

DÉPARTEMENT D'INFORMATIQUE

LICENCE FONDAMENTALE : SCIENCES MATHÉMATIQUES ET INFORMATIQUE (SMI)

ANNÉE UNIVERSITAIRE : 2021-2022

Projet de Fin d'Études

Intitulé :

Embarquement d'une Caméra OpenMV sur Un Robot Mobile

Réalisé par : Bargache Kaoutar et Mejdoub Anas

Sous l'Encadrement de :

Abdelbaki El BELRHITI EL ALAOUI
Hicham ELAKHAL

Soutenue le 06/07/2022 devant le jury composé de :

Pr. A. EL BELRHITI EL ALAOUI, Professeur à la FS Meknès

Pr. H. BOURRAY, Professeur à la FS Meknès

Pr. E. ISMAILI ALAOUI, Professeur à la FS Meknès

Remerciements

Après avoir terminé ce projet de fin d'étude, nous réservons ces lignes pour exprimer nos remerciements les plus sincères à Dieu le tout puissant de nous avoir donné la capacité d'écrire et de réfléchir, la force d'y croire et la patience pour terminer ce travail Nous remercions tout d'abord Allah.

Nous tenons avec un grand plaisir et un grand respect à remercier notre encadrant **Mr. EL BELRHITI ELALAOUI ABDELBAKI**, pour nous avoir encadrées et fait confiance. Merci pour vos conseils précieux, votre gentillesse et votre patience durant la réalisation de ce travail. De même que nous remercions les membres du jury pour sa participation. nous leur exprimons notre vifs et sincères remerciement pour l'honneur d'avoir accepté de juger et d'évaluer ce travail. Nous adressons aussi nos reconnaissances à tous le staff administratif de la faculté pour leurs conseils et leurs connaissances qu'ils nous ont bien servis, durant notre formation.

Nous remercions du fond du cœur nos parents pour leur soutien, leurs encouragements et pour la réalisation de ces études dans les bonnes conditions

Nous tenons enfin à remercier tous les amis de notre promo et tous ceux qui ont collaborés de près ou de loin à l'élaboration de ce travail

Résumé

Le contexte de ce projet de fin d'études est d'embarquer la caméra OpenMV sur le robot mobile Zumo, par la réalisation d'une application qui consiste à contrôler le mouvement du robot et de capturer des images, au même temps la diffusion d'une vidéo en direct par la caméra. L'étude de ce projet a été répartie sur plusieurs étapes essentielles :

- Une étape de description des matérielles (Raspber pi, Caméra OpenMV et Robot...) et des logiciels (le système d'exploitation et les éditeurs).
- Une étape d'embarquement du caméra OpenMV avec la carte raspberry pi en utilisant les protocoles de communication serie UART et RPC.
- Une dernière étape la réalisation de l'application d'embarquement de la caméra Open MV avec le robot qui consiste à la télécommande du robot à partir d'un autre ordinateur a l'aide de son lien avec la RPI, en utilisant une communication série entre les deux cartes arduino et raspberry pi, puisque le robot zumo ni compatible qu'avec l'arduino.

Cette dernière étude nous a permis de mieux connaître les composants électriques et en particulier la caméra MV au niveau de la connexion avec d'autres parties électriques, le traitement de l'image et l'utilisation du RPI. Afin de savoir de quelle manière le lien entre les systèmes intégrés est établi.

LISTE DES FIGURES

| | | |
|----|---|----|
| 1 | Raspberry Pi | 11 |
| 2 | Caption | 12 |
| 3 | Pins du RPI | 13 |
| 4 | Liste des Pins GPIO | 13 |
| 5 | Caméra OpenMV | 14 |
| 6 | Caméra OpenMV M7 | 15 |
| 7 | Zummo Robot | 16 |
| 8 | Arduino Uno | 16 |
| 1 | Raspbian | 17 |
| 2 | Interface d'OpemMV IDE | 18 |
| 3 | Arduino IDE | 19 |
| 4 | VNC | 19 |
| 5 | Pyserial | 20 |
| 6 | Micropython | 20 |
| 7 | protocole UART | 22 |
| 8 | protocole RPC | 22 |
| 1 | Montage | 24 |
| 2 | Terminal de RPI | 25 |
| 3 | Serial terminal du OpenMV IDE | 25 |
| 4 | Stockage des images dans RPI | 26 |
| 5 | Image par Caméra OpenMV | 27 |
| 6 | Video en direct par OpenMV capturant la RPI | 28 |
| 7 | Schéma d'application | 29 |
| 8 | Montage | 29 |
| 9 | Interface | 30 |
| 10 | Raspberry PI Blob (partie 1) | 31 |
| 11 | Raspnerry PI Blob (partie 2) | 31 |
| 12 | Raspnerry PI Blob (partie 3) | 32 |
| 13 | Caméra OpenMV | 32 |
| 14 | Raspberry PI (Snapshot) | 33 |
| 15 | Caméra OpenMV (Snapshot) | 33 |
| 16 | Raspberry PI (Streaming) | 34 |

| | | |
|----|---------------------------|----|
| 17 | Raspberry PI (Streaming) | 34 |
| 18 | Caméra OpenMV (Streaming) | 35 |

ABBRÉVIATIONS

RPI : Raspberry Pi

IDE : Integrated Development Environment

GPIO : General Purpose Input/Output

UART : Universal Asynchronous Receiver Transmitter

RPC : Remote Procedure Call

MISO : Master Input, Slave Output

MOSI : Master Output, Slave Input

VNC : Virtual Network Computing

USB : Universal Serial Bus

E/S : Entrées et Sorties

QVGA : Quarter Video Graphics Array

RFB : Remonte Frame Buffer

SOMMAIRE

| | |
|--|-----------|
| Résumé | 3 |
| Introduction générale | 9 |
| 1 Cahier de Charge | 10 |
| 2 HARDWARE | 11 |
| I Carte Raspberry Pi | 11 |
| I.1 Présentaion générale | 11 |
| I.2 Description de carte RPI. | 12 |
| I.3 Pins du Raspberry Pi | 13 |
| II Caméra OpenMV | 14 |
| II.1 Présentaion générale | 14 |
| II.2 Caractéristiques du caméra OpenMV | 14 |
| II.3 Modèles caméra OpenMV M7 | 15 |
| III Robot Zumo | 16 |
| IV Arduino | 16 |
| 3 SOFTWARE | 17 |
| I Système d'Exploitation Raspbian | 17 |
| II OpenMV IDE | 18 |
| II.1 Bibliothèque d'interface | 18 |
| II.2 Bibliothèque RPC | 18 |
| III Arduino IDE | 19 |
| IV VNC Viewer | 19 |
| V Bibliothèques | 20 |
| V.1 Pyserial | 20 |
| V.2 Micropython | 20 |
| V.3 Zumo Shield | 21 |
| VI Protocoles de communication : | 21 |
| 4 Application | 23 |
| I Mise en ouevre du Matérielles | 23 |
| I.1 Raspberry Pi | 23 |

| | | |
|---|---|-----------|
| I.2 | Caméra OpenMV | 23 |
| I.3 | Robot Zumo | 23 |
| I.4 | Arduino | 23 |
| II | Communication entre la caméra OpenMV et le Raspberry Pi | 24 |
| II.1 | Communication UART | 24 |
| II.2 | Snapshot | 26 |
| II.3 | Streaming | 27 |
| III | Mise en oeuvre d'application | 28 |
| Annexe | | 31 |
| III.1 | Détection du blob | 31 |
| III.2 | Snapshot | 33 |
| III.3 | Streaming | 34 |
| III.4 | Contrôle du robot à distance | 36 |
| Conclusion générale et perspective | | 47 |
| Bibliographie et Webographie | | 48 |

Introduction générale

Les systèmes embarqués sont au cœur de nombreux produits, machines et opérations intelligentes, comme les applications d'apprentissage automatique et d'intelligence artificielle. En effet les applications de systèmes embarqués apparaissent aujourd'hui dans toutes les industries et tous les secteurs, les dispositifs et logiciels embarqués jouent un rôle crucial dans le fonctionnement des voitures, des dispositifs médicaux, des bornes interactives d'autres équipements que nous utilisons au quotidien.

Dans ce projet, nous avons fourni une application d'un système embarqué qui a pour but de combiner le traitement d'images à l'aide de la caméra Open MV, et la robotique au moyen du robot zumo, pour atteindre un objectif de réalisation d'un système capable de contrôler le robot à distance, à partir d'un ordinateur avec diffusion vidéo en temps réel, à l'aide de la caméra dans un environnement compatible.

Cette application servit le téléguidage de robot commander par la RPI. Au moyen d'une interface de télécommande, depuis un autre ordinateur et via un accès à distance au bureau de l'RPI, en envoyant ces commandes par protocole série. Tout ceci en même temps que la diffusion vidéo au bureau de RPI partageant l'entourage actuel de robot pour servir la conduite du robot.

Pour y parvenir, nous avons divisé notre projet en trois parties :

- **La première partie** : Le cahier des charges.
- **La deuxième et troisième parties** : Description du matériel et des logiciels.
- **La quatrième partie** : L'embarquement de la caméra OpenMV sur le Zumo.

CHAPITRE 1

CAHIER DE CHARGE

Le projet définitif sera un robot programmable alimenté par une source d'énergie de 5 volts, capable de se déplacer sur une surface plane avec une vitesse maximale de 0,6 m/s, de tourner avec plusieurs nombres des tours. Son parcours est contrôlé et dirigé à distance à l'aide d'une sélection de commandes et de diffusion du flux vidéo de la caméra en temps réel. Des principales commandes (marche avant, marche arrière, tourner à droite ou à gauche ...etc.) ont été implémentés ainsi qu'apprendre des captures avec la caméra OpenMV. Dans l'ensemble, les pièces suivantes sont utilisées :

la carte RPI, Caméra Open MV, Robot zumo avec Arduino, un ordinateur servit le côté client de Vnc viewer et une source d'alimentation.

