs, il maintient une consommation énergétique relativement basse. Cela permet son utilisation dans des projets nécessitant une certaine puissance de calcul sans un impact significatif sur la consommation énergétique totale du système.

C.SUNFOUNDER Smart Video Car

* Consommation Énergétique : Variable selon les opérations
* Source d'Alimentation: Batteries 18650
* Autonomie: 2-3 heures selon l'utilisation

La voiture robot SunFounder est conçue pour une utilisation modérée de l'énergie, avec une autonomie permettant des sessions d'utilisation suffisantes pour la plupart des applications éducatives et de loisirs. Les batteries 18650 offrent un compromis intéressant entre capacité, rechargeabilité et disponibilité.

D. Autres Composants

* Webcam Externe, HC-SR04, Baffle: Ces composants consomment relativement peu d'énergie (moins de 3 W chacun) et sont alimentés directement par le Raspberry Pi ou via une source externe de 5 V. Leur impact sur la consommation énergétique totale du système est minimal mais doit être pris en compte dans le cadre global.

## METHODOLOGIES

Dans ce présent travail, les méthodologies utilisées sont : la recherche scientifique et l’expérimentation au laboratoire.

Concernant la recherche scientifique, nous avons décidé d’effectuer des recherches approfondies sur l’état de l’art des BCIs et la semi-autonomie dans différents articles scientifiques sur la toile en mettant en exergue les points positifs et négatifs de chacun d’entre eux, dans le but de ressortir un travail aguerri basé sur les échecs et réussites de nos prédécesseurs dans l’art. Le mémoire de Pierre Sedi est la pièce maîtresse de cette œuvre de concept car nous nous sommes inspirés des paradigmes utilisés sous les expérimentations qui nous ont guidées.

Concernant la méthode expérimentale, nous y sommes allés par essai-erreur surtout dans l’obtention du comportement voulu tel que décrit dans le cahier de charge. Quoique pas certain, nous nous sommes dépassés avec plus de volonté, se mettant dans la peau des chercheurs scientifiques.

Nous avons ainsi conçu et réalisé un système robot autonome avec BCI, qui permet de piloter à distance un véhicule, le PiCar-V, en utilisant les signaux EEG captés par un casque, le EMOTIV EPOC+. Notre système intègre également des fonctionnalités de navigation intelligente, de réactivité aux conditions changeantes, et de communication interactive, grâce à l’utilisation de technologies telles que la caméra, le capteur ultrason, le microphone, le haut-parleur, le modèle MobileNetV1, et le contrôleur neuronal.

Pour évaluer la performance et la fiabilité de notre système, nous avons mené plusieurs tests et expériences, en utilisant des scénarios variés. Nous avons ainsi pu mesurer l’impact de différents facteurs, tels que la distance, la connexion, l’environnement, et les obstacles, sur la qualité du pilotage et la satisfaction de l’utilisateur.

# CHAPITRE 4. EXPERIMENTATIONS ET RESULTATS

## CRITERIUM

Ce chapitre fait une narration de tout ce qui constitue les résultats sur le pilotage de notre voiture robot que l’on souhaite rendre autonome dans des situations spécifiées au sein du chapitre de la conception du système.

Dans notre cas, s’agissant d’un *proof of concept*, nous avons pu implémenter des codes qui nous ont permis de tirer quelques métriques que nous avons jugées d’essentielles après avoir détaillé la partie commande du chapitre 2.

Dans notre travail, nous avions tenu à ajouter un quatrième code qui implique la télé présence de l’utilisateur du BCI en termes de communication avec les autres humains. Déclassé car celui-ci est qualifié de *hard ressource* pour notre Raspberry Pi qui se doit de rester fluide pour le bon fonctionnement de nos applications. En annexe nous trouverons les quatre codes et le protocole expérimental choisi pour présenter nos résultats.

Lors de la présentation des résultats, nous allons nous pencher sur les métriques enregistrées à l’intérieur du script principal, qui est un serveur d’écoute, comme du fichier log prélevant les événements dans le temps. En dehors du temps, y a aussi la distance qui est important pour affirmer si ce qu’on a fixé dans le code se passe comme prévu.

## EXPERIMENTATION

Ils étaient au nombre de trois (3), les sujets qui nous ont permis de mener ces expériences sur le comportement objectivé et souhaité de notre robot. Chaque sujet a eu droit au protocole expérimental complet prédéfini, le temps étant imparti pour tout un chacun.

### ENTRAINEMENT DU SUJET

D’entrer de jeu, l’expérience commence par le mode opératoire d’apprentissage élaboré dans le travail de P. Sedi sur la classification des expressions faciales, paradigme inspiré et maintenu pour le pilotage à distance de notre système semi-autonome, qui se résume en ceci :

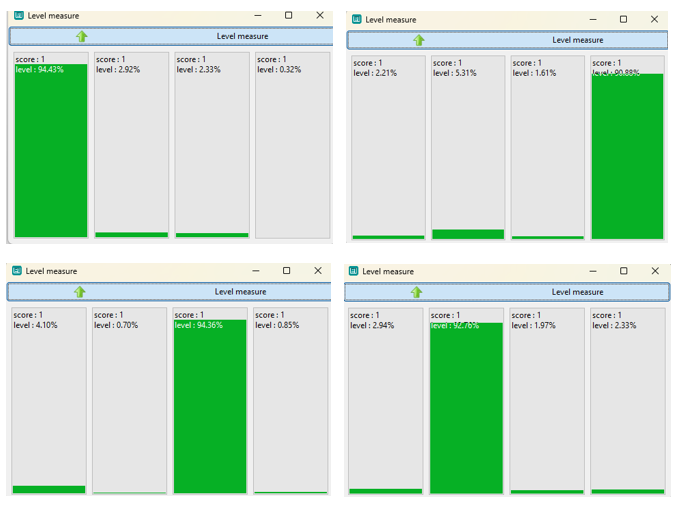
* Utilisation de CyKit et OpenViBE : Configuration d'OpenViBE pour l'acquisition, le filtrage, et le traitement des signaux EEG capturés par le casque EPOC+.
* Filtrage Spatial et Temporel : Application de filtres pour améliorer la qualité des signaux EEG et extraction des caractéristiques pertinentes.
* Entraînement du Classifieur : Utilisation d'un classifieur pour interpréter les signaux EEG et déterminer les intentions de mouvement.

