الجـــــــــــــــــــــمـهـوريـــــة الجـــــــــــــزائـريـة الديمـــــــقـراطيـة الـشـــــــــعبيـة

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التـــــــــــــــــــــــــــعـليـم العـــــــالي و البـــــــــحث العـــــلـمـــي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جـــــــــــــامعة أبي بكـــر بـلـقـايد – تـلمســـــــــــــان –

Université Aboubakr Belkaïd – Tlemcen –

Faculté de TECHNOLOGIE

MEMOIRE

Présenté pour l'obtention du **diplôme** de **MASTER**

En : Télécommunications

Spécialité : Réseaux Mobiles et Services de Télécommunications

Par : BELKACEM ANES
DIDOUH MOHAMED

Sujet

Commande à distance d'un robot mobile via un smartphone Android

Soutenu publiquement, le 12/06/2017, devant le jury composé de :

M.ZERROUKI Hadj	MCB	Univ. Tlemcen	Président
M. HADJILA Mourad	MCB	Univ. Tlemcen	Directeur de mémoire
M. MERZOUGUI Rachid	MCA	Univ. Tlemcen	Examinateur

Dédicace

Nous dédions ce modeste travail et nos profonde gratitude à :

Nos mères, sources de tendresse et d'amours pour leurs soutiens tout le long de notre vie scolaire.
Nos pères, qui nous ont toujours soutenus et qui ont fait tout possible pour nous aider.
Nos frères et nos sœurs, que nous aimons beaucoup.
Notre grande famille.
Nos cher ami (e) s, et enseignants, administratif, et le personnel de la faculté de technologie de Tlemcen.
Tout qu'on collaboré de prés ou de loin à l'élaboration de ce travail.
Que dieu leur accorde santé et prospérité.

Remerciements

Avant tout remerciement, louange à Dieu.

C'est avec le plus grand honneur que nous avons réservé l'ouverture de notre projet en signe de gratitude et de reconnaissance à l'égard de tous ceux qui nous ont aidés, de près ou de loin, à la réalisation de ce projet fin d'étude.

Nous tenons à adresser nos vifs remerciements à notre encadrant **Mr.HADJILA Mourad** enseignant Maitre de conférence à l'université Abou-bekr-Belkaid de Tlemcen pour sa présence, son encadrement, ses conseils fournis de façon efficace tout au long de la période de réalisation.

Nous voudrions aussi exprimer nos remerciements sincères à tous les professeurs de l'université Abou Bekr Belkaid de Tlemcen qui ont fait beaucoup d'efforts pour nous transmettre leurs connaissances. Vos compétences incontestables ainsi que vos qualités humaines vous valent l'admiration et le respect de tous. Nos sincères remerciements s'adressent aussi pour votre patience et votre encadrement durant toutes ces années.

Nos remerciements s'adressent également aux membres du Jury **Mr. ZERROUKI Hadj** et **Mr.MERZOUGUI Rachid** qui nous font l'honneur de participer à notre soutenance.

Merci également à la société « **Sunfounder** » dont le projet nous a inspiré et dont les tutoriels détaillés nous ont aidé à combler nos lacunes, ainsi que la répondre de toutes nos emails, et je remercie personnellement **Mr.Jake** qui envoie à moi (**Mr.BELKACEM Anes**) un colis le 20 mai 2017 contient une bras acrylique gratuitement parce que l'original a été cassé.

Table des matières

Listes des figures

Listes des tableaux

Listes des abréviations

AMD : Advanced Micro Devices.
A : Ampère.
CC : Courant Continu.
CISC : Complex Instruction Set Computer.
CAN (ou Convertisseur A/N) : Convertisseur Analogique Numerique.
CNA (ou Convertisseur N/A) : Convertisseur Numerique Analogique.
E/S : Entrées/Sorties.
FPGA : Field Programmable Gate Array.
HTTP : HyperText Transfert Protocol.
HDMI : High Definition Multimedia Interface.
IoT : Internet Of Things.
IDE : Integrated Development Environment.
IDLE : Integrated DeveLopment Environment.
IHM : Interface Homme Machine.
IP : Internet Protocole.
GNU : GNU's Not UNIX.
GUI : Graphical User Interface.
GNOME : GNU Network Object Model Environment.
GND : Ground.
GHz : GegaHertz.
Go : Gégaoctet
GPIO : General Purpose Input/Output.
GPS : Global Positionning System.
3G : Troisième Génération.
KDE : K Desktop Environment.
LCD : Liquid Crystal Display.
LED : Light-Emitting Diode.
LXDE : Lightweight X11 Desktop Environment.
MIPS : Microprocessor without Interlocked Pipeline Stages.
MYSQL : My Structured Query Language.
MHz : MegaHertz.
Mbps : Mega bit par seconde.
MIT : Massachusetts Institute of Technology.
Mo : Mégaoctet
mm : millimètre.
mA : milli- Ampère.
NASA : National Aeronautics and Space Administration.
OLED : Organic Light-Emitting Diode.
OE : Output Enable.
PSoC : Programmable System on Chip.
PC : Personal Computer.
PHP : Hypertext Preprocessor.
PWM : Pulse Width Modulation.
PDA : Personal Digital Assistant.
RISC : Reduced Instruction Set Computer.

RCA : Radio Corporation of America.
RAM : Random Access Memory.
RJ45 : Registered jack45.
RFID : Radio Frequency Identification.
SD : Secure Digital.
SoC : System on Chip.
SCP : Systèmes Cyber Physiques.
UART : Universal Asynchronous Receiver Transmitter.
UK : United Kingdom.
UTF-8 : Universal Character Set Transformation Format - 8 bits.
USB : Universal Serial Bus**.**GSM : Global System for Mobile.
V : Volts.
WiFi : Wireless Fidelity.

Introduction générale

L'omniprésence de l'informatique dans notre quotidien, ainsi que l'extension de l'Internet à toutes formes d'objet du quotidien (connue sous le nom d'Internet des Objets, en anglais Internet of Things, abrégé IoT), ont favorisé le développement et le déploiement d'une nouvelle génération d'objets interconnectés dotés de capacités avancées de capture (pour sonder l'environnement physique), de traitement (pour effectuer des calculs) et de communication (pour échanger des données et être contrôlés à distance).

Chaque année, le Consumer Electronics Show (CES) présente les innovations et les objets connectés de demain, dont le nombre dépasserait les 50 milliards d'ici 2020, soit plus de six objets connectés par personne (estimation Cisco). Ces objets connectés peuvent être de simples capteurs passifs (qui constituent la grande majorité), des caméras IP/WiFi, ou encore des systèmes embarqués complexes comme une voiture sans conducteur.

Aujourd'hui, des systèmes intelligents se retrouvent dans un nombre croissant d'environnements, tels que des maisons intelligentes (smart homes), des transports intelligents (smart transport), des hôpitaux intelligents (smart hospitals), ou encore des villes intelligentes (smart cities). De plus en plus on a besoin de rendre les applications accessibles sur le web. Les motivations sont multiples, vendre des services en ligne, faire communiquer des applications...ect, cette situation va conduire à une révolution en termes de création et de disponibilité de services et d'applications à forte valeur ajoutée (5 milliards d'applications IoT seront développées d'ici 2020), et va profondément changer (encore) notre façon de vivre et d'agir sur notre environnement. En bref, le monde physique fusionne progressivement avec le monde virtuel.

Dans ce cadre, aussi la robotique est omniprésente dans notre quotidien, surtout quand il s'agit de réaliser des tâches dangereuses, automatisables, ou demandant une grande précision. Ce qui est certain c'est la complexité croissante de ces systèmes embarquant de plus en plus de technologies. Dans ces systèmes robotisés autonomes ou semi-autonomes les capteurs occupent une place prédominante, c'est ce qui permet aux systèmes d'appréhender leur environnement, l'analyse quant à elle est rendu possible grâce aux microprocesseurs qui vont traité et réagir suivant les différents signaux qui lui sont rapportés grâce à divers protocoles de communication. Différents actionneurs permettent d'interagir avec l'environnement (servomoteurs, moteurs...). La problématique est la suivante : comment interagir intelligemment tout ces éléments et comment interconnecté le monde physique (électronique) avec le monde virtuel (programmation) d'une façon optimisé ?

D'abord, un robot est un dispositif mécatronique (alliant mécanique, électronique et informatique), il est utilisée dans plusieurs industries comme l'automobile, la médecine, l'électroménager et plusieurs autres. Le plus complexe des machines peuvent être assemblées à l'aide de la robotique. Les robots peuvent être fixes (bras manipulateur) ou mobiles (robot marcheur, robot à roues) selon l'application, les robots fixes sont généralement utilisés dans les usines, les hôpitaux, domaine agricole...etc. Cependant les robots mobiles sont généralement utilisés dans les environnements dangereux : nucléaires, militaires, déminage,...etc. Ce dernier type de robot est notre domaine d'intérêt dans ce mémoire.