CHAPITRE 2

HARDWARE

I Carte Raspberry Pi

I.1 Présentation générale

Développé par la raspberry Pi Fondation en association avec Broadcom, la carte RPI s'agit d'un ordinateur monocarte, un microprocesseur construit sur une seule carte de circuit imprimé qui fournit tous les circuits nécessaires à une tâche de contrôle utile, tels que les circuits d'E/S et intégrés, une RAM, un générateur d'horloge, une mémoire de programme stocké.



FIGURE 1 – Raspberry Pi

La RPI héberge un système d'exploitation , alors elle puisse exécuter plusieurs applications simultanément.

Elle comprend un processeur ARM quad-core 1,2 G. Hz 64 bits et un réseau local sans fil, et Bluetooth. Et inclut aussi 1 Go de RAM, 4 ports USB et un support HDMI intégré.

Elle est largement utilisé dans de nombreux domaines, tels que la surveillance météorologique, en raison de son faible coût, de sa modularité et de sa conception ouverte. Il est généralement utilisé par les amateurs d'informatique et d'électronique, en raison de son adoption des normes HDMI et USB.

I.2 Description de carte RPI.

carte RPI comprend plusieurs éléments [1] :

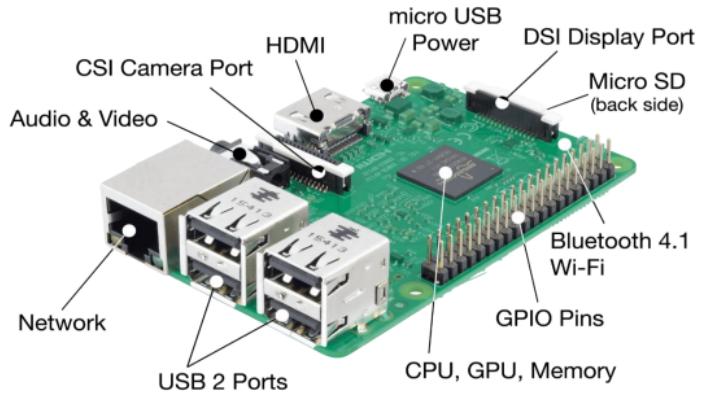


FIGURE 2 – Caption

- **CPU/GPU ARM** : Un système sur puce (Soc) Broadcom BCM2835 composés d'une unité centrale de traitement (CPU) ARM (Advanced RISC Machines) et d'une unité de traitement graphique (GPU) Video core 4.
Le CPU gère tous les calculs qui font fonctionner un ordinateur (il prend les données en entrée, effectuent des calculs et produit des résultats), et le GPU gère les résultats graphiques.
- **Pins GPIO** : Points de connexion d'E/S polyvalentes qui permettront aux vrais amateurs de matériel informatique de réparer.
- **VIDEO / AUDIO** : Une prise standard de 3,55 mm permettant de connecter des périphériques de sortie audio tels que des casques ou des haut-parleurs.
Il n'y a pas d'entrée audio. Et permet de connecter des téléviseurs analogiques et d'autres périphériques de sortie similaires.
- **USB** : C'est un port de connexion commune pour les périphériques frontaliers de tous types (y compris votre souris et votre clavier). Le RPI modèle A en possède un, et le RPI modèle B en possède deux. Vous pouvez utiliser un hub USB pour augmenter le nombre de ports ou brancher votre souris sur votre clavier s'il possède son propre port USB.
Vous pouvez utiliser un hub USB pour augmenter le nombre de ports ou brancher votre souris sur votre clavier s'il possède son propre port USB.
- **SORTIE HDMI** : Ce connecteur vous permet de brancher un téléviseur haute définition ou tout autre appareil approprié à l'aide d'un câble HDMI.
- **ALIMENTATION (MICRO USB POWER)** : Un connecteur d'alimentation micro USB 5v dans lequel vous pouvez brancher votre alimentation appropriée.
- **CARTE SD (BACK SIDE)** : Un emplacement pour carte SD de taille normale.
Une carte SD avec un système d'exploitation (OS) installé est nécessaire pour démarrer l'appareil.
Ils sont disponibles à l'achat auprès des fabricants, mais vous pouvez également télécharger un OS et le placer sur la carte vous-même si vous avez une machine

Linux et les moyens.

- **ETHERNET (NETWORK)** : Ce connecteur permet l'accès à un réseau câblé et il n'est disponible que sur le modèle B.
- **CSI Camera Port** : Facilite la connexion d'une petite caméra au processeur principal Broadcom . Il s'agit d'un port de caméra fournissant une connexion de bus électrique entre les deux appareils. Il s'agit d'une interface très simple et avec un peu de rétro-ingénierie avec un oscilloscope, il est possible de déterminer le brochage.

I.3 Pins du Raspberry Pi

Les pins sont des broches métalliques qui sortent de la carte servi à connecter des composants extérieurs à travers des câbles spéciaux sont appelées câbles de pontage.

Ces épingle sont regroupées, certains sont réservés à l'alimentation, et d'autres utiliser en connexion avec d'autres circuits ou désigner comme des broches de sortie ou d'entrée.

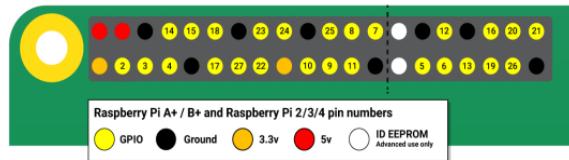


FIGURE 3 – Pins du RPI

Les broches sont numérotées de 0 à 31 sont également identifiées par une référence GPIO qui est plus simple à utiliser lors de la programmation, GPIO1, GPIO2, etc. Certains ont une fonction supplémentaire précisée entre parenthèses. Cette fonction n'empêche pas d'utiliser les GPIO de façon classique, comme E/S numérique.

Ces GPIO sont au nombre de 30. Ils ne fonctionnent qu'en tout ou rien, 0 ou 1, 0V ou 3,3. Il n'y a pas de port analogique.

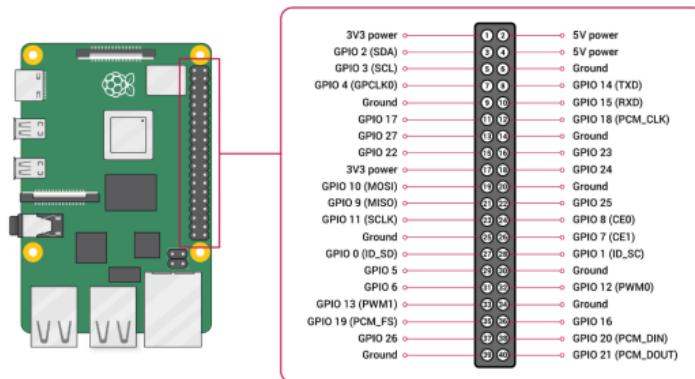


FIGURE 4 – Liste des Pins GPIO

II Caméra OpenMV

II.1 Présentaion générale

L'**OpenMV** est un système de vision intelligent comme un Arduino super puissant avec une caméra embarquée capable d'effectuer de la reconnaissance faciale et facilite l'exécution d'algorithmes de vision industrielle sur ce que voit la caméra OpenMV afin que nous permet de suivre les couleurs, détecter les visages, et les mouvements, de l'enregistrement vidéo sen GIF/MJPEG...etc. en quelques secondes, puis contrôler les broches d'E/S dans le monde réel [2].



FIGURE 5 – Caméra OpenMV

II.2 Caractéristiques du caméra OpenMV

- **MCU** : STMicro STM32H743VI Arm Cortex M7 micro-contrôleur, jusqu'à 216 MHz avec 512KB RAM, 2MB flash
- **Stockage externe** : support de carte micro SD supportant jusqu'à 100 Mbps en lecture/écriture pour enregistrer des vidéos et stocker des actifs de vision industrielle
- **Modules de caméra** : STMicro STM32H743VI Arm Cortex M7 micro-contrôleur, jusqu'à 400 MHz avec 1MB RAM, 2MB flash
- **USB** : Interface micro USB à pleine vitesse (12 Mbps), la carte apparaissant comme un port COM virtuel et une clé USB dans votre ordinateur.
- **E/S :**
 - 1x bus SPI jusqu'à 100 Mbps, 1x I2C
 - Bus CAN, Bus Série Asynchrone (TX/RX)
 - ADC 12 bits, DAC 12 bits

- 3x broches I/O pour le contrôle du servo
- Interruptions et PWM sur toutes les broches I/O (10 broches I/O sur la carte)

- **Divers :**

- LEDs RVB, 2x LEDs IR 850nm haute puissance
- Alimentation électrique
- 5V via le port micro USB
- Connecteur pour piles LiPo compatible avec les piles LiPo 3,7V
- Dimensions - 45 x 36 mm - Poids - 19 grammes

II.3 Modèles caméra OpenMV M7

- Processeur : ARM® 32-bit Cortex®-M7 CPU 216 MHz
- Mémoire RAM : 384KB SDRAM
- Mémoire Flash :
- Formats d'image supporté :
 - RGB565 : 320x240 and under
 - Grayscale JPEG : 640x480 and under
 - RGB565 JPEG : 640x480 and under
- Focal Length : 2.8mm
- Aperture : F2.0
- IR Cut Filter : 650nm (removable)

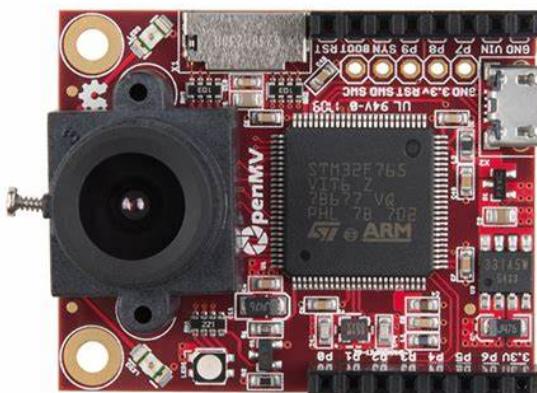


FIGURE 6 – Caméra OpenMV M7

III Robot Zumo

Le robot Zumo est un robot entièrement assemblé à un micro-contrôleur compatible Arduino intégré à la plate-forme robotique.[9]



FIGURE 7 – Zummo Robot

Les encodeurs pour le contrôle du moteur en boucle fermée et les capteurs de proximité pour le suiveur de ligne le rendent suffisamment polyvalent pour servir de petit robot à usage général.