Nous signalons que nous avons ajouté une quatrième classe afin de contrôler l’avancement du robot au lieu de le laisser être automatique dès le début du programme.

Les expressions faciales retenues sont :

Visage détendu (garder son calme sans rien faire) pas de direction à prendre, sourcils levés pour la direction gauche, mâchoires serrées pour la direction droite et large sourire en faisant montrer les dents pour le devant.

Cette étape est illustrée ci-dessous montrant ainsi l’apparition de la classe ajoutée par rapport au mode opératoire dont nous nous basons. Après une quinzaine de minutes chaque sujet était capable de voir comment il pouvait être en mesure de piloter le robot par suite d’un moment d’exercice lui demandant de refaire les expressions faciales suivant les directions apprises en mode online.



OpenVibe: Les 4 classes en mode online

Le script Python est appelé par OpenVibe pour permettre l’exécution, dans différents threads, de chaque classe lorsqu’elle est identifiée individuellement par l’EEG émanant du casque.

Une image contenant diagramme, texte, ligne

Description générée automatiquement

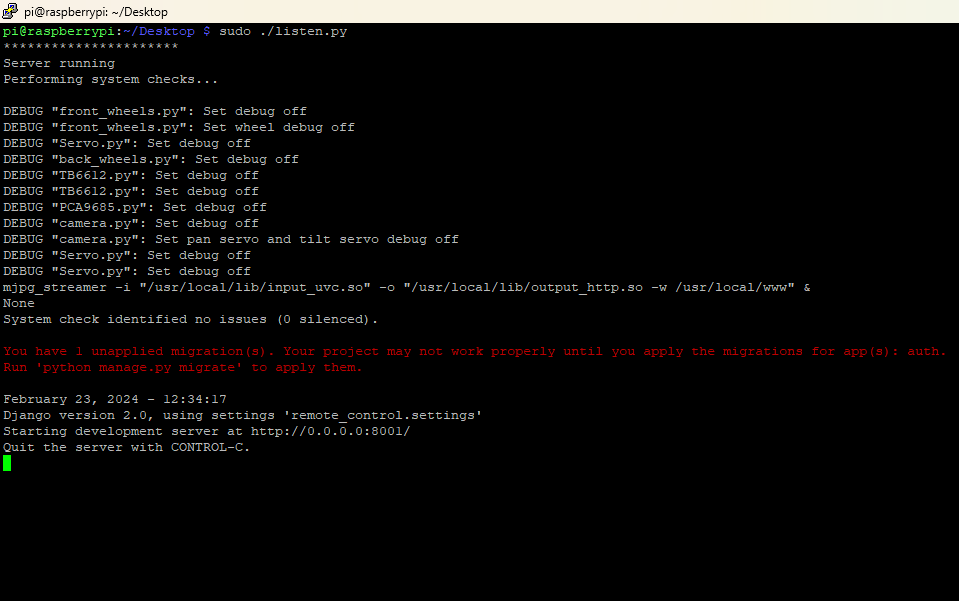
Design de l'arborescence des classes

### PILOTAGE DU ROBOT AVEC DES CAPACITES D’AUTONOMIE

Cette section est le cœur de notre travail qui a consisté à établir un système semi-autonome, en prenant ensemble un utilisateur du BCI et un robot en réel capacité de prendre des décisions sur des actions à opérer après que ce dernier ait reçu un ordre d’orientation.

Toujours avec nos sujets, en spécifiant que la dernière étape de l’apprentissage (en mode online comme précisé dans le mémoire de Sedi) sans se connecter au serveur qui se trouve dors et déjà placé sur le robot, était un aperçu du pilotage qui se fera directement sur celui-ci, nous allons connecter l’ICM à la partie restante du système, « Optimus Prime » notre robot.

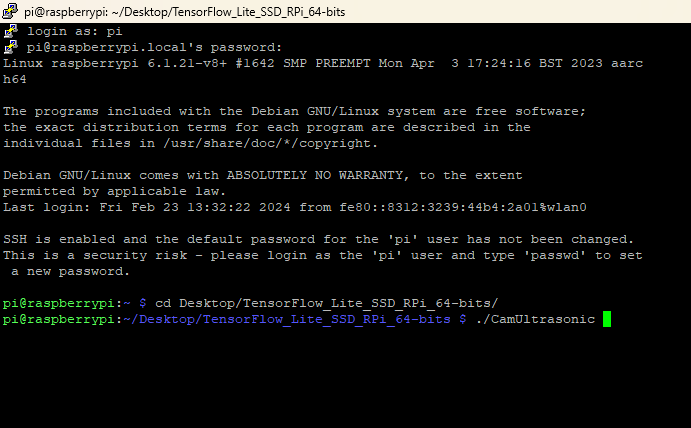
Nous allons lancer le serveur principal sur le port 8000 du Raspberry Pi qui va également, comme écrit dans le code python du serveur *listen.py*, lancer le serveur SunFounder Picar-V sur le port 8001 à l’aide de l’outil Putty pour Windows.



Lancement du serveur principal listen.py

Une fois lancé, le serveur est prêt à écouter les requêtes http en provenance de l’ICM via OpenVibe qui ouvrira son interface de colonne de niveau vu précédemment et qui permet une prise en main du pilotage à distance.