Notre mémoire est organisé en trois chapitres comme suit :
Le premier chapitre présente des généralités sur les systèmes embarqués temps réel, l'Internet des Objets, les systèmes cyberphysiques, Big Data et les différentes applications pour chaque technologie.

Le deuxième chapitre est devisé en trois parties :
- La première concerne une présentation sur la carte Raspberry Pi, l'historique, ses caractéristiques, ses composants, et les étapes de configuration pour doit être prêt à utiliser.
- La deuxième traite le système d'exploitation Linux Raspbian qui est recommandé par la fondation Raspberry Pi.
- La troisième partie était de comprendre l'ensemble des ports d'entrées et de sorties de la Raspberry Pi appelés GPIO, et comment le contrôler par la programmation en utilisant la langage Python.

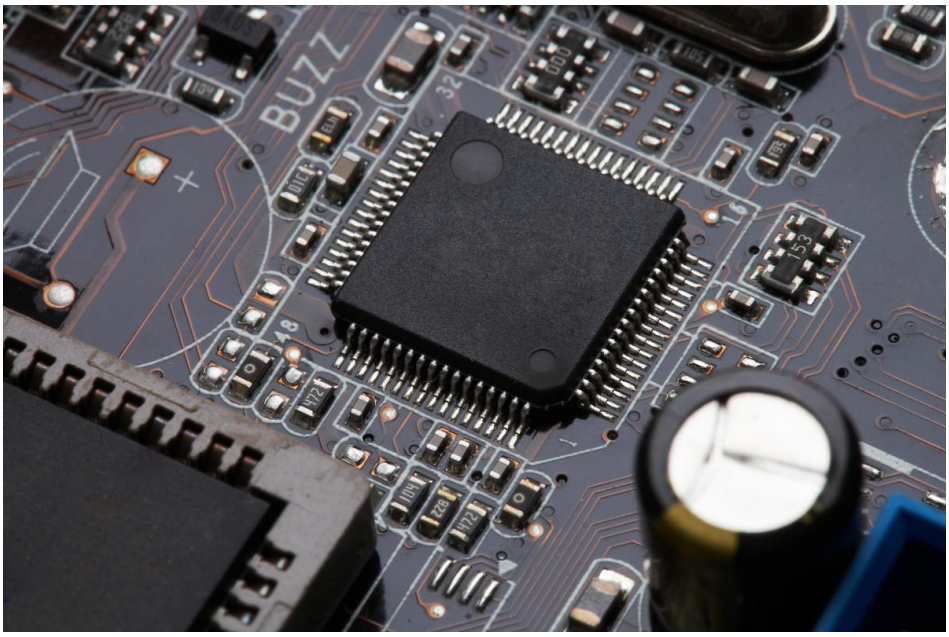
Finalement, le dernier chapitre décrit la réalisation pratique a pour but d'apprendre à utiliser et programmer notre Raspberry Pi dans le cadre de la robotique. Ce chapitre contient :
Une partie de description concernant les composantes de notre robot mobile.

La partie électronique, dans cette partie nous allons décrire l'ensemble des modules électroniques utilisés, et nous allons voir en détails la réalisation électronique de notre robot.

La partie informatique, ceci est consacrée aù contrôler le robot via une application desktop créé sous Python et aussi avec le web là où on explique les différentes procédures de développement de notre site web avec Django pour commander ce robot à travers une application Android qui constitue l'objectif de notre projet de fin d'étude et par la suite avec PHP. De plus, on a placer une webcam au robot pour nous suivre ce que est en train de faire en temps réel, et la possibilité de prendre des images sur l'environnement que le robot à explorer. Le contrôle à distance du robot est basé sur le réseau local de type WiFi en utilisant le module WiFi adaptateur. Enfin, nous finissons notre mémoire par une conclusion qui présente le bilan de ce travail et les perspectives envisagées.

Chapitre I

Les systèmes embarqués



I.1 Introduction

Les systèmes embarqués sont des systèmes électroniques et informatiques autonomes, soumis des fortes contraintes énergétiques et temps réel.

Ce chapitre présente le lecteur au monde des systèmes embarqués qui ont envahi notre vie quotidienne. Donc, nous allons comprendre dans ce chapitre les bases du système embarqué temps réel à partir de son apparition jusqu'à les technologies de nouvelle génération comme l'Internet des Objets, les systèmes cyberphysiques et Big Data.

I.2 Générations et l'histoire des systèmes embarqués

I.2.1 Première génération : 1945 - 1955

La technologie utilisée repose sur celle des tubes à vide (tube et tore de ferrite). C'est une technologie lourde, ce qui a posé des problèmes de place et de consommation électrique.

Figure I-1. Tubes à vide.

I.2.2 Deuxième génération : 1955 - 1965

Elle coïncide avec l'apparition des transistors dans les unités centrales. L'utilisation de cette technologie améliore les vitesses de traitement et de débits des informations dans les ordinateurs.
De plus, l'intégration de ces transistors permit la miniaturisation des différents composants d'un ordinateur. Ceci se confirme au travers de la loi empirique de Gordon Moore, cofondateur d'Intel, qui stipule que pour une surface de silicium donnée, on double le nombre de transistors intégrés tous les 18 mois.

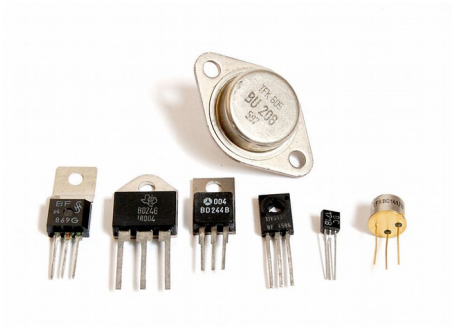


Figure I-2. Différents types de transistors.

Dans cette génération la programmation des systèmes embarqués directe sur le système en assembleur parrait difficile, lente et peu réutilisable.

I.2.3 Troixième génération : 1965 - 1970

Cette génération est caractérisée par l'apparition des circuits intégrés. En 1967, apparut Apollo Guidance Computer, premier système embarqué qui comporte environ un millier de circuits intégrés identiques (portes NAND) [1].

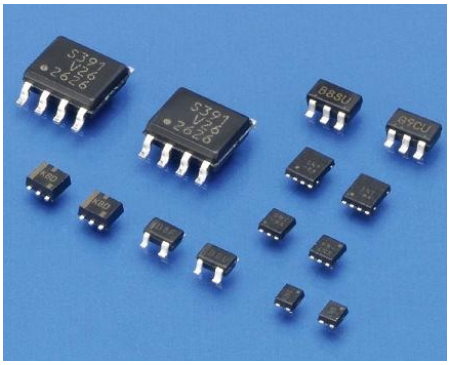


Figure I-3. Circuits intégrés.

I.2.4 Quatrième génération : 1970 - 2000

L'omniprésence des systèmes embarqués dans notre vie est liée à la révolution numérique opérée dans les années 1970 avec l'avènement des processeurs. Les processeurs, de plus en plus rapides, puissants et bon marché ont permis cette révolution et aussi le boom du marché de l'embarqué.

En 1971, Intel a introduit la première puce microprocesseur du monde appelée la 4004, premier circuit générique, personnalisable par logiciel. Elle a été conçue pour être utilisée dans les calculatrices commerciales. Elle a été produite par la société japonaise Busicom.

L'apparition des microprocesseurs a donné naissance à la micro-informatique et à la personnalisation des ordinateurs.

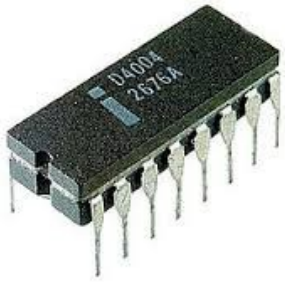


Figure I-4. Microprocesseur 4004.

I.2.5 Cinquième génération : 2000 - 2010

Programmable **S**ystem **o**n **C**hip (PSoC) est une famille de circuits intégrés introduits au début des années 2000 par Cypress.

Figure I-5. Schéma de PSoC programmé via un environnement de développement.

C'est un circuit intégré qui comprend un microcontrôleur et des fonctions logiques et analogiques configurables et interconnectables entre eux.

L'idée est de remplacer le microcontrôleur et les circuits d'interfaces analogiques (CAN et CNA, filtres, amplificateurs opérationnels, etc.) ou numériques (compteurs, timers, UART, interfaces pour divers bus, etc.) associés par un circuit unique. On intègre ainsi un système électronique embarqué complet dans un circuit intégré unique, ou tout au moins, on réduit très considérablement le nombre de composants.

Figure I-6. Deux maquettes avant et après le PSoC.