IV Arduino

La carte Arduino est une carte électronique porte micro-contrôleur programmable et open source, utilisée pour réaliser des montages électriques [10].

Facile d'utilisation, elle permet de s'initier aisément à l'électronique et à la programmation. Elle devra se connecter sur un ordinateur pour permet le téléchargement des commandes à exécuter (le code ou le programme). C'est une plateforme basée sur une interface E/S simple et sur un environnement de développement utilisant la technique du processing/Wiring.

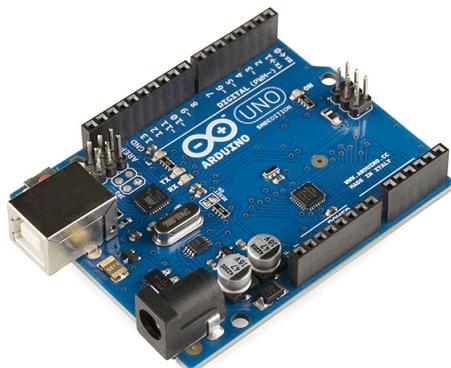


FIGURE 8 – Arduino Uno

CHAPITRE 3

SOFTWARE

I Système d'Exploitation Raspbian

Le système d'exploitation basé sur Linux Raspbian (un dérivé de Debian) et il est très régulièrement mis à jour. Officiellement supporté par la Fondation Raspberry Pi. Il s'agit donc d'un système d'exploitation spécifiquement optimisé pour une utilisation avec Raspberry Pi, polyvalent qui permettre defamiliariser très facilement avec le matériel.[7]



FIGURE 1 – Raspbian

Cependant, Raspbian fournit plus qu'un système d'exploitation pur, il permet :

- Livrer avec plus de 35000 paquets, des logiciels précompilés regroupés dans un format agréable pour une installation facile sur votre Raspberry Pi
- Fournir plusieurs environnements de programmation, BlueJ Java IDE, python 2, Mathematica....
- Aux applications de gagner la même performance grâce à l'utilisation des instructions avancées du CPU ARMv6 du Raspberry Pi.

II OpenMV IDE

OpenMV IDE est le premier environnement de développement intégré à utiliser avec votre OpenMV Cam. Il dispose d'un puissant éditeur de texte, d'un terminal de débogage et d'une visionneuse de tampon de cadre avec un affichage d'histogramme. OpenMV IDE facilite la programmation de votre OpenMV Cam

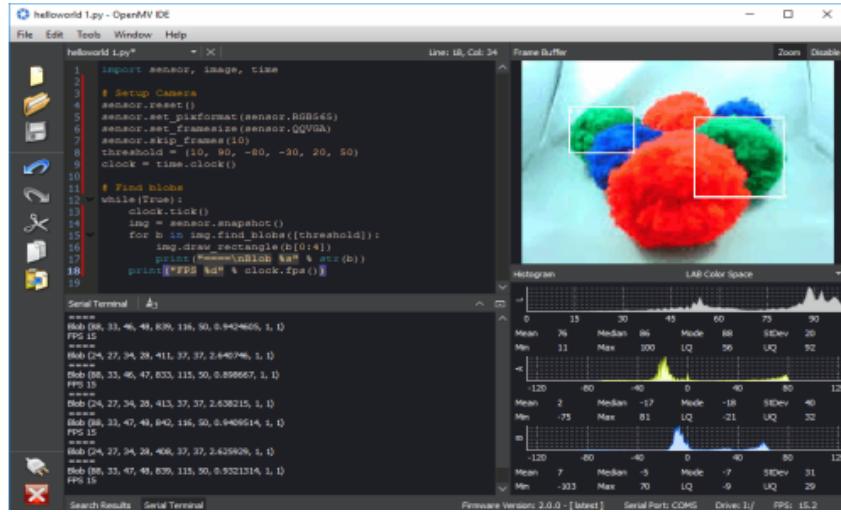


FIGURE 2 – Interface d’OpemMV IDE

II.1 Bibliothèque d’interface

L’**OpenMV Cam** est livré avec une bibliothèque RPC (Remote Python/Procedure Call) qui facilite la connexion de l’OpenMV Cam à notre ordinateur, un SBC (single board computer) comme le RaspberryPi ou le Beaglebone, ou un microcontrôleur comme l’Arduino ou l’ESP8266/32. La bibliothèque d’interface RPC fonctionne sur :

- Série asynchrone (UART) - jusqu’à 7,5 Mb/s
- Bus I2C - jusqu’à 1 Mb/s.
- Bus SPI - jusqu’à 80 Mb/s

Avec la bibliothèque RPC, nous pouvons facilement obtenir des résultats de traitement d’image, diffuser des données d’image RAW ou JPG, ou faire en sorte que l’OpenMV Cam contrôle un autre microcontrôleur pour un contrôle matériel de bas niveau, comme la commande de moteurs.

II.2 Bibliothèque RPC

Le module RPC sur l’OpenMV Cam vous permet de connecter votre OpenMV Cam à un autre microcontrôleur ou ordinateur et d’exécuter des appels python (ou procédure) à distance sur votre OpenMV Cam. Le module RPC permet également l’inverse si vous souhaitez que votre OpenMV Cam puisse exécuter des appels de procédure (ou python) à distance sur un autre microcontrôleur ou ordinateur [5].

III Arduino IDE

Le logiciel open source Arduino (IDE) est une interface rapide, puissante et réactive comprenant un éditeur avec un débogueur en direct qui facilite l'écriture de code et son téléchargement sur n'importe quelle carte Arduino. Ces programmes écrits ou importés à l'aide du logiciel sont appelés des croquis (Sketch) et sont enregistrés avec l'extension de fichier .ino, afin d'être vérifié puis exporté dans la carte qui exécute le script en boucle. L'éditeur dispose de fonctionnalités de navigation et manipulation dans le code et configuration de port série, accompagné d'une zone de message (Console) qui affiche la sortie texte du logiciel et fournit des commentaires, des messages d'erreur de débogage et d'autre informations lors de l'enregistrement et de l'exportation du code.

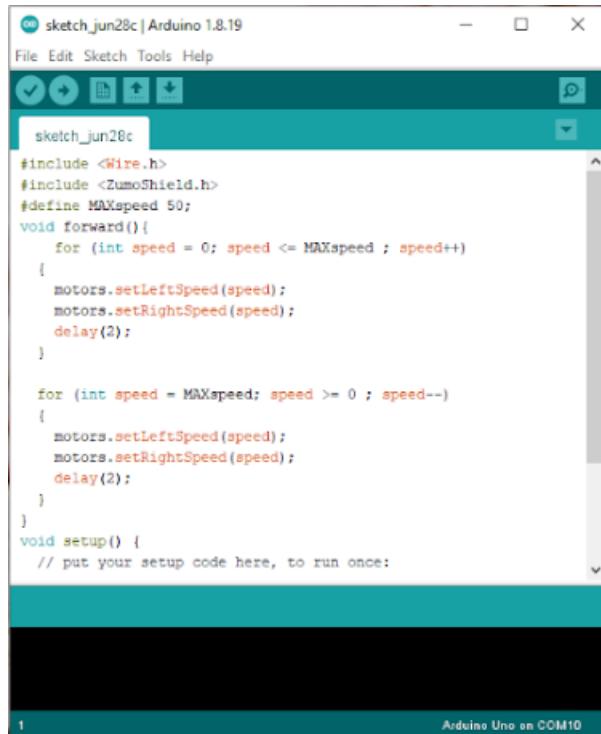


FIGURE 3 – Arduino IDE

IV VNC Viewer

Virtual Network Computing (VNC) est un système graphique de partage de bureau qui utilise le protocole RFB pour contrôler à distance un autre ordinateur. Il transmet les entrées du clavier et de la souris d'un ordinateur à un autre, relayant les mises à jour de l'écran graphique, sur un réseau.



FIGURE 4 – VNC

V Bibliothèques

V.1 Pyserial

Une bibliothèque du langage de programmation Python d'accès au port série. Il présente de nombreuses caractéristiques. [3] :

- Fonctionnement avec ou sans délai de réception.
- Le port est configuré pour une transmission binaire. Pas de suppression de l'octet NUL, cela rend ce module universellement utile.
- Compatible avec la bibliothèque IO E/S.



FIGURE 5 – Pyserial

V.2 Micropython

La bibliothèque MicroPython fournit les fonctionnalités de base d'OpenMV. Plus des modules standard du langage python, voici quelques modules de micropython utilisés pour programmer la camera open MV [4] :

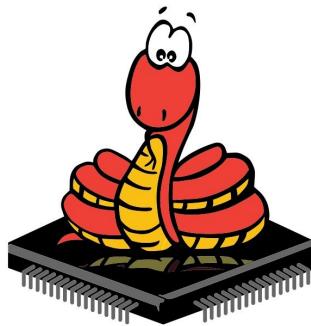


FIGURE 6 – Micropython

- **Pyb** : Contient des fonctions spécifiques liées à la carte.
- **Sensor** : Contrôler le capteur et les paramètres de la caméra.
- **Image** : Utilisé pour la vision artificielle et blob détection.

- **Time** : Fournit des fonctions pour obtenir l'heure et la date actuelles, mesurer les intervalles de temps et les retards.
- **Struct** : Compresser et décompresser les types de données primaires.
- **Os** : Contient des fonctions pour l'accès et le montage du système de fichiers, la redirection et la duplication de terminaux.
- **Math** : Offre certaines fonctions mathématiques de base pour le travail avec des nombres à virgule flottante.
- **json** : Convertir entre les objets Python et le format de données JSON.
- **Machine** : Contient des fonctions spécifiques liées au matériel permettant d'accéder et de contrôler directement et sans restriction les blocs matériels d'un système (comme le processeur, les temporiseurs, les bus, etc).

V.3 Zumo Shield

Une bibliothèque compatible pour un microcontrôleur Arduino pilotant le robot Zumo.. il fournit des fonctions pour vous aider à interagir avec le Pololu Zumo Shield, le kit de robot Zumo et le Zumo Reflectance Sensor Array.