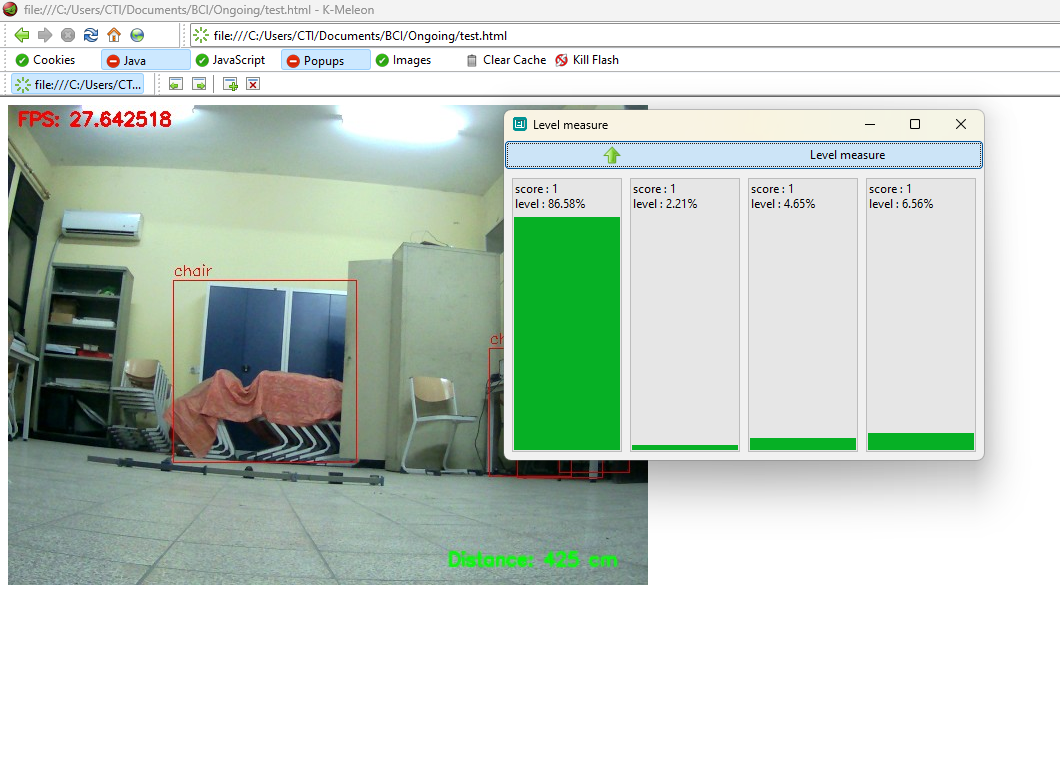
Le script de prise de décision du robot sera lancé ensuite pour enclencher les capteurs utilisés par Optimus Prime. Nous duplicons la session Putty en cours pour exécuter le fichier compilé *CamUltrasonic*.



CamUltrasonic prêt à être exécuté

Ainsi, en temps réel nous obtenons les mesures de distances et de tailles des objets sur le robot directement. Afin de faciliter le pilotage du robot par l’utilisateur, nous avons créé une page html de visualisation en streaming des mesures prises par les capteurs sur le robot.

Ces données sont présentées dans un navigateur léger, K-Meleon que nous avons pris soin de choisir compte tenu de son workspace dans un processeur.



Streaming du webcam sur K-meleon browser

Le sujet n’a plus qu’à respecter le protocole du scénario tracé et observer le comportement du robot piloter dans les situations d’objet trop proche, d’embourbement accidentellement créé, et de perte de connexion.

## RESULTATS

Sur bases des expériences réalisées sur nos 3 sujets et sur les métriques que nous avons énumérées en début du chapitre, nous présentons les résultats suivants sur un tableau :

|  |  |  |  |
| --- | --- | --- | --- |
| Métrique | Sujet 1 | Sujet 2 | Sujet 3 |
| Entrée/sortie du mode débogage "unstuck" (en s) | 11:26:22 à 11:26:32 (10s) | 12:35:20 à 12:35:30 (10s) | 13:27:56 à 13:28:06 (10 s) |
| Entrée/sortie du mode débogage "conn" (en s) | 11:28:48 à 11:28:55 (7s) |  | 13:30:19 à 13:30:25 (6s) |
| Temps de réaction à l'erreur de connexion (en s) | 6 | 5 | 6 |
| Distance d'arrêt pour obstacle (< 39 cm) | Oui | Oui | Oui |
| Avertissement objet à 70%/Arrêt selon taille objet à 85% | Oui | Oui | Oui |

Nous avons pris soin d’enregistrer les informations de chaque appel à une requête reçu par le serveur *raspberrypi.local :8000* et les différents comportements du robot dans le fichier *raspberry.log*, qui s’écrase automatiquement à chaque lancement du serveur.

Pour chaque sujet nous essayons d’interpréter les résultats que nous avons pu recueillir dans leur fichier .log renommé à chaque fois avant de pouvoir relancer le serveur d’écoute.

**Interprétation des résultats**

Nous allons interpréter les enregistrements des fichiers de nos sujets.

Le Sujet 1 a commencé son pilotage à 11h23 avec le démarrage du serveur, puis a directement reçu des requêtes de vérification de l’embourbement du robot car dans l’autre session Putty la caméra et l’ultrasonic étaient déjà lancés. L’utilisateur a commencé par regardé le flux vidéo sur la page web créée et ensuite la distance ne bougeait pas pour le robot. Le capteur de distance, l’ultrasonic a envoyé ces requêtes pour prévenir l’utilisateur que le robot pourrait être bloqué.

Puis comme dans les réseaux locaux, le ping continu d’un terminal vers un autre, ici du serveur RPI 4 vers le PC (comme écrit dans le script en annexe), a subi un temps de non réponse d’où le message reçu à la ligne 6 qui montre que l’état du Wi-Fi du RPI 4 est down.

Après qu’il soit de nouveau connecté au réseau et au PC, il y a eu un décalage dans l’arrivé des messages à cause de la connexion car le Sujet a demandé au robot d’aller de l’avant 5 fois de suite puis le robot s’est arrêté 4 fois puisque le code dit une fois forward, un second forward vient stopper l’avancement. La 5e fois étant la bonne, Optimus Prime est entré en mon déboggage suite à un embourbement superficiel créé dans la salle (cfr. Protocole en annexe).

Ensuite le robot a reçu l’ordre de prendre la direction droite et sortir de la salle jusqu’à faire face à un objet qui se trouve à moins de 39cm de lui puis stopper.

Il prendra alors la direction gauche où il ira jusqu’à perdre la connexion au réseau comme indiqué à 11h28.