Le PSoC est un circuit reconfigurable en fonctionnement, on peut par exemple imaginer un système embarqué qui s'arrête de fonctionner 1 à 2 minutes par jour pour se reconfigurer en modem et envoyer toutes les données qu'il a acquis dans la journée. Les fonctions utilisées pour la mesure sont reconfigurées en modem.

En 2000 a été créé le consortium Linux embarqué (Embedded Linux Consortium) dont le but est de centraliser et de promouvoir les développements de solutions Linux embarqué. Parallèlement s'est mis en place le portail « linuxdevices.com » dédié à Linux pour l'embarqué. Tout cela a contribué à la percée de Linux dans le monde de l'embarqué [2].

I.2.6 Sexième génération : 2010 - L'avenir

I.2.6.1 Internet des Objets (Internet Of Things)

Les objets communicants deviennent de plus en plus « connectés » au web. Ils sont de plus en plus autonomes, programmables, ils tendent à devenir des systèmes embarqués et connectés.

Figure I-7. Objets communicants via le monde virtual Internet.

I.2.6.2 Systèmes Cyber Physiques (SCP)

Les systèmes cyberphysiques, nouvelle génération de systèmes embarqués dans lesquels les éléments informatiques interagissent avec les processus physiques, prennent une place croissante dans la société digitalisée d'aujourd'hui, comme le secteur de l'énergie avec les « smart-grids », celui du transport avec les véhicules autonomes ou la gestion du trafic routier, l'organisation des villes avec les « smart cities », la défense avec l'utilisation de drones...ect.

La complexité de ces systèmes qui conjuguent rigueur mathématique et observations empiriques représente un véritable défi pour l'avenir. Par ailleurs, ils doivent encore être acceptés par les utilisateurs et bénéficier de leur confiance pour être utilisés [3].

I.3 Architecture générale et modes de fonctionnement des systèmes embarqués

Les systèmes embarqués utilisent généralement des microprocesseurs à basse consommation d'énergie ou des microcontrôleurs, dont la partie logicielle est en partie ou entièrement dans le matériel (on parle alors de firmware).

Par définition, un système embarqué est un système complexe qui intègre du logiciel et du matériel conçus ensemble afin de fournir des fonctionnalités données. Il contient généralement un ou plusieurs microprocesseurs destinés à exécuter un ensemble de programmes définis lors de la conception et stockés dans des mémoires.

D'autre part, c'est un système électronique et informatique autonome, souvent temps réel, spécialisé dans une tâche bien précise. Ses ressources sont limitées.

Pour optimiser les performances et la fiabilité de ces systèmes, des circuits numériques programmables FPGA (Field Programmable Gate Array), des circuits dédiés à des applications spécifiques ASIC (Application Specific Circuits) ou des modules analogiques sont en plus utilisés.

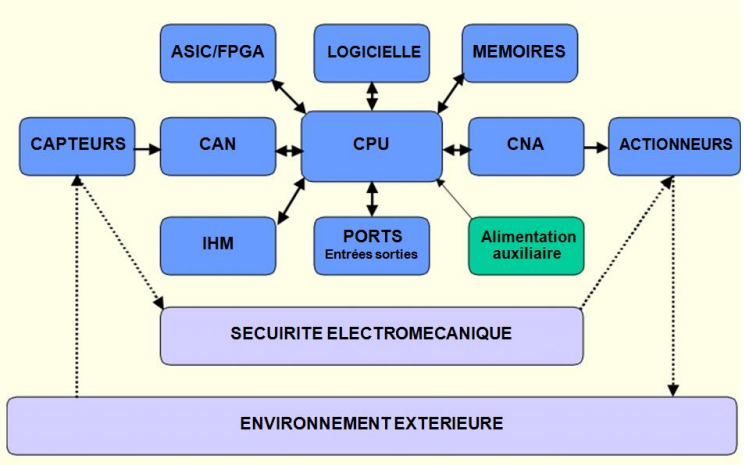


Figure I-8. Système embarqué typique.

D'après la figure I-8 qui présente le système embarqué typique, On retrouve en entrée des **capteurs** généralement analogiques couplés à des convertisseurs A/N. On retrouve en sortie des **actionneurs** généralement analogiques couplés à des convertisseurs N/A. Au milieu, on trouve le calculateur mettant en œuvre un **processeur** embarqué et ses périphériques d'E/S. Il est à noter qu'il est complété généralement d'un circuit FPGA jouant le rôle de coprocesseur afin de proposer de accélérations matérielles au processeur.

Sur ce schéma théorique se greffe un paramètre important c'est le rôle de **l'environnement extérieur**, l'environnement dans lequel opère le système embarqué n'est pas contrôlé ou contrôlable. Cela suppose donc de prendre en compte ce paramètre lors de conception. On doit par exemple prendre en compte les évolutions des caractéristiques électriques des composants en fonction de la température, des radiations…etc [4].

I.3.1 Caractéristiques principales d'un système embarqué

Les principales caractéristiques d'un système embarqué sont les suivantes :

- C'est un système principalement numérique.
- Il met en œuvre généralement un processeur.
- Dédié pour une application spécifique.
- Coût réduit.
- Espace restreint (volume, capacité mémoire).
- Capacité de calcul appropriée et adaptée.
- Exécution temps réel.
- Fiabilité et sécurité de fonctionnement.
- Consommation d'énergie maitrisée, voir très faible en cas d'utilisation sur batterie.

I.3.2 Domaines d'application d'un système embarqué

Les grands secteurs de l'embarqué concernent les domaines suivants :

- Contrôle de systèmes : automobile, process chimique, process nucléaire, système de navigation…
- Traitement du signal : radar, sonar, compression vidéo.

• Communication et réseaux : transmission d'information et commutation, téléphonie, Internet, serveur de temps, pare-feu.
• Électroménager : télévision, four à micro-ondes, machine à laver …etc.
• Informatique : disque dur, lecteur de disquette, etc.
• Équipement médical.

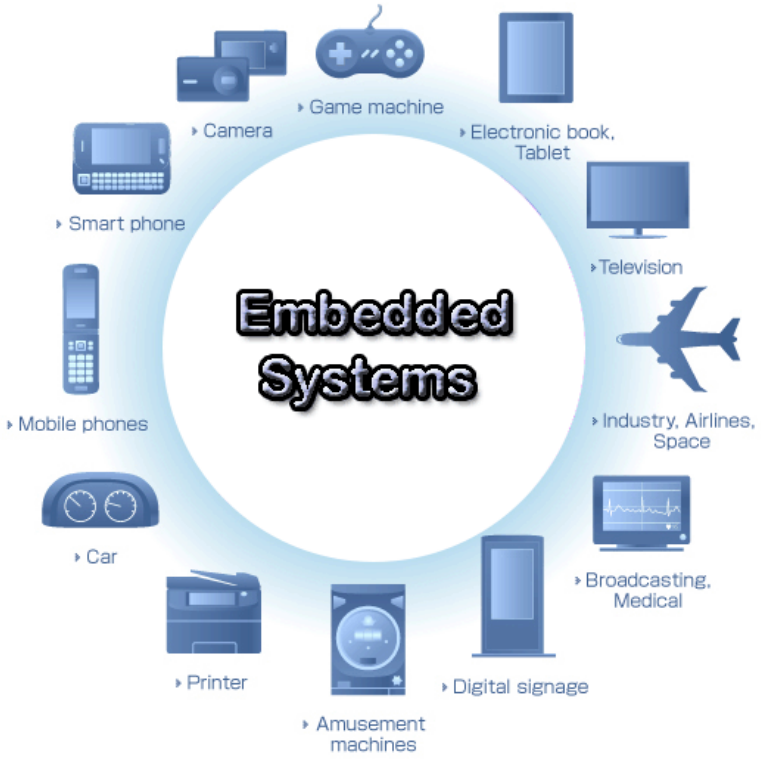


Figure I-9. Les domaines dans lesquels on trouve des systèmes embarqués.

I.4 Systèmes embarqués temps réel

La plupart des systèmes embarqués temps réel sont avant tout des systèmes de **contrôle commande**.

Un système de contrôle-commande est un système informatique de contrôle de procédé. Le terme procédé est un terme générique désignant un système physique contrôlé.

Afin de contrôler le procédé, le système informatique est en relation avec l'environnement physique externe par l'intermédiaire de capteurs et/ou d'actionneurs. Les grandeurs physiques acquises grâce aux capteurs permettent au système de contrôle-commande de s'informer de l'état du procédé ou de son environnement. Le système de contrôle-commande, à partir de consignes décrivant l'état voulu du procédé, calcule des commandes qui sont alors appliquées sur le procédé par l'intermédiaire d'actionneurs. Donnons ainsi une définition générale d'un système de contrôle commande (figure I-10) :

« Un système de contrôle-commande reçoit des informations sur l'état du procédé externe, traite ces données et, en fonction du résultat, évalue une décision qui agit sur cet environnement extérieur afin d'assurer un **état voulu** ».