Les fonctionnalités des classes fournies par la librairie sont les suivantes :

- **ZumoMotors** : Contrôler la vitesse et la direction du moteur.
- **ZumoBuzzer** : Émet des signaux sonores et de la musique sur le Zumo Shield.
- **ZumoReflectanceSensorArray** : Lire à partir du réseau de capteurs de réflectance.

VI Protocoles de communication :

- **Communication Série :**

L'interface série est un ensemble de protocoles qui communiquent entre hôtes. C'est un moyen de transférer des informations en série ou en tranches bit par bit, parallèlement au microcontrôleur ou à un autre ordinateur récepteur qui regroupe ces bits à la tranche d'origine transférée.

Protocole UART :

est un protocole série asynchrone, ce qui signifie pas d'horloge partagée pour s'assurer la connexion en même temps. lorsque l'UART récepteur détecte un bit de démarrage, il commence à lire les bits entrants à une fréquence spécifique appelée baudrate.

Les deux UART doivent également être configurés pour transmettre et recevoir la même structure de paquets de données.

— à l'aide des gpio :

UART est qu'il n'utilise que deux fils pour transmettre les données entre les appareils. Les données circulent de la broche Tx de l'UART émetteur vers la broche Rx de l'UART récepteur.

— à l'aide d'adaptateur USB :

Pour la communication USB, il existe un hôte USB, ou le périphérique maître, qui initie toutes les communications qui se produisent sur le bus. Le dispositif périphérique, ou le dispositif esclave, est connecté à l'hôte et est programmé pour fournir à l'hôte les informations requises pour fonctionner avec succès.

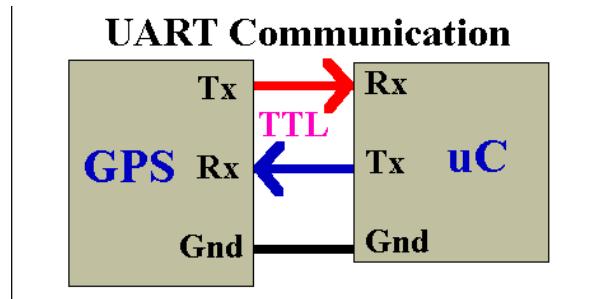


FIGURE 7 – protocole UART

- **RPC :**

RPC est un protocole réseau permettant de faire des appels de procédures sur un ordinateur distant à l'aide d'un serveur d'applications. Ce protocole est utilisé dans le modèle client-serveur pour assurer la communication entre le client, le serveur et d'éventuels intermédiaires. Même principe s'applique dans le modèle maître esclave (master-slave).

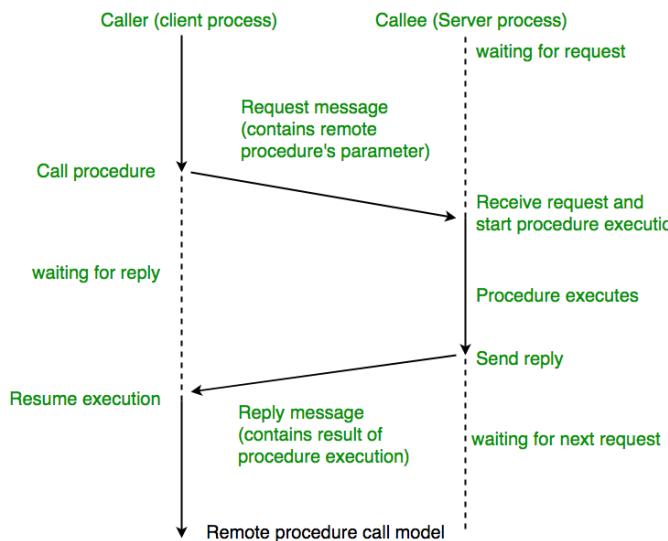


FIGURE 8 – protocole RPC

CHAPITRE 4

APPLICATION

I Mise en oeuvre du Matérielles

I.1 Raspberry Pi

Pour familiariser avec la carte Raspberry Pi nous avons passé par plusieurs étapes, premièrement par l'installation de systèmes d'exploitation Raspbian la plus compatible avec la carte RPI, deuxièmement par l'étude et la connaissance de ces composant spécialement les pins, et dernièrement par des simulations comme exemple alimentation des LEDS a l'aide de la carte Raspberry Pi

I.2 Caméra OpenMV

La caméra OpenMV offre plusieurs fonctions pour la reconnaissance et le traitement des images qui donnent une vision intelligente, alors pour entraîner l'utilisation de cette caméra nous avons essayé de connaître ses modules et ses bibliothèques, et tester quelques exemples en utilisant l'éditeur OpenMv : détection de visage, détection des yeux...

I.3 Robot Zumo

Afin de nous familiariser avec le robot Zumo, nous avons testé certaines fonctions de commande du mouvement du robot, notamment l'exemple de tracker de ligne.

I.4 Arduino

Pour maîtriser bien la carte arduino nous avons traité plusieurs simulations par exemple l'alimentation du led, au début on nous avons travailler sur le simulation en ligne d'arduino.

II Communication entre la caméra OpenMV et le Raspberry Pi

II.1 Communication UART

Objectif :

Transfert des coordonnées hexadécimales de la position des objets détectés par la caméra OpenMV au RPI par le protocole de communication série UART [6]. Ces objets sont détectés à l'aide d'algorithmes de détection de blob, intégré au module image de la bibliothèque micropython.

Définition et description de détection de présence et position des blobs :

La détection d'objets blob permet de détecter des zones dans une image numérique qui diffèrent par des propriétés telles que la luminosité ou la couleur par rapport aux espaces environnantes. Ces zones sont appelées blobs, correspondant à des morceaux de pixels similaires décrits par des seuils de couleur.

Afin d'alimenter l'algorithme de détection de blob avec une image ; on capture un instantané de caméra Open MV. L'image résultante doit ensuite être transmise en paramètre à la fonction find blobs avec une liste de tuples spécifiant les valeurs de couleur LAB qui sont contenues dans les blobs souhaités à détecter. Par exemple, si l'il s'agit d'un blob purement rouge sur un fond noir, la gamme de couleurs résultante serait la valeur LAB correspondante pour le rouge pur (53,80, 67). L'algorithme analysera alors l'image et affichera les coordonnées des blobs trouvées. Afin d'envoyer ces coordonnées retournées par la fonction directement au port série du RPI.

La liaison entre la caméra et la RPI :



FIGURE 1 – Montage

Résultat :

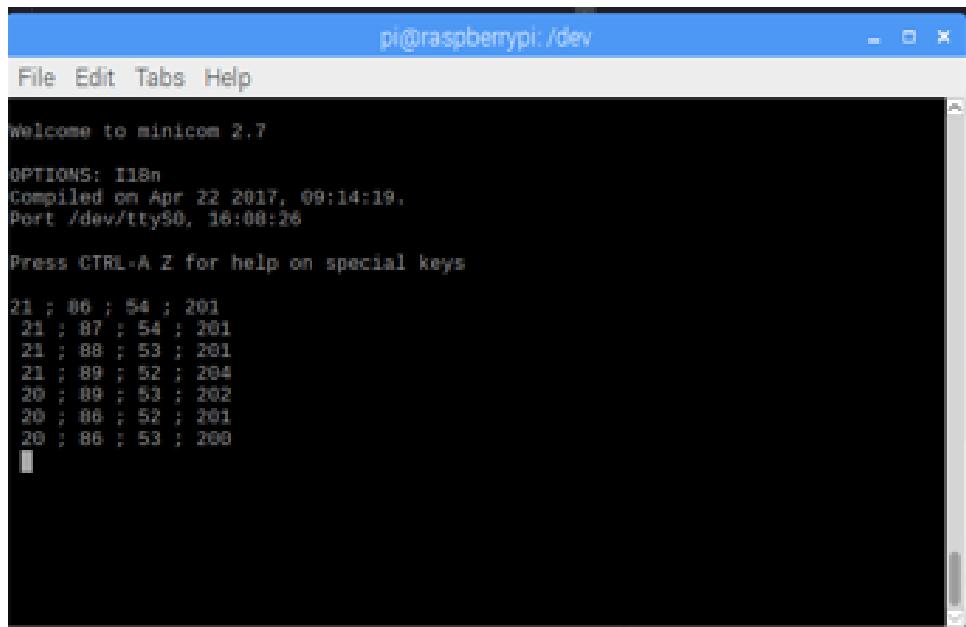


FIGURE 2 – Terminal de RPI

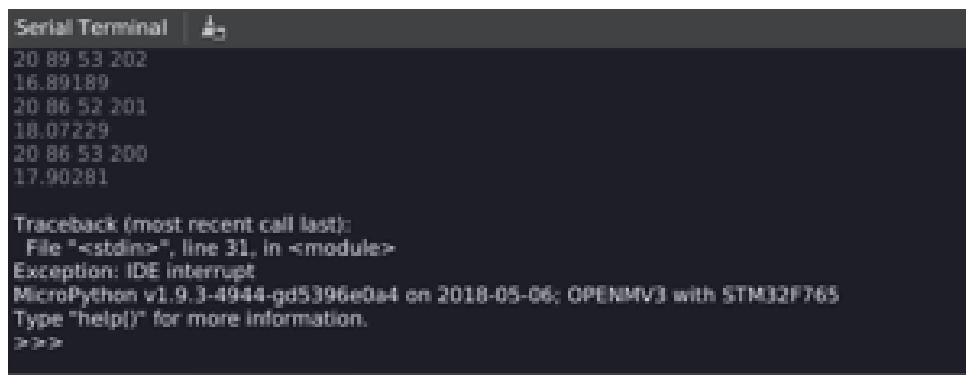


FIGURE 3 – Serial terminal du OpenMV IDE

Description du Résultat :

les résultats de détection du blob sont affichés sous forme des coordonnées du rectangle qui entoure le blob détecté qui sont blob x, blob y, blob w, blob h, ce résultat affiche dans le serial terminal de l'éditeur OpenMV IDE pour la caméra, et dans le terminal du RPI en utilisant Minicom qui s'agit d'un programme lisant les données passées via les ports de communication série, pour détecter ces transmitions passant avec un baud rate de 9600 signifiant 9600 bits par seconde.