Le Sujet 2 commence le pilotage à 12h33 et lance quelques secondes après le forward que le robot exécute jusqu’à s’embourber devant l’objet de taille inférieure à l’objectif de l’ultasonic créé pour simuler le stuck.

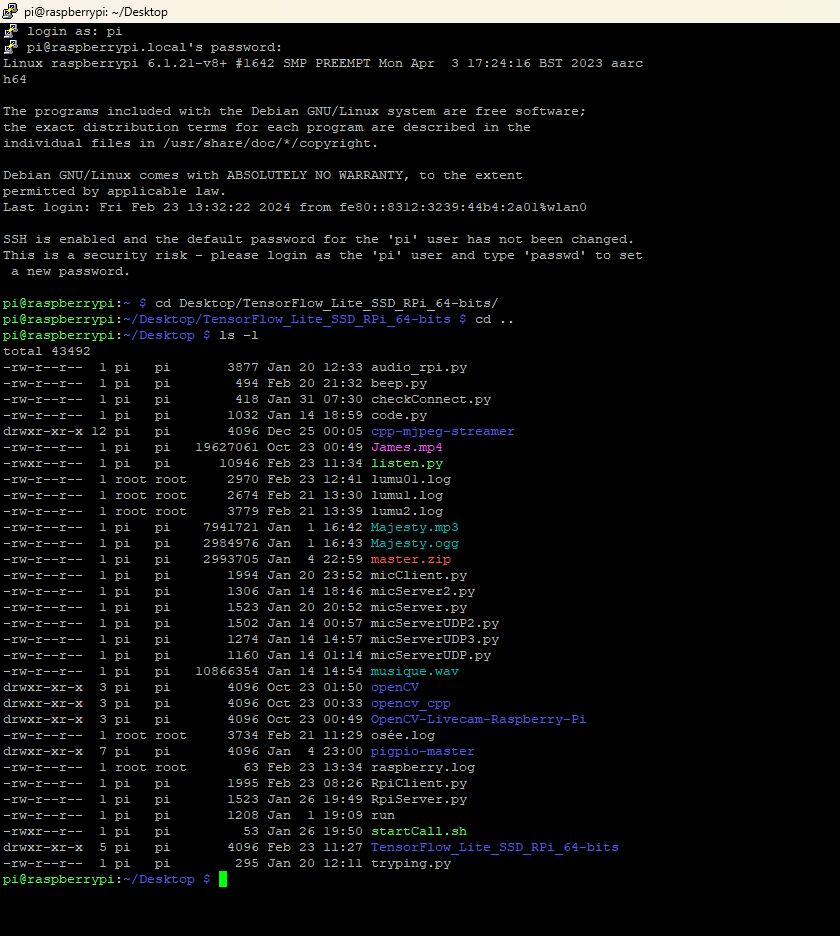
Sortie du stuck, le serveur reçoit la commande d’aller à droite en direction de la sortie de la salle. En moins de 39cm de l’objet il stoppe pour prendre la gauche comme prochaine commande. Il avancera à une distance où il constate que le PC n’est plus connecté. Sùurement un ploblème de connexion car le PC fixe dans le réseau et proche du modem.

Le scénario s’est arrêté car les batteries commençaient à lâcher et les leds sur le picar-V clignotaient.

Le Sujet 3, quant à lui, commence bien son pilotage tout comme pour le Sujet 2, un forward puis l’entrée en mode unstuck. Il perd la connexion au PC à cause de l’état du Wi-Fi du RPI 4 qui est sur down, puis se reconnecte.  
Le moment passé sans agir a envoyé un checkStuck qui ne sera pas considéré, ensuite le Sujet fait un forward right pour sortir le robot de la salle où il s’arrêtera à moins de 39cm de l’objet.

Puis il fait un forward left pour longer le couloir jusqu’à perdre la connexion lorsqu’il sort du réseau.

Le scénario avait prévu au départ un événement de conversation avec un étudiant au couloir, d’où le second foward envoyé à chaque fois par les utilisateurs pour stopper le robot et lancer le dialogue.   
Après avoir constaté que la conversation qui se faisait entre le RPI 4 et le PC resté dans la salle, via un autre port du RPI 4, consommait beaucoup de ressources, on s’est résolu de l’enlever du protocole.





Les fichiers .log après leur sauvegarde

## DIFFICULTES RENCONTREES

Nous ne pouvons pas finir ce chapitre sans parler des difficultés rencontrés…

## FEEDBACK DES UTILISATEURS

Nous nous sommes fixés de fournir un travail qui respecte les bases d’une bonne utilisation du coté sujets, offrant une ergonomie totale et …

# CONCLUSION ET PERSPECTIVES

# BIBLIOGRAPHIE

# ANNEXES

## SCRIPTS

Le serveur principal listen.py :