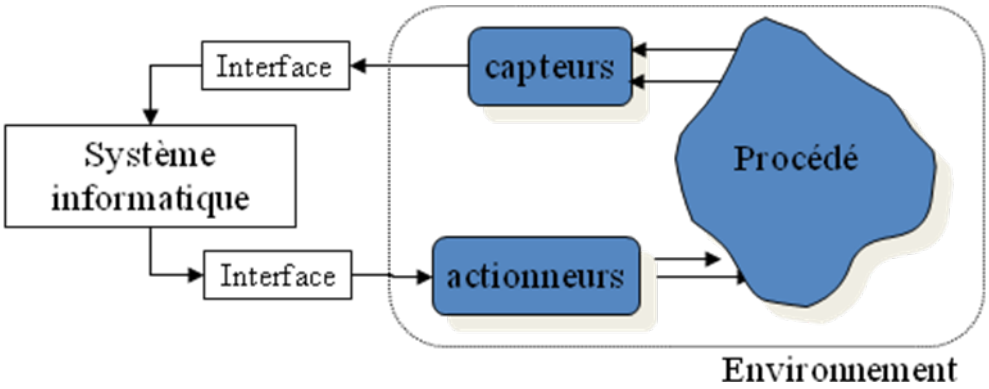


Figure I-10. Représentation schématique d'un système de contrôle commande.

Exemple

La figure I-11 représente un exemple de système de contrôle-commande pour un drone. Un drone est un aéronef autonome, ou semi-autonome. L'un des buts principaux du système de contrôle-commande est de contrôler l'attitude (tangage, roulis) et la vitesse du drone : cela s'appelle la commande de vol. Le système de contrôle-commande est exécuté par un calculateur qu'on appelle **microcontrôleur**.

Les capteurs utilisés sont une centrale inertielle, permettant de connaître les angles de tangage et roulis du drone, et un récepteur GPS (Global Positionning System) permettant notamment de connaître la position absolue (latitude, longitude), la vitesse par rapport au repère sol et la vitesse ascensionnelle.

Un modem sans fil permet au drone de recevoir des consignes de la part d'un opérateur (par exemple un angle de roulis, une vitesse et une vitesse ascensionnelle), et de lui envoyer des informations sur le vol. À la fois capteur (lit des informations en provenance de l'opérateur) et actionneur (permet l'affichage ou l'enregistrement d'informations pour l'opérateur), ce modem est qualifié d'organe de dialogue. Enfin, trois actionneurs permettent de commander respectivement la puissance du moteur et la position des deux ailevons. Le différentiel des ailevons (l'un plus haut, l'autre plus bas), permet de donner une vitesse angulaire sur l'axe de roulis de l'aéronef, alors que leur position parallèle par rapport à l'angle d'équilibre permet d'appliquer une vitesse angulaire sur l'axe de tangage. La vitesse ascensionnelle dépend de la puissance du moteur et des angles de tangage et roulis.

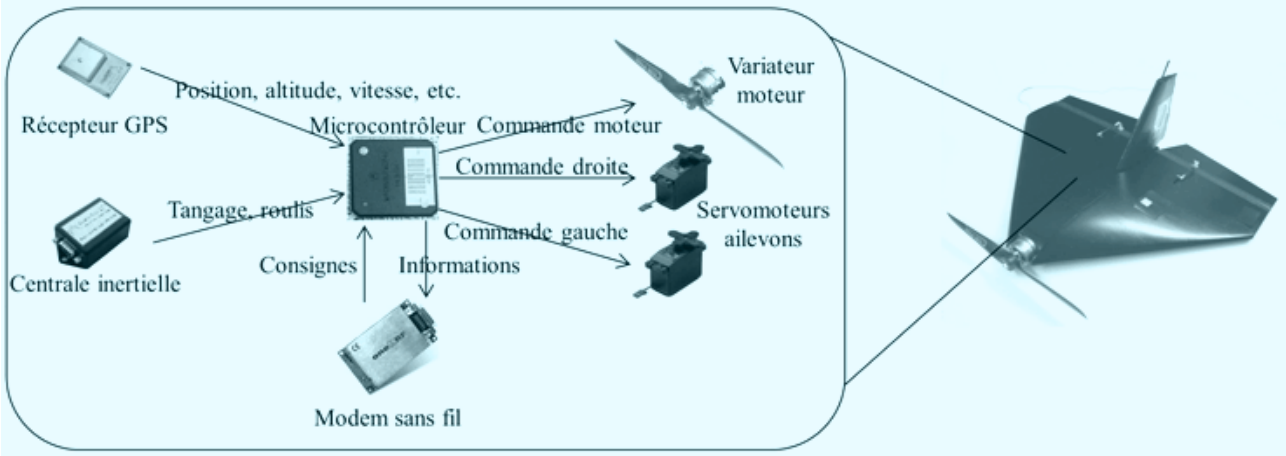


Figure I-11. Système de contrôle-commande embarqué sur un mini-drone.

Un système temps réel est un système informatique soumis à des contraintes de temps. Une définition du temps réel peut être :

Un système temps réel est un système dans lequel l'exactitude des applications ne dépend pas seulement de l'exactitude du résultat mais aussi du temps auquel ce résultat est produit.

Une autre définition du temps réel peut être :

« Un système est dit temps réel s'il doit s'exécuter suffisamment vite par rapport à la dynamique du procédé contrôlé. La notion relative de vitesse s'exprime par des contraintes temporelles. »

La notion « suffisamment vite » est relative à la dynamique du procédé contrôlé. Par exemple, le drone de la figure I-11, est considéré comme un système temps réel. De par sa forme d'aile delta, il est très instable en roulis; par conséquent, il est important que la commande de vol se rafraîchisse fréquemment, par exemple toutes les 20 millisecondes. Si le drone avait une forme de planeur, plus stable, une fréquence de rafraîchissement plus faible (de l'ordre de 10 Hz, soit 100 millisecondes) pourrait peut être suffire à assurer une commande de vol acceptable.

Exemples de grandeur des contraintes temporelles :

- La **milliseconde** pour les systèmes de radar.

- La **seconde** pour les systèmes de visualisation humaine.

- Quelques **heures** pour le contrôle de production impliquant des réactions chimiques.

- **24 heures** pour les prévisions météo.

- Plusieurs **mois** ou **années** pour les systèmes de navigation de sonde spatiale.

I.4.1 Les contraintes de temps et les systèmes embarqués

Un système **critique** est un système informatique pour lequel une défaillance, totale ou partielle, du système, peut entraîner des conséquences graves, par exemple dans les domaines économique, humain, écologique, etc. La défaillance d'un drone d'observation entraîne des conséquences

économiques (perte de l'engin) mais surtout peut entraîner des conséquences humaines en cas de collision, moteur allumé, avec une personne. Un drone est donc considéré comme un système critique lorsqu'il évolue dans une zone habitée. Un système est temps réel critique si une défaillance temporelle (le système ne réagit pas suffisamment vite par rapport à la dynamique du procédé) peut aussi entraîner des conséquences graves. Le drone peut être considéré comme temps réel critique, puisque le fait de ne pas être capable d'appliquer les commandes de vol à temps peut aboutir à une perte de contrôle. La plupart des systèmes de transport sont des systèmes embarqués de contrôle-commande, temps réel critique.
Si les contraintes temporelles de l'application ne sont pas respectées, on parle de défaillance du système. Il est donc essentiel de pouvoir garantir le respect des contraintes temporelles du système. Ceci nécessite que le système permette un taux d'utilisation élevé, tout en respectant les contraintes temporelles identifiées.

Voila des définitions qualifiant des systèmes temps réel ayant des spécifications particulières. Ces contraintes temporelles peuvent être de plusieurs types :

I.4.1.1 Contraintes temporelles strictes ou dures ou critiques (hard real-time)

Le non-respect des échéances peut avoir des conséquences graves sur le fonctionnement du système ou sur son environnement (auto-pilotage, freinage, assistance médicalisée…). Les échéances ne doivent jamais être dépassées.
Donc, le non respect de contraintes temporelles entraîne la faute du système.

Les fautes temporelles ne sont pas tolérables.

Exemple critique : le contrôleur de frein d'une voiture.

I.4.1.2 Contraintes temporelles relatives ou lâches (temps réel souple : soft real-time)

Le respect des échéances est important mais le non respect des échéances ne peut occasionner de graves conséquences.

Les fautes temporelles sont tolérables.

Exemples : projection vidéo (décalage entre le son et l'image).