II.2 Snapshot

Objectif

la RPI contrôle la caméra open MV pour prendre des images, puis les transférer codés en octets, enfin de les stocker dans son système de fichiers.

Description :

Nous capturons, compressons et décodons des images en octets grâce à la caméra Open MV. puis les stockées en raspberry pi qui Contrôle via USB la caméra avec l'aide de classes fournies par la bibliothèque RPC.

Au début la raspberry cherche les ports disponibles pour choisir le port USB liée au OpenMV Cam. après la connexion de port, elle crée un appel maître RPC en utilisant le protocole RPC vers la caméra qui joue le rôle d'esclave, et cela répond à l'exécution des fonctions trouvées dans la main de la caméra pour capturer des images, et stockées dans une variable. Si les images capturées ne sont pas vides, elle les stocke avec la date de capture en tant que nom et affiche les résultats dans le terminal, sinon il affiche None signifiant aucune capture.

Hors d'appel de RPI (Master), la fonction main de l'open Mv (Slave) capture les images avec une résolution du QVGA, pixformat RGB565 et les compresses avec haute qualité, après les envoyées vers le port connecté au RPI.

Résultat :

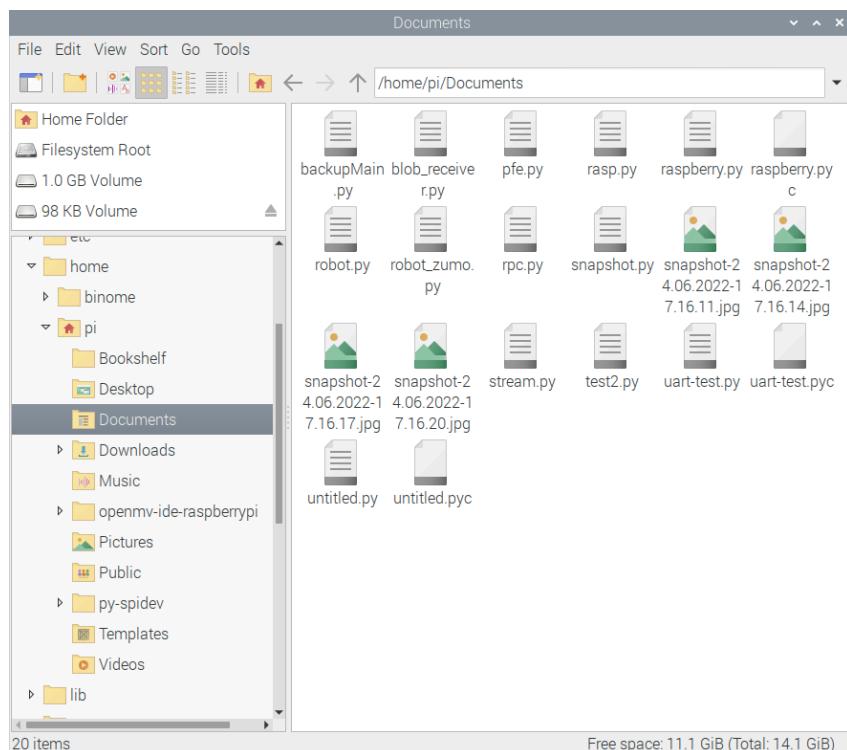


FIGURE 4 – Stockage des images dans RPI



FIGURE 5 – Image par Caméra OpenMV

Description du Résultat :

Caméra OpenMV capture les images et le compresses ,puis le stocker dans le raspberry pi dans le fichier ce qu'on veut,dans ce cas on enregistre juste dans le même répertoire qui contient ce code.(on a choisi le repertoire Documents du RPI), les images sont stocker avec leur date de capturation ,et une qualité moyenne (on peut changer la résolution)

II.3 Streaming

Objectif

Transférer la diffusion vidéo sur l'IDE de Cam OpenMV vers notre RPI, sous forme d'image JPEG.

Description

Commençons par un appel RPC Master depuis la RPI. Qui se termine avec un appel réponse Slave RPC de la caméra, suivi des paramètres de la configuration du capteur image de cette dernière, taille et format des images prises leur de la diffusion. Pour générer un buffer de streaming de même taille. Ainsi qu'une séquence d'images capturées en temps réel est compressée en haute qualité. afin de mettre fin à la diffusion et à la réponse RPC Slave.

Résultat :



FIGURE 6 – Video en direct par OpenMV capturant la RPI

Description de résultat :

Le résultat est sous forme d'un frame buffer qui affiche le stream en temps réels, avec une qualité moyenne.

III Mise en oeuvre d'application

Objectif :

Commande à distance d'un robot zumo, à partir d'une connexion Bureau à distance de RPI. Nous exécutons deux programmes dans le terminal de celle-ci. L'un chargé à présenter une interface télécommande affichant plusieurs options de mobilité et qui gère les choix d'utilisateurs et un autre servit la diffusion vidéo, décrivant champ de vision de robot en temps réel à l'aide de caméras open MV.

Description :

La programmation des fonctionnalités du mouvement est développée par un programme arduino permettant de piloter les deux moteurs, à l'aide de la bibliothèque Zumo shield et de ses fonctions sur son module motor. Ce programme aussi s'interagit avec des chaînes de caractères présentant les commandes, reçues du programme python exécuté sur la RPI qui est chargée de la transmission de ces chaînes symbolisent l'entrée du clavier à l'aide du module curses. Cette transmission de chaînes se fait par une communication série entre ses deux composants.

Alors que la programmation de diffusion du champ de vision est citée précédemment.

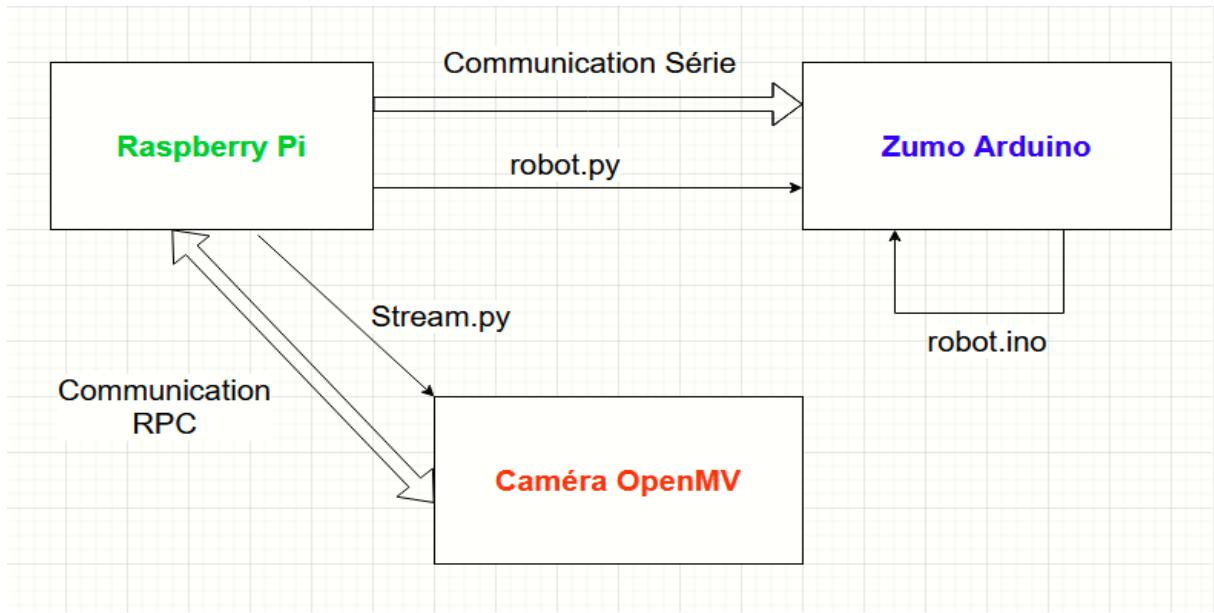


FIGURE 7 – Schéma d’application

Montage d’application :



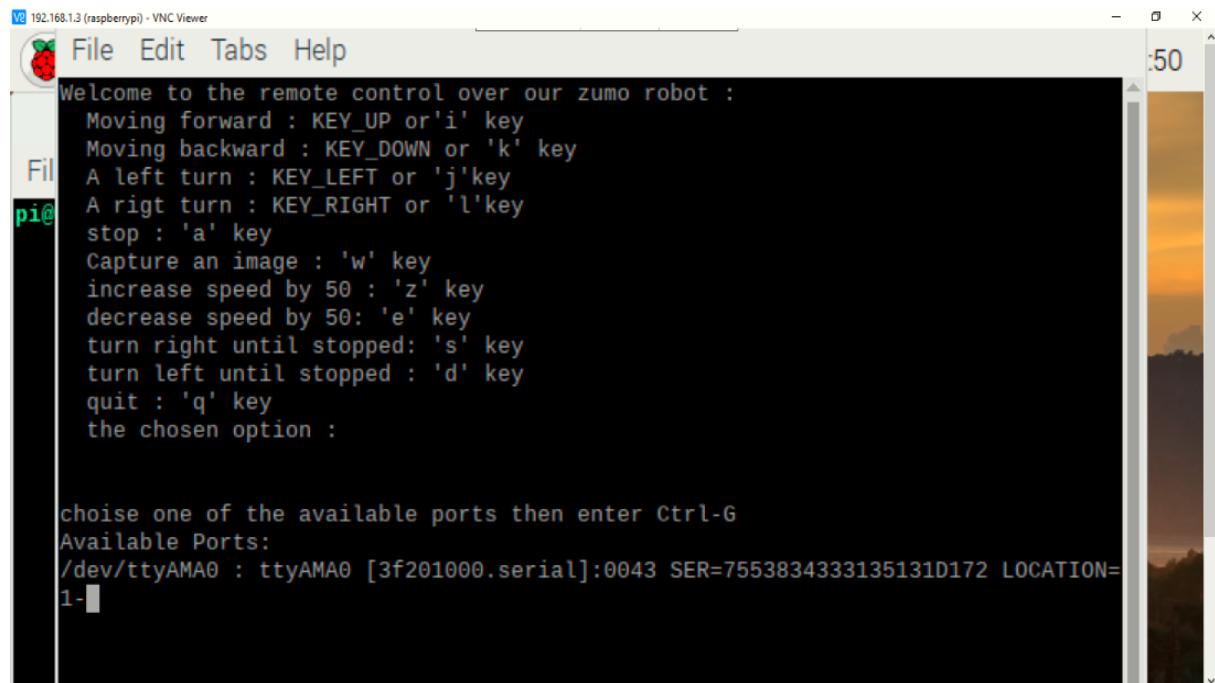
FIGURE 8 – Montage

Dans l’assemblage courant, le robot zumo et son microcontrôleur se connectent directement au pi en utilisant un adaptateur USB. Lequel, à son tour, est lié à la caméra et sur l’accès à distance assistant par le VNC, qui nous permet un accès complet aux différentes ressources du dernier et en particulier l’écran de bureau dans un autre appareils.