|  |
| --- |
| #!/usr/bin/python  **import** **socketserver**  **from** **http.server** **import** BaseHTTPRequestHandler,HTTPServer  **import** **time**  **import** **http.client**  **import** **subprocess**  **from** **urllib.request** **import** urlopen  **import** **threading**  **from** **multiping** **import** multi\_ping  **import** **logging** # Importer le module logging  **from** **logging.handlers** **import** RotatingFileHandler # Importer la classe RotatingFileHandler  **import** **os**  **import** **pyaudio**  **from** **math** **import** pi  **import** **numpy** **as** **np**  actionsList = []  runningFW = **False**  runningLeft = **False**  runningRight = **False**  stuckRequests = **0**  silenceServer = **False**  setRunSpeed = **False**  disconnected = **False**  startDebugTime = **0**  backTime = **7**  isStop = **False**  PC = '192.168.0.100'  **if** os.path.exists("raspberry.log"): os.remove("raspberry.log")  logger = logging.getLogger(\_\_name\_\_)  logger.setLevel(logging.INFO) # Définir le niveau de sévérité à INFO  handler = RotatingFileHandler("raspberry.log", mode="w", maxBytes=**10**\***1024**\***1024**, backupCount=**0**) # Créer un objet handler pour écrire dans un fichier  formatter = logging.Formatter("%(asctime)s - %(levelname)s - %(message)s") # Définir le format du message  handler.setFormatter(formatter) # Associer le format au handler  logger.addHandler(handler) # Associer le handler au logger  **try**:  subprocess.Popen(["sudo /home/pi/SunFounder\_PiCar-V/remote\_control/start"], shell=**True**)  print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')  logger.info("Démarrage du serveur PiCar-V") # Utiliser logger.info pour enregistrer un message  **except**:  print("Error: Unable to run PiCar-V server.")  logger.exception("Erreur lors du démarrage du serveur PiCar-V") # Utiliser logger.exception pour enregistrer une erreur  **def** **make\_sinewave**(frequency, length, sample\_rate=**44100**):  length = int(length \* sample\_rate)  factor = float(frequency) \* (pi \* **2**) / sample\_rate  waveform = np.sin(np.arange(length) \* factor)  **return** waveform  **def** **piCar**(actionPiCar):  **global** setRunSpeed  **if** **not** setRunSpeed:  urlopen("http://0.0.0.0:8001/run/?speed=54")  setRunSpeed = **True**  urlopen("http://0.0.0.0:8001/run/?action="+actionPiCar)  wave = make\_sinewave(**1000**, **1**) # Créer un signal de 1000 Hz et 1 s  **def** **check\_wifi\_connection**():  **global** disconnected  **global** startDebugTime  **global** backTime  # Ajouter une variable pour compter le nombre de fois que le PC est injoignable  **global** unreachable\_count  **try**:  responses, no\_responses = multi\_ping([PC], timeout=**2**, retry=**0**)  # Augmenter le compteur de 1 si le PC ne répond pas  **if** len(responses)==**0**:  unreachable\_count += **1**  # Réinitialiser le compteur à zéro si le PC répond  **else**:  unreachable\_count = **0**  # Vérifier si le compteur est supérieur ou égal à 3  **if** unreachable\_count >= **3**:  print("Error: Pc is unreachable.")  logger.exception("Erreur de connexion au PC")  piCar("stop")  result = subprocess.check\_output("cat /sys/class/net/wlan0/operstate", shell=**True**)  operstate = result.decode().strip()  **if** operstate == "down":  logger.exception('down down down, Je viens de perdre la connexion au réseau')  **if** startDebugTime==**0**: startDebugTime = time.time()  debugMode('conn')  **else**:  print("Error: I'm in Wi-Fi but PC is out of Wi-Fi.")  logger.exception("Erreur: I'm in Wi-Fi but PC is out of Wi-Fi.")  # Afficher le code d'erreur  print("Error: PC is unreachable for 3 times or more.")  # Faire beeper le robot  p = pyaudio.PyAudio()  stream = p.open(format=pyaudio.paFloat32, channels=**1**, rate=**44100**, output=**1**,)  beeping = **True**  logger.info("Beeping est activé")  **if** beeping:  stream.write(wave.astype(np.float32).tostring())  time.sleep(**1**)  beeping = **False**  stream.stop\_stream()  stream.close()  p.terminate()  disconnected = **True**  # Sinon, si le PC répond, continuer normalement  **elif** len(responses) > **0**:  beeping = **False**  startDebugTime = **0**  backTime = **7**  **if** disconnected:  logger.info("De nouveau connecté")  disconnected = **False**  **except**:  print("Error: Unable to check Wi-Fi connection")  logger.exception("Erreur lors de la vérification de la connexion Wi-Fi")  timer = threading.Timer(**3**, check\_wifi\_connection) # Créer un objet timer qui appelle la fonction check\_wifi\_connection après 3 secondes  timer.start()  **def** **debugMode**(a):  **global** actionsList  **global** silenceServer  **global** backTime  **global** startDebugTime  silenceServer = **True**  print("Entering debug mode...")  logger.info("Entrée en mode debug") # Utiliser logger.info pour enregistrer un message  piCar('stop')  piCar('fwstraight')  #for i in range(0, 3):  **if** actionsList[-**1**][**0**] == 'forward':  piCar('backward')  #elif actionsList[-1][0] == 'backward':  # piCar('forward')  **elif** actionsList[-**1**][**0**] == 'fwleft':  piCar('fwleft')  piCar('backward')  time.sleep(**4**)  piCar('fwstraight')  backTime = **7**  startDebugTime -= **7**  **elif** actionsList[-**1**][**0**] == 'fwright':  piCar('fwright')  piCar('backward')  time.sleep(**4**)  piCar('fwstraight')  backTime = **7**  startDebugTime -= **7**  **if** a=='conn' **and** actionsList[-**1**][**0**] == 'forward':  logger.info("Fin du mode debug connexion")  time.sleep(backTime) #( int(actionsList[-1][1] - actionsList[-2][1]) )  startDebugTime -= backTime # for the next back, if it ever have to happen  **if** backTime < **7** : actionsList.pop() # si le robot est revenu en arriere pendant moins de 7 secondes, alors on enleve le dernier objet de la liste  **if** ( startDebugTime - actionsList[-**1**][**1**] ) < **7** :  backTime = startDebugTime - actionsList[-**1**][**1**]  **else**: backTime = **7**  **elif** a=='unstuck':  logger.info("Fin du mode debug unstuck")  time.sleep(**10**)  piCar('stop')  piCar('fwstraight')  logger.info("Sortie du mode debug")  silenceServer = **False**  **class** **MyHandler**(BaseHTTPRequestHandler):  **def** **do\_GET**(self):  self.send\_response(**200**)  self.send\_header('Content-type', 'text/plain')  self.end\_headers()  **global** silenceServer, runningFW, runningLeft, runningRight, stuckRequests, isStop  **if** silenceServer: **return**  **if** self.path == '/run/?action=fwleft':  **if** **not** stuckRequests: # Ajouter une condition pour vérifier si stuckRequests est False  actionsList.append(['fwleft',time.time()])  runningLeft = silenceServer = **True**  piCar('fwleft')  piCar('forward')  time.sleep(**7**)  piCar('fwstraight')  actionsList.append(['forward',time.time()])  runningLeft = silenceServer = **False**  runningFW = **True**  isStop = **False** # Mettre isStop à False  logger.info("Requête GET /run/?action=fwleft reçue et exécutée") # Utiliser logger.info pour enregistrer un message  **elif** self.path == '/run/?action=forward':  **if** **not** stuckRequests: # Ajouter une condition pour vérifier si stuckRequests est False  **if** runningFW:  piCar('stop')  runningFW = **False**  **else**:  actionsList.append(['forward',time.time()])  runningFW = silenceServer = **True**  piCar('forward')  #time.sleep(3)  silenceServer = **False**  isStop = **False** # Mettre isStop à False  logger.info("Requête GET /run/?action=forward reçue et exécutée") # Utiliser logger.info pour enregistrer un message  **elif** self.path == '/run/?action=fwright':  **if** **not** stuckRequests: # Ajouter une condition pour vérifier si stuckRequests est False  actionsList.append(['fwright',time.time()])  runningRight = silenceServer = **True**  piCar('fwright')  piCar('forward')  time.sleep(**7**)  piCar('fwstraight')  actionsList.append(['forward',time.time()])  runningRight = silenceServer = **False**  runningFW = **True**  isStop = **False** # Mettre isStop à False  logger.info("Requête GET /run/?action=fwright reçue et exécutée") # Utiliser logger.info pour enregistrer un message  **elif** self.path == '/run/?action=fwstraight':  **if** **not** stuckRequests: # Ajouter une condition pour vérifier si stuckRequests est False  piCar('fwstraight')  isStop = **False** # Mettre isStop à False  logger.info("Requête GET /run/?action=fwstraight reçue et exécutée") # Utiliser logger.info pour enregistrer un message  **elif** self.path == '/run/?action=backward': # Ne sera jamais appelE  #actionsList.append(['backward',time.time()])  silenceServer = **True**  piCar('backward')  time.sleep(**7**)  silenceServer = **False**  isStop = **False** # Mettre isStop à False  logger.info("Requête GET /run/?action=backward reçue et exécutée") # Utiliser logger.info pour enregistrer un message  **elif** self.path == '/run/?action=stop':  piCar('stop')  print('receive stop')  #time.sleep(7)  isStop = **True** # Mettre isStop à True  stuckRequests = **False** # Mettre stuckRequests à False  logger.info("Requête GET /run/?action=stop reçue et exécutée") # Utiliser logger.info pour enregistrer un message  **elif** self.path == '/run/?action=checkStuck':  **if** **not** isStop: # Ajouter une condition pour vérifier si isStop est False  **if** (runningFW **or** runningLeft **or** runningRight):  # Mettre stuckRequests à True  stuckRequests = **True**  print('The robot is stucked.')  debugMode('unstuck')  #time.sleep(20)  runningFW = runningLeft = runningRight = **False**  **else**:  # Mettre stuckRequests à False  stuckRequests = **False**  print('The robot is not stucked.')  logger.info("Requête GET /run/?action=checkStuck reçue et traitée")  httpd = socketserver.TCPServer(("", **8000**), MyHandler)  check\_wifi\_connection()  **try**:  httpd.serve\_forever()  **except** **KeyboardInterrupt**:  print("Shutting down the server...")  httpd.shutdown() |