I.4.1.3 Contraintes temporelles fermes (temps réel ferme : firm real-time)

Temps réel souple avec le manquement occasionnel des échéances. les fautes temporelles sont autorisées dans une certaine limite, par exemple une erreur toutes les trois exécutions au plus.
Exemple : projection vidéo (perte de quelques trames d'images).

I.4.2 Caractéristiques temporelles des systèmes temps réel

Les contraintes temporelles qui sont classiquement présentées sont des **contraintes de bout en bout**, appelées aussi **contraintes de latence**. Ces contraintes représentent le délai maximal entre lecture de l'état du système (lecture des capteurs) et commande de celui-ci (commande des actionneurs).
Par exemple, la commande de vol du drone présenté dans l'exemple, puisqu'elle doit s'exécuter à 50 Hz, c'est-à-dire avec une période de 20 millisecondes, se verra vraisemblablement munie de

contraintes temporelles de bout en bout sur la commande de vol, par exemple de 20 millisecondes, afin d'obliger le système à s'exécuter une fois par période.

Le respect du protocole de communication avec les capteurs, actionneurs, ou bus de communication est une autre source, très importante, de contraintes temporelles. Ainsi, à chaque fois qu'une trame est disponible sur un réseau, le système doit la lire et soit la traiter, soit la stocker pour traitement ultérieur, sous peine de la voir être remplacée (ou « écrasée ») par la trame suivante [5].

I.4.3 Principales caractéristiques des systèmes embarqués temps réel

• **Taille et complexité**
- Un système temps réel interagit avec un environnement extérieur souvent complexe et en évolution.
- Il doit respecter des échéances temporelles, garantir une fiabilité permanente.
- Il doit pouvoir interagir avec différents types d'éléments matériels.
• **Implémentation efficace** : restrictions dans l'utilisation des constructions du langage.
• **Certification :** garantir un fonctionnement conforme aux spécifications.
• **Simulation, prototypage** : vérification a priori de la conception.
• **Plate-formes d'essai** : vérification a posteriori du bon fonctionnement.

I.4.4 Exemples d'applications des systèmes embarqués temps réel

Le temps réel touche des domaines très variés tels que :
- Le pilotage ou la navigation (de bateaux, d'avions, de trains, d'automobiles).
- Le domaine des transactions bancaires.
- Transports (métro, aérospatiale, automobiles, etc.).
- Médias (décodeurs numériques).
- Services téléphoniques (terminal GSM, autocommutateur).
- Supervision médicale.
- Le contrôle de processus industriels et systèmes de production industrielle :
centrale nucléaire, chaîne de montage, usine chimique, etc.
- Robotique (ex. PathFinder : sonde lancée par la NASA en mars 1996, composée d'une station au sol et d'un robot mobile Sojourner).

I.5 Les logiciels libres et les systèmes embarqués

Les logiciels libres et en particulier GNU/Linux sont de plus en plus employés dans l'embarqué depuis qu'ils ont commencé à faire leur preuve il y a quelques années.
Regardons Linux. Pourquoi retrouve-t-on Linux dans l'embarqué ? Tout d'abord pour ses qualités qu'on lui reconnaît maintenant dans l'environnement grand public :
• C'est un logiciel libre disponible gratuitement au niveau source.
• Il est stable et efficace.
• Il existe une aide rapide en cas de problèmes par la communauté Internet des développeurs Linux.
• Il y a un nombre de plus en plus important de logiciels libres disponibles.
• La connectivité IP chère aux systèmes embarqués est en standard.
Linux possède d'autres atouts très importants pour l'embarqué :

• Il existe un portage pour processeurs autres que la famille x86 : PowerPC, ARM (par exemple Raspberry Pi) , MIPS, 68K, ColdFire…
• La taille du noyau est modeste et compatible avec les tailles de mémoires utilisées dans un système embarqué (800 ko pour µClinux sur processeur ColdFire).
• Différentes distributions spécialisées existent pour coller à un type d'application : routeur IP, PDA, téléphone…etc.
• Le chargement dynamique de modules (drivers) est autorisé, ce qui permet d'optimiser la taille du noyau.
• La migration pour un spécialiste Linux à Linux embarqué est rapide et en douceur, ce qui réduit les temps de formation et les coûts.
Qui dit systèmes embarqués dit souvent temps réel. Linux pour l'embarqué supporte aussi différentes extensions temps réel [6].

I.6 Les systèmes cyberphysiques et l'Internet des objets dans les systèmes embarqués

I.6.1 Les systèmes cyberphysiques

La première définition que l'on peut trouver de systèmes cyber-physiques (CPS) date de 2006, lors de travaux avec la National Science Foundation (NSF) américaine. Les CPS intègrent des process physiques et computationnels. Des ordinateurs et réseaux embarqués surveillent et contrôlent les process physiques, généralement avec des boucles de rétroaction où les process physiques affectent les calculs et vice versa. En d'autres mots, les CPS utilisent des computations et de la communication profondément intégrée et interagissant avec les process physiques afin de produire de nouvelles capacités du système. Un CPS peut être considéré aussi bien à une petite échelle (e.g. pace maker) qu'à de grandes échelles (un réseau national de distribution d'énergie).

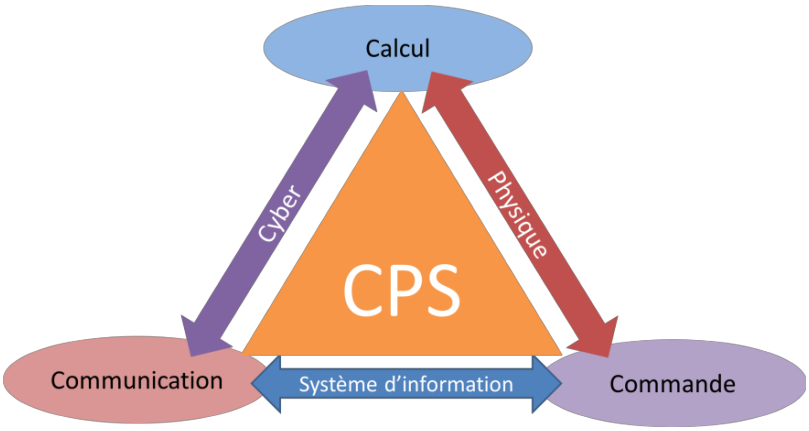


Figure I-12. Capacités de base d'un CPS.

I.6.1.1 Caractéristiques d'un CPS

• Haut niveau d'intégration physique/cyber.
• Capacités de traitement dans chaque composant physique, dû au fait que les ressources en traitement et communication sont généralement limitées.
• Hautement connectés, via réseaux avec ou sans fil, Bluetooth, GSM, GPS etc.

• Adapté à des échelles temporelles et spatiales multiples.
• Capable de reconfiguration/réorganisation dynamique.
• Hautement automatisés, en boucles fermées.
• Fiables, voire certifiés dans certains cas.

I.6.1.2 Classification des CPS

• **C1**. Au niveau **Connexion**, le CPS opère sur un réseau Plug & Play et utilise des données envoyées par un réseau de capteurs.
• **C2**. Au niveau **Conversion**, le CPS sait traiter l'information et la retranscrire en informations de plus haut niveau.
• **C3**. Au niveau **Cyber**, le CPS a une connaissance des autres CPS de l'environnement et peut interagir avec eux pour enrichir son propre traitement d'information.
• **C4.** Au niveau **Cognition**, le CPS est capable d'établir un diagnostic basé sur des simulations de son propre comportement et une analyse différentielle des données de capteurs.
• **C5.** Au niveau **Configuration**, le CPS peut s'adapter seul en cas de défaillance, se reconfigurer ou ajuster de manière autonome ses paramètres afin de retourner à un comportement nominal.

I.6.1.3 Exemples d'application

Dans ce quit suit, on présente des exemples d'application des systèmes cyberphysiques.
• Santé
La figure montre un exemple d'application des systèmes cyberphysiques dans le domaine de santé.

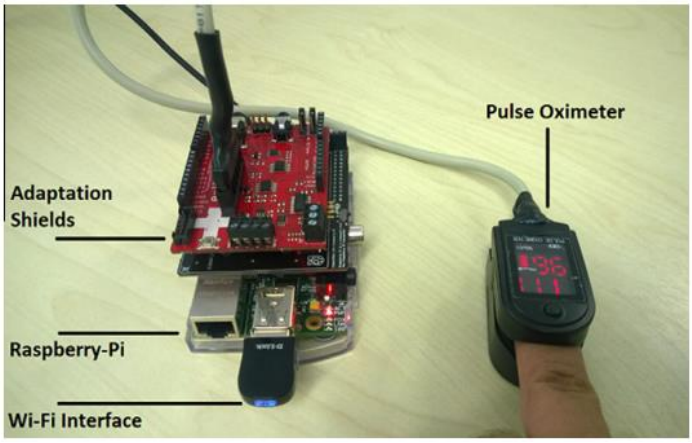


Figure I-13. Les systèmes cyberphysiques dans le domaine de santé.