Interface de contrôle du Robot :

La table décrit les listes de commandes disponibles :

| Fonctionnalité | keys |
|------------------------------|-------------------------------|
| Avancée | KEY Up, 'i' |
| Retour en arrière | KEY down, 'k' |
| Tournée à gauche ou à droite | KEY LEFT, KEY RIGHT, 'j', 'l' |
| Augmentation du vitesse | 'z' |
| Diminution du vitesse | 'e' |
| Capture d'image | 'w' |
| Quitter | 'q' |



```
VNC 192.168.1.3 (raspberrypi) - VNC Viewer
File Edit Tabs Help
Welcome to the remote control over our zumo robot :
Moving forward : KEY_UP or 'i' key
Moving backward : KEY_DOWN or 'k' key
A left turn : KEY_LEFT or 'j'key
A right turn : KEY_RIGHT or 'l'key
stop : 'a' key
Capture an image : 'w' key
increase speed by 50 : 'z' key
decrease speed by 50: 'e' key
turn right until stopped: 's' key
turn left until stopped : 'd' key
quit : 'q' key
the chosen option :

choose one of the available ports then enter Ctrl-G
Available Ports:
/dev/ttymA0 : ttymA0 [3f201000.serial]:0043 SER=7553834333135131D172 LOCATION=
1-
```

FIGURE 9 – Interface

Annexe

III.1 Détection du blob

Code Pour Raspberry Pi :

```
blob_receiver.py ✘
1 import serial
2 import string
3 import time
4
5 class OpenMV(object):
6
7     def __init__(self):
8         self.ser = serial.Serial('/dev/ttyS0',baudrate=9600, timeout = 0.5)
9         self.ser.flushInput();
10
11    def blob_xysize(self):
12        print("blobxysize")
13
14        blob_found = False
15        blob_x = 0
16        blob_y = 0
17        blob_radius = 0
18        w = 0
19
20        self.ser.flushInput();
21        print("1")
22        line = self.ser.readline()
23        print("2")
24
25        print(line.decode('utf-8'))
26        words = string.split(line , ' ; ')
```

FIGURE 10 – Raspberry PI Blob (partie 1)

```
25        print(line.decode('utf-8'))
26        words = string.split(line , ' ; ')
27        if len(words) > 3:
28            blob_found = True
29            blob_x = int (words [0])
30            blob_y = int (words [1])
31            w = int (words [2])
32            blob_radius = w
33
34        else:
35            blob_found = False
36            blob_x = 0
37            blob_y = 0
38            blob_radius = 0
39            w = 0
40
41
42        print(blob_found, blob_x, blob_y, blob_radius)
43
```

FIGURE 11 – Raspnerry PI Blob (partie 2)

```

def main(self):
    while(True):
        self.blob_xysize()

        exit(1)
    self.ser.close()

open_mv=OpenMV()

if __name__=="__main__":
    open_mv.main()

```

FIGURE 12 – Raspberry PI Blob (partie 3)

Code Pour Caméra OpenMV :

```

helloworld_1.py*  ▾ | X | Line: 27, Col: 1
1 # Untitled - By: Kawthar - Mon Jun 20 2022
2
3 import image, math, pyb, sensor, struct, time
4 uart_baudrate = 9600
5 uart = pyb.UART(3, uart_baudrate, timeout_char =1)
6 clock = time.clock()
7
8 threshold_index = 0
9
10 thresholds = [(23, 48, -128, 74, -37, 66), # generic_red_thresholds
11 (30, 100, -64, -8, -32, 32), # generic_green_thresholds
12 (0, 30, 0, 64, -128, 0)] # generic_blue_thresholds
13
14 sensor.reset()
15 sensor.set_pixformat(sensor.RGB565)
16 sensor.set_framesize(sensor.QVGA)
17 sensor.skip_frames(30)
18 sensor.set_auto_gain(False) # must be turned off for color tracking
19 sensor.set_auto_whitebal(False) # must be turned off for color tracking
20 sensor.skip_frames(time = 2000)
21
22 while(True):
23     clock.tick()
24     start = pyb.millis()
25     img = sensor.snapshot()
26     for blob in img.find_blobs([thresholds[threshold_index]], pixels_threshold=100, area_threshold=100,
27     merge=False):
28         img.draw_rectangle(blob.rect())
29         img.draw_cross(blob.cx(), blob.cy())
30
31         print(blob.cx(), blob.cy(), blob.w(),blob.h())
32
33         uart.write("%d ; %d ; %d ; %d \r\n" % (blob.cx(), blob.cy(),blob.w(),blob.h()))
34     print(clock.fps())
35

```

FIGURE 13 – Caméra OpenMV

III.2 Snapshot

Code Pour Raspberry Pi :

```
1 import json, rpc, serial, serial.tools.list_ports, struct, sys
2 from datetime import datetime
3
4 print("\nAvailable Ports:\n")
5 for port, desc, hwid in serial.tools.list_ports.comports():
6     print("{} : {} [{}]".format(port, desc, hwid))
7 sys.stdout.write("\nPlease enter a port name: ")
8 sys.stdout.flush()
9 interface = rpc.rpc_usb_vcp_master(port=input())
10 print("")
11 sys.stdout.flush()
12
13 def exe_jpeg_snapshot():
14     result = interface.call("jpeg_snapshot")
15     if result is not None:
16         name = "snapshot-%s.jpg" % datetime.now().strftime("%d.%m.%Y-%H.%M.%S")
17         print("Writing jpeg %s..." % name)
18         with open(name, "wb") as snap:
19             snap.write(result)
20
21 while(True):
22
23     | exe_jpeg_snapshot()
```

FIGURE 14 – Raspberry PI (Snapshot)

Code pour Caméra OpenMV :

```
1 import image, network, math, rpc, sensor, struct, tf
2
3 sensor.reset()
4 sensor.set_pixformat(sensor.RGB565)
5 sensor.set_framesize(sensor.QVGA)
6 sensor.skip_frames(time = 2000)
7 interface = rpc.rpc_usb_vcp_slave()
8
9 def jpeg_snapshot(data):
10    sensor.set_pixformat(sensor.RGB565)
11    sensor.set_framesize(sensor.QVGA)
12    return sensor.snapshot().compress(quality=90).bytarray()
13
14
15 interface.register_callback(jpeg_snapshot)
16
17 interface.loop()
```

FIGURE 15 – Caméra OpenMV (Snapshot)

III.3 Streaming

Code pour Raspberry Pi :

```
1 import io, pygame, rpc, serial, serial.tools.list_ports, socket, sys
2 try: input = raw_input
3 except NameError: pass
4 |
5 print("\nAvailable Ports:\n")
6 for port, desc, hwid in serial.tools.list_ports.comports():
7     print("{} : {} [{}]".format(port, desc, hwid))
8 sys.stdout.write("\nPlease enter a port name: ")
9 sys.stdout.flush()
10 interface = rpc.rpc_usb_vcp_master(port=input())
11 print("")
12 sys.stdout.flush()
13 pygame.init()
14 screen_w = 640
15 screen_h = 480
16 try:
17     screen = pygame.display.set_mode((screen_w, screen_h), flags=pygame.RESIZABLE)
18 except TypeError:
19     screen = pygame.display.set_mode((screen_w, screen_h))
20 pygame.display.set_caption("Frame Buffer")
21 clock = pygame.time.Clock()
22
23 def jpg_frame_buffer_cb(data):
24     sys.stdout.flush()
```

FIGURE 16 – Raspberry PI (Streaming)

```
24 sys.stdout.flush()
25 try:
26     screen.blit(pygame.transform.scale(pygame.image.load(io.BytesIO(data)), "jpg"), (screen_w, screen_h)), (0, 0))
27     pygame.display.update()
28     clock.tick()
29 except pygame.error: pass
30
31 print(clock.get_fps())
32
33 for event in pygame.event.get():
34     if event.type == pygame.QUIT:
35         pygame.quit()
36         quit()
37
38 while(True):
39     sys.stdout.flush()
40     result = interface.call("jpeg_image_stream", "sensor.RGB565,sensor.QQVGA")
41     print(result)
42     if result is not None:
43         interface.stream_reader(jpg_frame_buffer_cb, queue_depth=8)
44
45     for event in pygame.event.get():
46         if event.type == pygame.QUIT:
47             pygame.quit()
48             quit()
49
```

FIGURE 17 – Raspberry PI (Streaming)