Les configurations utilisées dans le travail :

|  |
| --- |
| g++ -o ultrasonic ultrasonic.cpp -lwiringPi  g++ -std=c++**17** -fexceptions -pthread -g -Wall -o avoidance avoidance.cpp -lwiringPi `pkg-config opencv4 --cflags --libs` `pkg-config gstreamer-**1.0** --cflags --libs` -lopenal -lsndfile -lpthread -I/home/pi/tensorflow -I/home/pi/tensorflow/tensorflow/lite/tools/make/downloads/flatbuffers/include -I/home/pi/tensorflow/tensorflow/lite/tools/make/downloads/absl -I/home/pi/Desktop/cpp-mjpeg-streamer/include -L/home/pi/tensorflow/tensorflow/lite/tools/make/gen/linux\_aarch64/lib -L/home/pi/tensorflow/tensorflow/lite/tools/make/downloads/flatbuffers/build -ltensorflow-lite -lcurl -ldl  gcc -o ultra ultra.cpp -lpigpio -lpthread -lrt -ldl -lstdc++  g++ server2.cpp -lasound -lpthread -o server2  g++ -std=c++**17** -o server server.cpp -lpthread -lportaudio  g++ -std=c++**17** -fexceptions -pthread -g -Wall -o CamUltrasonic CamUltrasonic.cpp -lwiringPi `pkg-config opencv4 --cflags --libs` `pkg-config gstreamer-**1.0** --cflags --libs` -lopenal -lsndfile -lpthread -I/home/pi/tensorflow -I/home/pi/tensorflow/tensorflow/lite/tools/make/downloads/flatbuffers/include -I/home/pi/tensorflow/tensorflow/lite/tools/make/downloads/absl -I/home/pi/Desktop/cpp-mjpeg-streamer/include -L/home/pi/tensorflow/tensorflow/lite/tools/make/gen/linux\_aarch64/lib -L/home/pi/tensorflow/tensorflow/lite/tools/make/downloads/flatbuffers/build -L/path/to/openal -L/path/to/sndfile -ltensorflow-lite -lcurl -ldl  g++ -std=c++**17** -fexceptions -pthread -g -Wall -o CamUltrasonic CamUltrasonic.cpp -lwiringPi `pkg-config opencv4 --cflags --libs` `pkg-config gstreamer-**1.0** --cflags --libs` -lopenal -lsndfile -lpthread -I/home/pi/tensorflow -I/home/pi/tensorflow/tensorflow/lite/tools/make/downloads/flatbuffers/include -I/home/pi/tensorflow/tensorflow/lite/tools/make/downloads/absl -I/home/pi/Desktop/cpp-mjpeg-streamer/include -L/home/pi/tensorflow/tensorflow/lite/tools/make/gen/linux\_aarch64/lib -L/home/pi/tensorflow/tensorflow/lite/tools/make/downloads/flatbuffers/build -L/path/to/openal -L/path/to/sndfile -ltensorflow-lite -lcurl -ldl  g++ -std=c++**17** -fexceptions -pthread -g -Wall -o CamUltrasonic CamUltrasonic.cpp -lwiringPi `pkg-config opencv4 --cflags --libs` `pkg-config gstreamer-**1.0** --cflags --libs` -lopenal -lsndfile -lpthread -I/home/pi/tensorflow -I/home/pi/tensorflow/tensorflow/lite/tools/make/downloads/flatbuffers/include -I/home/pi/tensorflow/tensorflow/lite/tools/make/downloads/absl -I/home/pi/Desktop/cpp-mjpeg-streamer/include -L/home/pi/tensorflow/tensorflow/lite/tools/make/gen/linux\_aarch64/lib -L/home/pi/tensorflow/tensorflow/lite/tools/make/downloads/flatbuffers/build -ltensorflow-lite -lcurl -ldl  g++ -std=c++**17** -fexceptions -pthread -g -Wall -o CamUltrasonic CamUltrasonic.cpp -lwiringPi `pkg-config opencv4 --cflags --libs` `pkg-config gstreamer-**1.0** --cflags --libs` -lopenal -lsndfile -lpthread -I. -I/home/pi/tensorflow -I/home/pi/tensorflow/tensorflow/lite/tools/make/downloads/flatbuffers/include -I/home/pi/tensorflow/tensorflow/lite/tools/make/downloads/absl -I/home/pi/Desktop/cpp-mjpeg-streamer/include -L/home/pi/tensorflow/tensorflow/lite/tools/make/gen/linux\_aarch64/lib -L/home/pi/tensorflow/tensorflow/lite/tools/make/downloads/flatbuffers/build -ltensorflow-lite -ldl  sudo fuser -n tcp -k **8001**  sudo shutdown -h now ou sudo halt  python3 .\CyKIT.py **127.0.0.1** **5151** **6** openvibe+generic+nocounter+noheader+nobattery+ovdelay:**100**+**float**+ovsamples:**004** |