Les systèmes cyberphysiques, avec les différent capteurs embarqués, forment aujourd'hui une source d'information en temps réel d'une valeur inestimable pour l'innovation dans ce secteur [7].

• Véhicules autonomes

Un véhicule autonome est capable de se conduire lui-même sans intervention humaine. véhicule autonome comprend trois grandes fonctions (voir la figure I-14) : la récupération d'informations sur l'environnement (aussi appelée «perception»), la décision d'une trajectoire («planning») et enfin la réalisation de la trajectoire (l'«action»). En anglais, l'on parle de «sense, understand/decide, act».

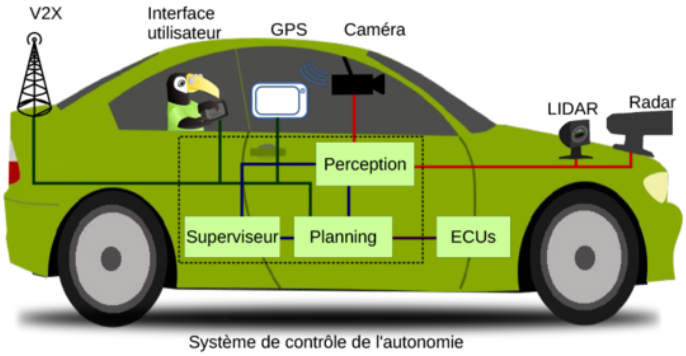


Figure I-14. Architecture d'un véhicule autonome.

En cas d'urgence, un conducteur doit pouvoir reprendre la main à tout moment : la voiture est alors une voiture classique. Mais, par le fait qu'elle comporte un mode «autonome», cela veut dire que les aspects conduites de la voiture sont intégralement robotisés. Dit différemment, l'ensemble des fonctions de la conduite (accélération, freinage, direction) est accessible de façon électronique/informatique [8].

Véhicules autonomes robuste. Il peut être contrôlé via un logiciel autonome, via un contrôleur GPS

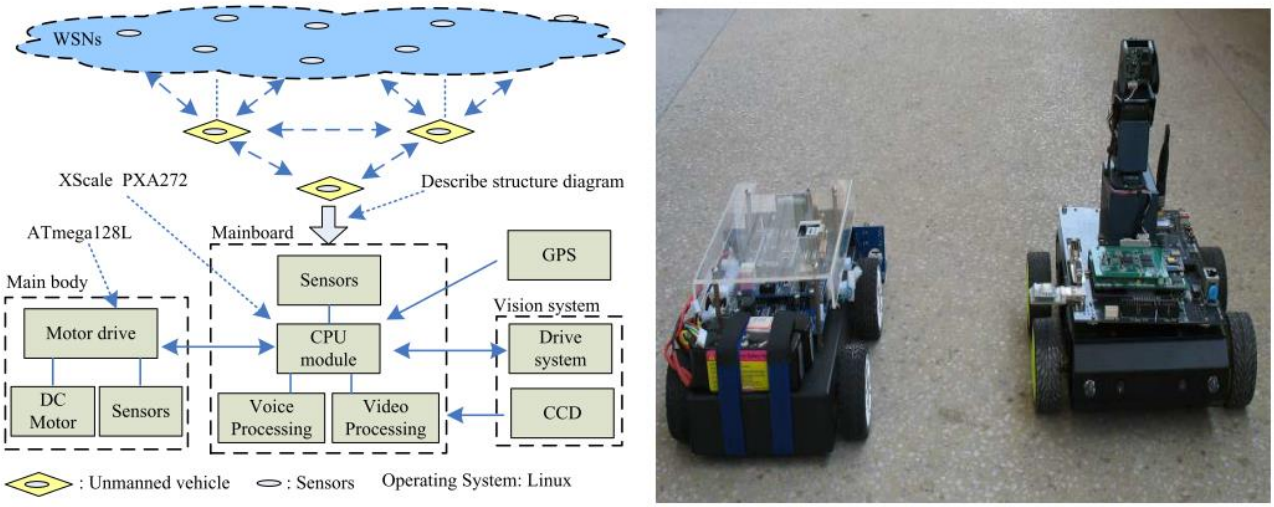


Figure I-15. Véhicules autonomes contrôlé via un logiciel.

• Maintenance industrielle

Le produit à fabriquer possède toutes les informations nécessaires pour chaque étape de la production, c'est étapes sont configurés de façon flexible afin de répondre aux changements de situation.

Figure I-16. Production adaptative avec les systèmes cyberphysiques.

• Transports intelligents

Nos voitures embarquant de nombreux équipements électroniques seront bientôt communicantes, et pourront dialoguer entre elles, ainsi qu'avec les infrastructures de la route.

Cet échange permettra d'améliorer la sécurité routière en offrant des informations pertinentes en temps réel aux conducteurs, pour éviter des accidents ou des embouteillages.

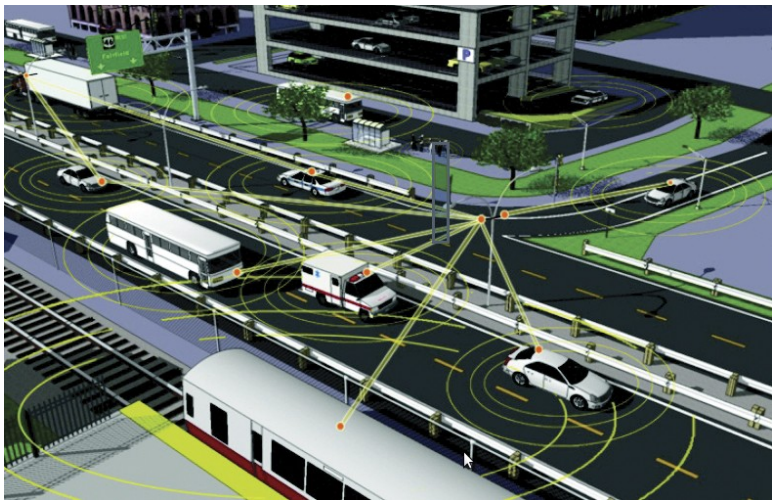


Figure 1-17. Transports intelligents.

I.6.2 Internet des Objets

Les systèmes embarqués sont aujourd'hui fortement communicants, cela est possible grâce aux puissances de calcul offertes par les processeurs pour l'embarqué et grâce aussi à l'explosion de l'usage la connectivité Internet ou connectivité IP. La connectivité IP permet fondamentalement de contrôler à distance un système embarqué par Internet.

Cela permet l'emploi des technologies modernes du web pour ce contrôle à distance par l'utilisateur, il suffit d'embarquer un serveur web dans son équipement électronique pour pouvoir le

contrôler ensuite à distance, de n'importe où, à l'aide d'un simple navigateur. Il n'y a plus d'IHM spécifique à concevoir pour cela, ce rôle étant rempli par le navigateur web.

Dans ce contexte, l'Internet des Objets (ie.l'interconnexion d'objets communicants au réseau Internet) doit délivrer tout son potentiel. Le concept de l'Internet des objets vise à intégrer le monde virtuel de l'Internet avec le monde physique des objets intelligents communicants, tels que les capteurs et les actionneurs, afin d'assurer une meilleure accessibilité et exploitation des ressources du monde réel. À travers un tel paradigme, une gigantesque expansion de l'Internet d'aujourd'hui est anticipée avec de nouveaux domaines d'applications comprenant la surveillance, la sécurité, la santé, les maisons et villes intelligentes, l'agriculture de précision, et les systèmes de logistique et de transportation intelligents.

Exemple : Raspberry Pi et sa caméra utilisée comme détecteur de mouvements avec enregistrement des prises de vue et stockage des fichiers sur un disque dur WiFi ou un emplacement défini.

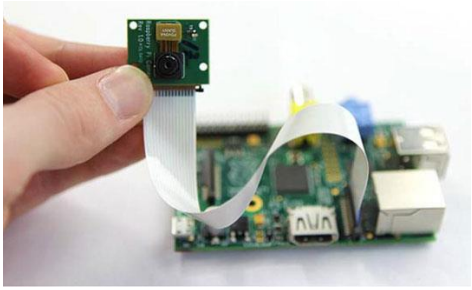


Figure I-18. Raspberry Pi comme détecteur de mouvements.

L'évolution technologique autour de l'Internet des Objets, comme indiqué sur la figure de I-19 SRI Business Consulting Intelligence, montre bien les premières applications de l'IoT dans le milieu des chaines de distribution, avec l'introduction massive de la technologie RFID. Les prochaines applications sont autour de l'exploitation des technologies de capteurs, ensuite les services basés sur les informations de localisation, et sur l'horizon 2020 on attend des services de contrôle-commande à distance des objets connectés dans différents secteurs verticaux [9].