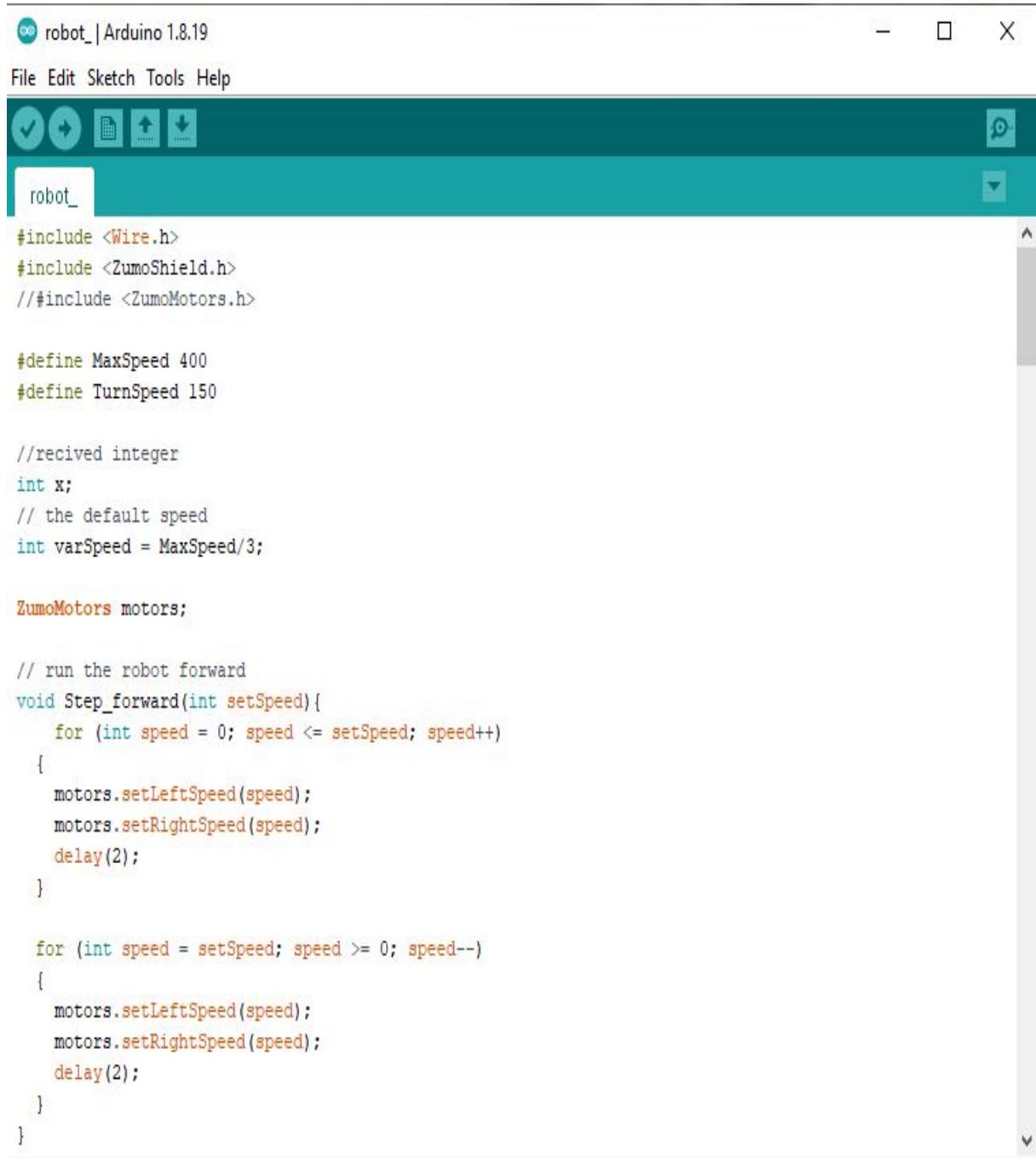
Code pour Caméra OpenMV :

```
1 import network, omv, rpc, sensor
2 sensor.reset()
3 sensor.set_pixformat(sensor.RGB565)
4 sensor.set_framesize(sensor.QVGA)
5 sensor.skip_frames(time = 2000)
6
7 omv.disable_fb(True)
8 interface = rpc.rpc_usb_vcp_slave()
9
10 def stream_generator_cb():
11     return sensor.snapshot().compress(quality=90).bytarray()
12
13 def jpeg_image_stream_cb():
14     interface.stream_writer(stream_generator_cb)
15
16 def jpeg_image_stream(data):
17     pixformat, framesize = bytes(data).decode().split(",")
18     sensor.set_pixformat(eval(pixformat))
19     sensor.set_framesize(eval(framesize))
20     interface.schedule_callback(jpeg_image_stream_cb)
21     return bytes()
22
```

FIGURE 18 – Caméra OpenMV (Streaming)

III.4 Contrôle du robot à distance

Code pour Arduino :



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** robot_ | Arduino 1.8.19
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Includes icons for Save, Run, Stop, Upload, Download, and a magnifying glass.
- Sketch Name:** The sketch is titled "robot_" in the title bar.
- Code Area:** The main window contains the following C++ code for a Zumo robot control:

```
#include <Wire.h>
#include <ZumoShield.h>
//#include <ZumoMotors.h>

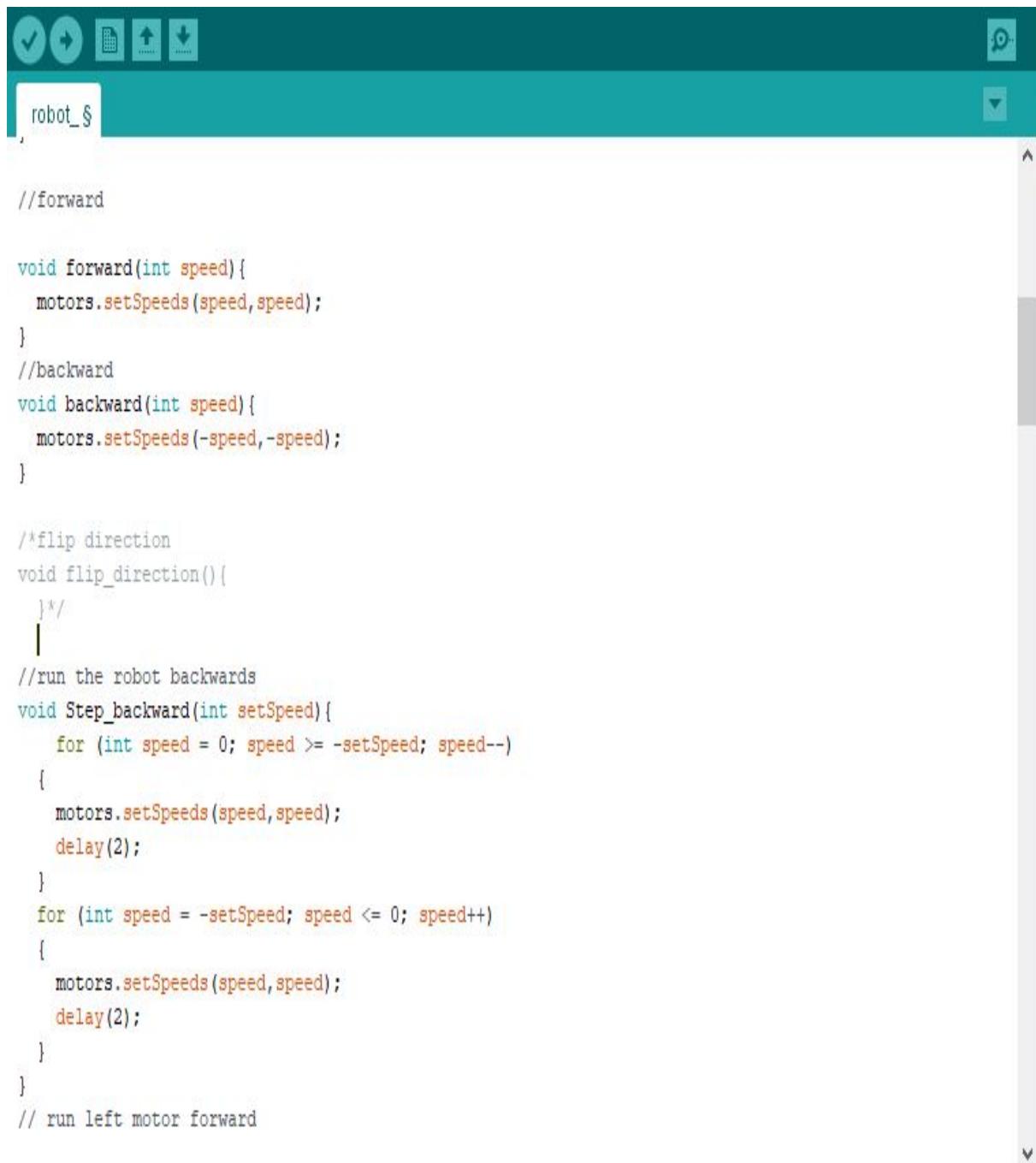
#define MaxSpeed 400
#define TurnSpeed 150

//received integer
int x;
// the default speed
int varSpeed = MaxSpeed/3;

ZumoMotors motors;

// run the robot forward
void Step_forward(int setSpeed){
    for (int speed = 0; speed <= setSpeed; speed++)
    {
        motors.setLeftSpeed(speed);
        motors.setRightSpeed(speed);
        delay(2);
    }

    for (int speed = setSpeed; speed >= 0; speed--)
    {
        motors.setLeftSpeed(speed);
        motors.setRightSpeed(speed);
        delay(2);
    }
}
```



```
//forward

void forward(int speed){
    motors.setSpeeds(speed,speed);
}

//backward
void backward(int speed){
    motors.setSpeeds(-speed,-speed);
}

/*flip direction
void flip_direction(){
    */
}

//run the robot backwards
void Step_backward(int setSpeed){
    for (int speed = 0; speed >= -setSpeed; speed--)
    {
        motors.setSpeeds(speed,speed);
        delay(2);
    }
    for (int speed = -setSpeed; speed <= 0; speed++)
    {
        motors.setSpeeds(speed,speed);
        delay(2);
    }
}
// run left motor forward
```

```
robot_
```

```
// run left motor forward
void leftMotoeForward(int turnSpeed){
    for (int speed = 0; speed <= turnSpeed; speed++)
    {
        motors.setLeftSpeed(speed);
        delay(2);
    }

    for (int speed = turnSpeed; speed >= 0; speed--)
    {
        motors.setLeftSpeed(speed);
        delay(2);
    }
}

// run left motor backward
void leftMotorBackwards(int turnSpeed){

    for (int speed = 0; speed >= -turnSpeed; speed--)
    {
        motors.setLeftSpeed(speed);
        delay(2);
    }

    for (int speed = -turnSpeed; speed <= 0; speed++)
    {
        motors.setLeftSpeed(speed);
        delay(2);
    }
}
```

```
robot_| Arduino 1.8.19
File Edit Sketch Tools Help
robot_
// run right motor forward
void rightMotorForward(int turnSpeed){
for (int speed = 0; speed <= turnSpeed; speed++)
{
  motors.setRightSpeed(speed);
  delay(2);
}

for (int speed = turnSpeed; speed >= 0; speed--)
{
  motors.setRightSpeed(speed);
  delay(2);
}
}

// run right motor backward
void rightMotorBackwards(int turnSpeed){

for (int speed = 0; speed >= -turnSpeed; speed--)
{
  motors.setRightSpeed(speed);
  delay(2);
}

for (int speed = -turnSpeed; speed <= 0; speed++)
{
  motors.setRightSpeed(speed);
  delay(2);
}
delay(500);
}
```

```
//turn robot
void turnLeftTurn(int turnSpeed) {
    for (int speed = 0; speed <= turnSpeed; speed++)
    {
        motors.setRightSpeed(speed);
        motors.setLeftSpeed(-speed);
        delay(2);
    }
    for (int speed = turnSpeed; speed <= 0; speed++)
    {
        motors.setRightSpeed(speed);
        motors.setLeftSpeed(-speed);
        delay(2);
    }
    delay(500);
}
```

```
void setup()
{
    Serial.begin(115200);
    Serial.setTimeout(1);

    // uncomment one or both of the following lines if your motors directions need to be flipped
    //motors.flipLeftMotor(true);
    //motors.flipRightMotor(true);
}
```

```
robot_
```

```
void loop()
{
    while (!Serial.available());
    x = Serial.readString().toInt();
    switch(x){
        //forward curses.key_up
        case 1: forward(varSpeed);
        break;
        //backward curses.key_down
        case 2: backward(varSpeed);
        break;
        //turn left curses.key_left
        case 3: turnLeftTurn(varSpeed);
        break;
        //turn right
        case 4: turnRightTurn(varSpeed);
        break;
        //case of capture and break belongs to the py script
        //stop
        case 5: motors.setSpeeds(0,0);
        break;
        case 6: if( varSpeed >= 0 && varSpeed < MaxSpeed-50 ) varSpeed+=50;
        break;
        //decrease speed
        case 7: if( varSpeed > 50 && varSpeed <= MaxSpeed ) varSpeed-=50;
        break;
        case 8: turnRight(MaxSpeed/2);
        break;
    }
}
```