Les restes des scripts utilisés dans ce mémoire se retrouvent dans notre repository github. Vous pouvez utiliser ce lien : <https://github.com/newisdartagnan/TFE>

## PROTOCOLE EXPERIMENTAL

## LOGS DES SUJETS

|  |
| --- |
| Sujet **1** :  **2024**-02-21 **11**:23:10,576 - INFO - Démarrage du serveur PiCar-V  **2024**-02-21 **11**:24:30,741 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-21 **11**:24:36,776 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-21 **11**:24:42,796 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-21 **11**:24:48,886 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-21 **11**:24:49,487 - ERROR - Erreur de connexion au PC  NoneType: None  **2024**-02-21 **11**:24:49,548 - ERROR - Erreur WiFi is down  NoneType: None  **2024**-02-21 **11**:24:49,697 - INFO - Beeping est activé  **2024**-02-21 **11**:24:54,899 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-21 **11**:24:54,934 - INFO - De nouveau connecté  **2024**-02-21 **11**:25:00,923 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-21 **11**:25:06,964 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-21 **11**:25:07,248 - INFO - Requête GET /run/?action=forward reçue et exécutée  **2024**-02-21 **11**:25:13,065 - INFO - Requête GET /run/?action=forward reçue et exécutée  **2024**-02-21 **11**:25:19,954 - INFO - Requête GET /run/?action=forward reçue et exécutée  **2024**-02-21 **11**:25:43,101 - INFO - Requête GET /run/?action=stop reçue et exécutée  **2024**-02-21 **11**:25:51,381 - INFO - Requête GET /run/?action=stop reçue et exécutée  **2024**-02-21 **11**:25:59,721 - INFO - Requête GET /run/?action=stop reçue et exécutée  **2024**-02-21 **11**:26:08,039 - INFO - Requête GET /run/?action=stop reçue et exécutée  **2024**-02-21 **11**:26:09,029 - INFO - Requête GET /run/?action=forward reçue et exécutée  **2024**-02-21 **11**:26:11,923 - INFO - Requête GET /run/?action=forward reçue et exécutée  **2024**-02-21 **11**:26:16,021 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-21 **11**:26:22,184 - INFO - Entrée en mode debug  **2024**-02-21 **11**:26:22,398 - INFO - Fin du mode debug unstuck  **2024**-02-21 **11**:26:32,488 - INFO - Sortie du mode debug  **2024**-02-21 **11**:26:32,488 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-21 **11**:26:32,540 - INFO - Requête GET /run/?action=stop reçue et exécutée  **2024**-02-21 **11**:26:39,476 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-21 **11**:26:45,571 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-21 **11**:26:51,625 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-21 **11**:27:03,556 - INFO - Requête GET /run/?action=fwright reçue et exécutée  **2024**-02-21 **11**:27:03,560 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-21 **11**:27:10,756 - INFO - Requête GET /run/?action=fwright reçue et exécutée  **2024**-02-21 **11**:27:27,096 - INFO - Requête GET /run/?action=stop reçue et exécutée  **2024**-02-21 **11**:27:34,312 - INFO - Requête GET /run/?action=fwleft reçue et exécutée  **2024**-02-21 **11**:27:51,494 - INFO - Requête GET /run/?action=forward reçue et exécutée  **2024**-02-21 **11**:27:58,591 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-21 **11**:27:58,882 - INFO - Requête GET /run/?action=forward reçue et exécutée  **2024**-02-21 **11**:28:48,233 - INFO - Entrée en mode debug  **2024**-02-21 **11**:28:48,233 - INFO - Fin du mode debug conn  **2024**-02-21 **11**:28:55,529 - INFO - Sortie du mode debug  **2024**-02-21 **11**:29:08,571 - INFO - Requête GET /run/?action=checkStuck reçue et traitée |