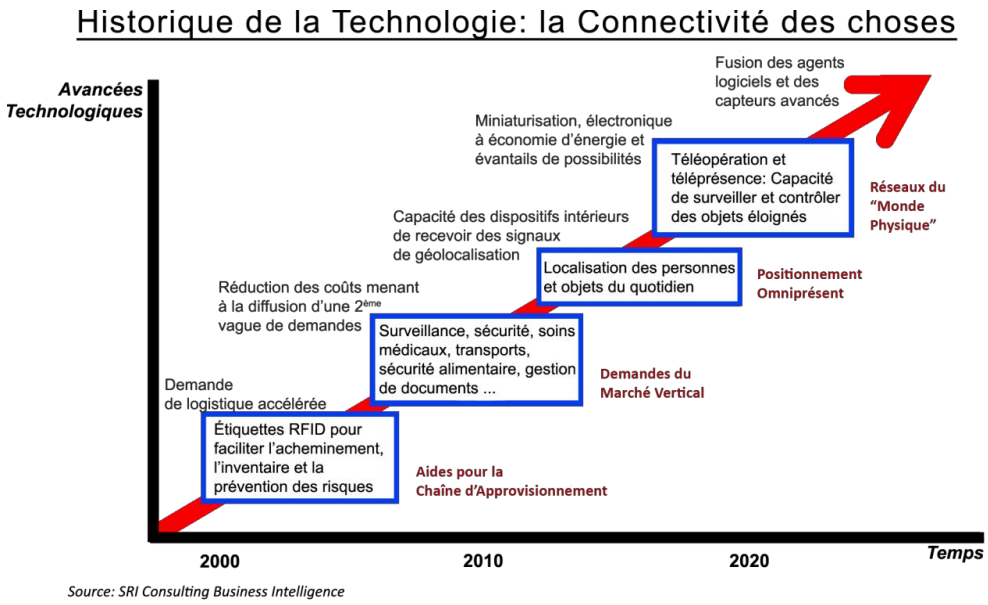


Figure I-19. Le développement de l'Internet des Objets.

I.7 Big Data

Chaque jour, nous générons 2,5 trillions d'octets de données. A tel point que 90% des données dans le monde ont été créées au cours des deux dernières années seulement. Ces données proviennent de partout : de messages sur les sites de médias sociaux, d'images numériques et de vidéos publiées en ligne, d'enregistrements transactionnels d'achats en ligne et de signaux GPS de téléphones mobiles, et surtout de capteurs utilisés dans les systèmes cyberphysiques et de l'Internet des Objets, Ces données sont appelées **Big Data** ou volumes massifs de données.

Figure I-20. Croissance de la capacité mondiale de stockage de données et informations.

Ces ensembles de données qui deviennent tellement volumineux qu'ils en deviennent difficiles à travailler avec des outils classiques de gestion de base de données ou de gestion de l'information. Les perspectives du traitement des Big Data comprennent de nouvelles possibilités d'exploration de l'information, de connaissance et d'évaluation, d'analyse tendancielle et prospective et de gestion des risques.

Il existe de nombreuses utilisations du Big Data dans chaque secteur d'activité, depuis l'analyse de volumes de données en constante croissance exponentielle pour obtenir des réponses plus précises et donc plus pertinentes, jusqu'à l'analyse de données en mouvement afin d'exploiter des opportunités autrefois perdues [10].

- **Telecoms** : réduction de 92 % de la durée de traitement grâce à l'analyse des données réseau et d'appels.
- **Services publics** : amélioration de 99 % de la précision de placement des ressources énergétiques grâce à l'analyse de 2,8 pétaoctets de données non exploitées.
- **Santé** : diminution de 20 % de la mortalité des patients grâce à l'analyse en continu des données patient.
- **Autorités/Police** : surveillance multimodale en temps réel, détection de la cyber-criminalité.
- **Transports** : réduction de la congestion du trafic routier.

- **Médias numériques** : analyse de sites web...etc [11].

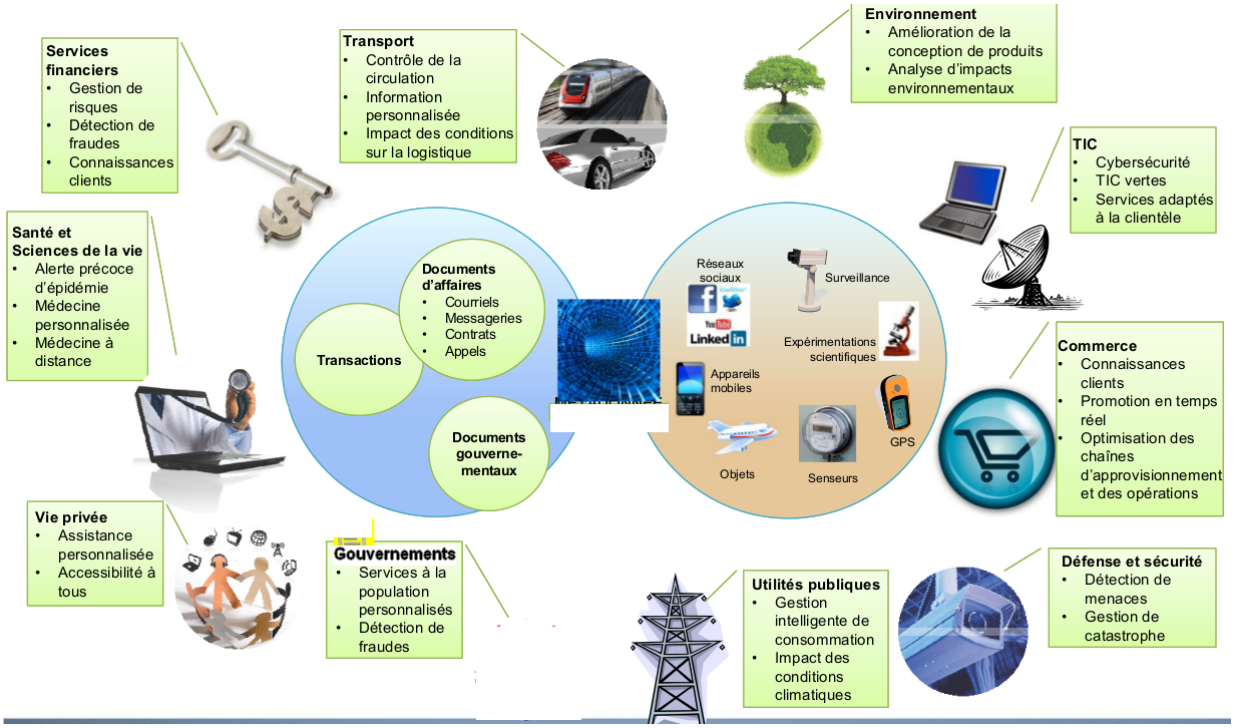


Figure I-21. Les appications du Big Data en résumé.