robot_ | Arduino 1.8.19

File Edit Sketch Tools Help

robot_

```
break;
//turn left curses.key_left
case 3: turnLeftTurn(varSpeed);
break;
//turn right
case 4: turnRightTurn(varSpeed);
break;
//case of capture and break belongs to the py script
//stop
case 5: motors.setSpeeds(0,0);
break;
case 6: if( varSpeed >= 0 && varSpeed < MaxSpeed-50 ) varSpeed+=50;
break;
//decrease speed
case 7: if( varSpeed > 50 && varSpeed <= MaxSpeed ) varSpeed-=50;
break;
case 8: turnRight(MaxSpeed/2);
break;

case 9: turnLeft(MaxSpeed/2);
break;
/*
//step by step move
case 10: Step_forward(varSpeed);
//break;
//decrease speed
case 11: Step_backward(varSpeed);
//break;*/
}
}
```

Code pour Raspberry Pi :

robot.py

```
1 import curses,serial,time, os, subprocess
2 import json, rpc, serial.tools.list_ports, struct, sys
3 import snapshot
4 from datetime import datetime
5 from curses.textpad import Textbox, rectangle
6
7 def portUsb():
8     print("\nAvailable Ports:\n")
9     for port, desc, hwid in serial.tools.list_ports.comports():
10         print("{} : {} [{}]".format(port, desc, hwid))
11
12 #see details about the USB interface by running lsusb -v -d 1ffb
```

```
14 portUsb()
15 arduino = serial.Serial(port=input(), baudrate=115200, timeout=1)
16 input = curses.initscr()
17 curses.noecho()
18 curses.cbreak()
19 input.keypad(True)
20 #set remot text
21 input.addstr(0, 0, "Welcome to the remote control over our zumo")
22 input.addstr(1, 2, "Moving forward : KEY_UP or'i' key")
23 input.addstr(2, 2, "Moving backward : KEY_DOWN or 'k' key")
24 input.addstr(3, 2, "A left turn : KEY_LEFT or 'j'key")
25 input.addstr(4, 2, "A right turn : KEY_RIGHT or 'l'key")
```

```
26 input.addstr(5, 2, "stop : 'a' key ")
27 input.addstr(6, 2, "Capture an image : 'w' key")
28 input.addstr(7, 2, "increase speed by 50 : 'z' key")
29 input.addstr(8, 2, "decrease speed by 50: 'e' key")
30 input.addstr(9, 2, "turn right until stopped: 's' key")
31 input.addstr(10, 2, "turn left until stopped : 'd' key")
32 #input.addstr(11, 2, " ' key")
33 #input.addstr(12, 2, ": 'd' key")
34 input.addstr(11, 2, "quit : 'q' key")
35 input.addstr(12, 2, "the chosen option :")
```

```
37 while True:
38     c = input.getch()
39     if c == curses.KEY_UP or c == ord('i'):
40         arduino.write(bytes("1", 'utf-8'))
41         input.addstr(15, 0, "UP")
42     elif c == curses.KEY_DOWN or c == ord('k'):
43         arduino.write(bytes("2", 'utf-8'))
44         input.addstr(15, 0, "DOWN")
45     elif c == curses.KEY_LEFT or c == ord('j'):
46         arduino.write(bytes("3", 'utf-8'))
47         input.addstr(15, 0, "LEFT")
48     elif c == curses.KEY_RIGHT or c == ord('l'):
49         input.addstr(15, 0, "RIGHT")
```

```
50     arduino.write(bytes("4", 'utf-8'))
51 elif c == ord('s'):
52     input.addstr(15, 0, "Turn right")
53     arduino.write(bytes("8", 'utf-8'))
54 elif c == ord('d'):
55     input.addstr(15, 0, "turn left")
56     arduino.write(bytes("9", 'utf-8'))
57 #capture part
58 elif c == ord('w'):
59     #while(True):
60     input.addstr(15, 0,"choise one of the available ports th
61     input.addstr(16,0,"Available Ports:")
```

```
62 for port, desc, hwid in serial.tools.list_ports.comports
63     input.addstr(17,0,"{} : {} [{}]" .format(port, desc, l
64 editwin = curses.newwin(5,30, 18,2)
65 #rectangle(input, 18,2, 1+5+1, 1+30+1)
66 input.refresh()
67 box = Textbox(editwin)
68 # Let the user edit until Ctrl-G is struck.
69 box.edit()
70 # Get resulting contents
71 port_ = box.gather()
72 #portUsb()
73 #curses.echo()
```

```
14     #port_=input.getstd()
15     snapshot.exe_jpeg_snapshot(port_)
16     #subprocess.call("./snapshot.py", shell=True)
17     #exec(open("./snapshot.py").read())
18 #speed part
19 elif c == ord('a'):
20     arduino.write(bytes("5", 'utf-8'))
21     input.addstr(15, 0, "Stop")
22 elif c == ord('z'):
23     arduino.write(bytes("6", 'utf-8'))
24     input.addstr(15, 0, "Seed increased")
25 elif c == ord('e'):
26     arduino.write(bytes("7", 'utf-8'))
```

```
85 elif c == ord('e'):
86     arduino.write(bytes("7", 'utf-8'))
87     input.addstr(15, 6, "speed decreased")
88 elif c == ord('q'):
89     break # Exit the while loop
90 elif c == curses.KEY_HOME:
91     x = y = 0
92
93 #else :
94 #exec(open("./").read())
95
96
```

Conclusion générale et perspective

Grâce à ce projet de fin d'études, nous avons acquis beaucoup de nouvelles connaissances que nous n'avions jamais vues auparavant. Nous avons réussi à maîtriser la manipulation et le couplage des différents composants électriques, à l'intérêt de mettre en place des systèmes embarqués capables d'exécuter des tâches précises.

Pendant le processus, nous avons familiarisé avec la RPI et Arduino. Et nous avons appris des connaissances des protocoles de communication et des moyennes de transmission des données entre composants électriques et systèmes embarqués.

Bien familiarisée avec le dispositif matériel, logicielle et programmation du robot zumo et caméra Open MV, plus des fonctionnalités de traitement d'images, grâce aux exemples de la détection des blobs, la capture des images et le streaming. Ainsi qu'améliore notre compétence en programmation python et Cpp. Tous en développant une conscience sur l'équipage des différents systèmes embarqués.

En terme de perspective, notre application peut être enrichie par l'ajout des quelques fonctionnalités, concernant la détection des objets ou des obstacles à l'aide d'apprentissage automatique et variation d'angle de vue de caméra. pour permettre une conduction automatique du robot, ou par ajouter un moyen d'alimentation plus compatible.

Bibliographie et Webographie

- [1] <https://www.raspberrypi.com/documentation/> , (7 Avril 2022)
- [2] <https://openmv.io/collections/products/products/openmv-cam-h7-r2> , (12 Avril 2022)
- [3] <https://pythonhosted.org/pyserial/pyserial.html#overview>, (23 Avril 2022)
- [4] <https://docs.openmv.io/library/> , (12 Avril 2022)
- [5] <https://github.com/0penMV/0penMV/> , (15 Mai 2022)
- [6] <https://forums.openmv.io/t/connection-between-raspberrypi-and-openmvcam/784>, (8 Mai 2022)
- [7] <https://www.tomshardware.fr/raspbian-une-debian-optimisee-raspberry-pi/>, (23 Avril 2022)
- [8] <https://github.com/openmv/openmv/tree/master/tools/rpc>, (13 Avril 2022)
- [9] <https://www.pololu.com/category/129/zumo-robots-and-accessories>, (10 Juin 2022)
- [10] <https://docs.arduino.cc/>, (10 juin 2022)