|  |
| --- |
| Sujet **2** :  **2024**-02-23 **12**:33:54,324 - INFO - Démarrage du serveur PiCar-V  **2024**-02-23 **12**:34:16,131 - INFO - Requête GET /run/?action=forward reçue et exécutée  **2024**-02-23 **12**:34:34,629 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-23 **12**:35:20,211 - INFO - Entrée en mode debug  **2024**-02-23 **12**:35:20,315 - INFO - Fin du mode debug unstuck  **2024**-02-23 **12**:35:30,374 - INFO - Sortie du mode debug  **2024**-02-23 **12**:35:30,375 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-23 **12**:36:02,476 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-23 **12**:36:36,686 - INFO - Requête GET /run/?action=fwright reçue et exécutée  **2024**-02-23 **12**:37:43,403 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-23 **12**:37:52,734 - INFO - Requête GET /run/?action=stop reçue et exécutée  **2024**-02-23 **12**:38:07,646 - INFO - Requête GET /run/?action=fwleft reçue et exécutée  **2024**-02-23 **12**:38:07,695 - INFO - Requête GET /run/?action=stop reçue et exécutée  **2024**-02-23 **12**:38:15,380 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-23 **12**:38:21,444 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-23 **12**:38:27,457 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-23 **12**:38:34,108 - INFO - Requête GET /run/?action=forward reçue et exécutée  **2024**-02-23 **12**:39:45,999 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-23 **12**:39:58,127 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-23 **12**:40:04,163 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-23 **12**:40:10,219 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-23 **12**:40:16,256 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-23 **12**:40:22,269 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-23 **12**:40:28,352 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-23 **12**:40:56,966 - ERROR - Erreur de connexion au PC  NoneType: None  **2024**-02-23 **12**:40:57,020 - ERROR - Erreur: I'm in Wi-Fi but PC is out of Wi-Fi.  NoneType: None  **2024**-02-23 **12**:40:57,178 - INFO - Beeping est activé  **2024**-02-23 **12**:40:58,450 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-23 **12**:41:02,378 - INFO - De nouveau connecté  **2024**-02-23 **12**:41:04,532 - INFO - Requête GET /run/?action=checkStuck reçue et traitée |

|  |
| --- |
| Sujet **3** :  **2024**-02-21 **13**:26:10,850 - INFO - Démarrage du serveur PiCar-V  **2024**-02-21 **13**:27:03,580 - INFO - Requête GET /run/?action=forward reçue et exécutée  **2024**-02-21 **13**:27:38,282 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-21 **13**:27:56,434 - INFO - Entrée en mode debug  **2024**-02-21 **13**:27:56,666 - INFO - Fin du mode debug unstuck  **2024**-02-21 **13**:28:06,798 - INFO - Sortie du mode debug  **2024**-02-21 **13**:28:06,799 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-21 **13**:28:08,520 - ERROR - Erreur de connexion au PC  NoneType: None  **2024**-02-21 **13**:28:08,552 - ERROR - Erreur WiFi is down  NoneType: None  **2024**-02-21 **13**:28:08,702 - INFO - Beeping est activé  **2024**-02-21 **13**:28:14,177 - INFO - De nouveau connecté  **2024**-02-21 **13**:28:14,793 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-21 **13**:28:31,819 - INFO - Requête GET /run/?action=fwright reçue et exécutée  **2024**-02-21 **13**:28:39,038 - INFO - Requête GET /run/?action=stop reçue et exécutée  **2024**-02-21 **13**:28:54,574 - INFO - Requête GET /run/?action=fwleft reçue et exécutée  **2024**-02-21 **13**:28:54,634 - INFO - Requête GET /run/?action=stop reçue et exécutée  **2024**-02-21 **13**:29:02,680 - INFO - Requête GET /run/?action=forward reçue et exécutée  **2024**-02-21 **13**:29:10,989 - INFO - Requête GET /run/?action=stop reçue et exécutée  **2024**-02-21 **13**:29:30,043 - ERROR - Erreur de connexion au PC  NoneType: None  **2024**-02-21 **13**:29:30,076 - ERROR - Erreur WiFi is down  NoneType: None  **2024**-02-21 **13**:29:30,202 - INFO - Beeping est activé  **2024**-02-21 **13**:29:31,146 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-21 **13**:29:35,449 - INFO - De nouveau connecté  **2024**-02-21 **13**:29:37,182 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-21 **13**:29:49,275 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-21 **13**:29:55,396 - INFO - Requête GET /run/?action=forward reçue et exécutée  **2024**-02-21 **13**:30:11,483 - INFO - Requête GET /run/?action=stop reçue et exécutée  **2024**-02-21 **13**:30:19,570 - INFO - Entrée en mode debug  **2024**-02-21 **13**:30:19,724 - INFO - Fin du mode debug conn  **2024**-02-21 **13**:30:25,784 - INFO - Sortie du mode debug  **2024**-02-21 **13**:30:49,830 - INFO - Requête GET /run/?action=checkStuck reçue et traitée  **2024**-02-21 **13**:30:55,859 - INFO - Requête GET /run/?action=checkStuck reçue et traitée |

## PHOTO D’OPTIMUS PRIME

1. [L’enjeu d’une Interface Cerveau Ordinateur fonctionnant pour chacun - Hello Future Orange](https://hellofuture.orange.com/fr/lenjeu-dune-interface-cerveau-ordinateur-fonctionnant-pour-tout-le-monde/) [↑](#footnote-ref-1)
2. [Apprendre à contrôler une interface cerveau-ordinateur : le projet BrainConquest | Cairn.info](https://www.cairn.info/revue-realites-industrielles-2021-3-page-16.htm) [↑](#footnote-ref-2)
3. P.Sedi, « Conception d’une interface cerveau-machine temps-réel asynchrone pour le pilotage à distance des systèmes robotiques à l’aide du casque Emotiv EPOC+ : Cas du robot Smart Video Car » [↑](#footnote-ref-3)
4. [Smart Robot Car Kit for Raspberry Pi - SunFounder](https://www.sunfounder.com/products/smart-video-car) [↑](#footnote-ref-4)
5. https://institutducerveau-icm.org/fr/actualite/comprendre-le-cerveau-et-son-fonctionnement/ [↑](#footnote-ref-5)
6. https://institutducerveau-icm.org/fr/actualite/comprendre-le-cerveau-et-son-fonctionnement/ [↑](#footnote-ref-6)