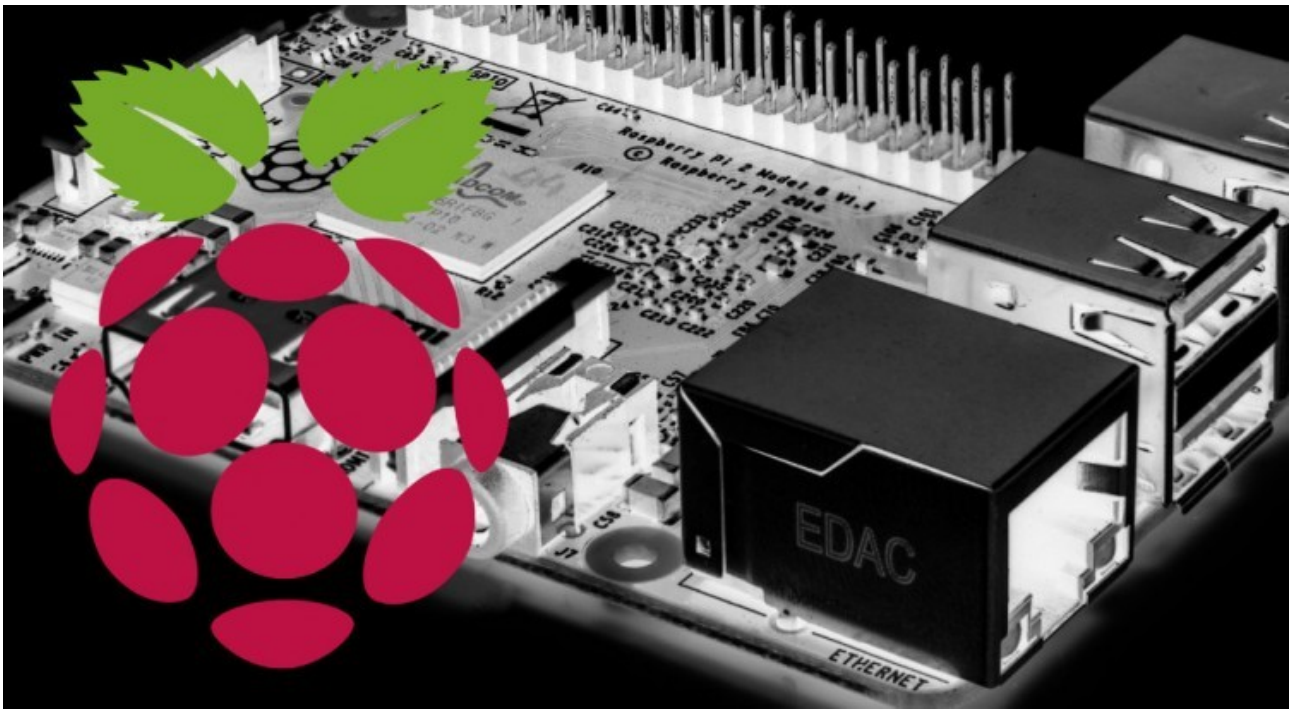
I.8 Conclusion

Nous sommes donc aujourd'hui entourés, on pourrait presque dire envahis, par les systèmes embarqués puisqu'ils sont présents absolument partout. La conception de ces systèmes pose de nombreuses contraintes spécifiques : consommation d'énergie, encombrement, parcimonie des ressources matérielles disponibles, etc. Quelque soit le contexte de l'application embarquée à développer, les logiciels libres offrent des solutions viables aussi bien techniquement qu'économiquement.

Donc nous avons vu au cours de ce chapitre qu'est-ce que les systèmes embarqués leur branches comme l'Internet des Objets et les systèmes cyberphysiques. Les points les plus importants ont été présentés, et on a terminer par les Big Data qui devient important pour l'avenir. Dans ce contexte, l'évolution des technologies des systèmes sur puce SoCs (System on Chip) offre aujourd'hui des perspectives d'intégration importante. Désormais, des systèmes complexes peuvent être intégrés dans des processeurs sur le même circuit SoC, par exemple le **Raspberry Pi** que nous allons voir dans le chapitre suivant.

Chapitre II

Le Raspberry Pi



II.1 Introduction

Les ordinateurs embarqués sous le système d'exploitation Linux sont massivement présents dans les technologies modernes (transports, multimédia, téléphone mobile, appareils photos...).

Dans ce chapitre on va voir un ordinateur qui devenu un phénomène mondialde s'appelle Raspberry Pi, de taille d'une carte de crédit, pas cher et peu gourmand en énergie, équipé d'un processeur ARM, s' exécute sous le système d'exploitation Linux.

Ce chapitre présente les composantes, l'installation et la configuration de la carte électronique Raspberry Pi. Après que nous avons installé notre Raspberry Pi et qu'il est prêt à l'emploi, nous pouvons commencer à explorer ses caractéristiques et les bases de Linux.

Un des grands intérêts du Raspberry Pi est qu'il dispose de GPIO utilisables comme entrée ou sortie afin de lire des capteurs ou commandes des systèmes. Nous nous verrons ici comment contrôler ses broches et les configurer en mode sortie, à travers un exemple concret ou nous commanderons une LED. Cela peut être réalisé par n'importe quel langage de programmation par exemple Python.

II.2 Raspberry Pi

II.2.1 Un peu d'histoire

Le concept de Raspberry Pi a été dévoilé autour de 2006 par Eben Upton et ses collègues de l'université de Cambridge ont remarqué que les aspirants à un diplôme en informatique avaient moins d'expérience que ceux des années 1990. Ils ont attribué ce recul, entre autres facteurs, à « l'émergence chez les particuliers du PC et des consoles de jeu qui ont remplacé les Amiga, BBC Micro, Spectrum ZX et Commodore 64, des machines avec lesquelles la génération précédente avait appris à programmer ». Le fait que l'ordinateur soit devenu si important pour tous les membres du foyer peut aussi décourager les jeunes de s'aventurer dans ce domaine, à cause du risque d'endommager un outil important pour toute la famille. Mais récemment, les technologies qui ont permis aux téléphones mobiles et tablettes de devenir de moins en moins chers et de plus en plus puissants, ont projeté le Raspberry Pi dans le monde des cartes « ultra-économiques-mais-super-faciles-à-utiliser ».

Dans une interview pour BBC News, Linus Torvalds, fondateur du noyau Linux, a même dit que le Raspberry Pi « rend l'échec abordable » [11].

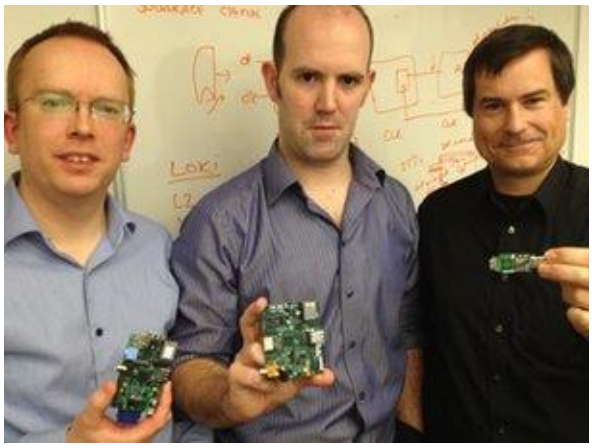


Figure II-1. Eben Upton (en milieu) et ses collègues.

II.2.2 Description de la carte Raspberry Pi

Raspberry Pi est un petit ordinateur sous le système d'exploitation Linux sur carte SD destiné à des applications d'informatique embarquée. Le cœur de l'ordinateur est un FPGA (Broadcom 2835) intégrant un processeur ARM11 cadencé à 700MHz et de nombreux périphériques. Raspberry Pi peut être directement connecté à une IHM (Interface Homme Machine) classique, souris/clavier/écran HDMI ou vidéo composite, cependant comme tout ordinateur Linux. Raspberry Pi peut intégrer ses propres outils de développement et une IHM reposant sur SSH contrôlable depuis un autre ordinateur par Ethernet ou WiFi.

Le connecteur d'extension supporte les entrées/sorties parallèles ainsi que la plupart des bus de communication. C'est un support particulièrement économique et puissant qui peut être facilement mis en œuvre dans de petits systèmes nécessitant un accès au monde physique par des capteurs/actionneurs disposants d'interfaces numériques [12].

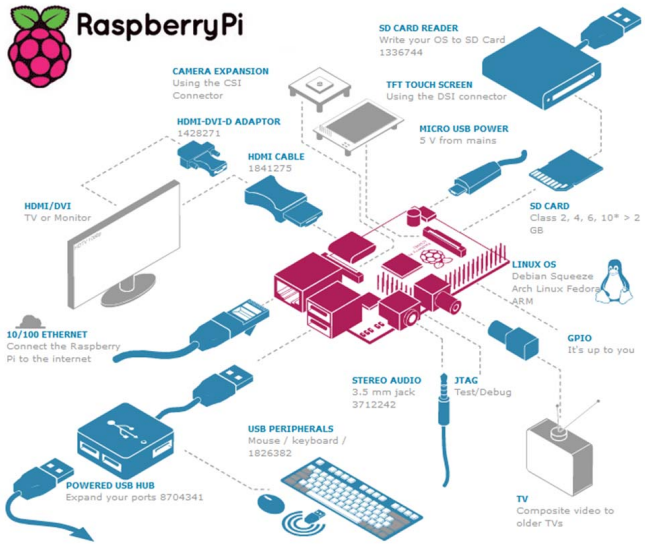


Figure II-2. Architecture de connexions du Raspberry Pi.

II.2.3 Modèles de Raspberry Pi

Deux versions de Raspberry Pi sont commercialisées; la version A et la version B. Si le processeur qui les équipe est identique, tout comme leur architecture interne, voici ce qui les différencie :

Le Pi modèle B est doté de 512 Mo de RAM alors que le modèle A ne dispose que de 256 Mo (mais il consomme moins d'électricité).

Le modèle B possède deux ports USB, contrairement au modèle A qui n'en possède qu'un.

Le modèle B du Raspberry Pi possède une prise Ethernet standard au format RJ45. Le modèle A n'en a pas mais il est possible de le connecter à un réseau avec un adaptateur USB vers Ethernet [12].

Le 2 février 2015, la fondation Raspberry Pi annonce la sortie du Raspberry Pi 2, plus puissant, il est équipé d'un processeur Broadcom BCM 2836, quatre cœurs ARMv7 à 900 MHz, accompagné de 1Go de RAM [13].

Nous nous pencherons ici sur les modèles les plus couramment diffusés.

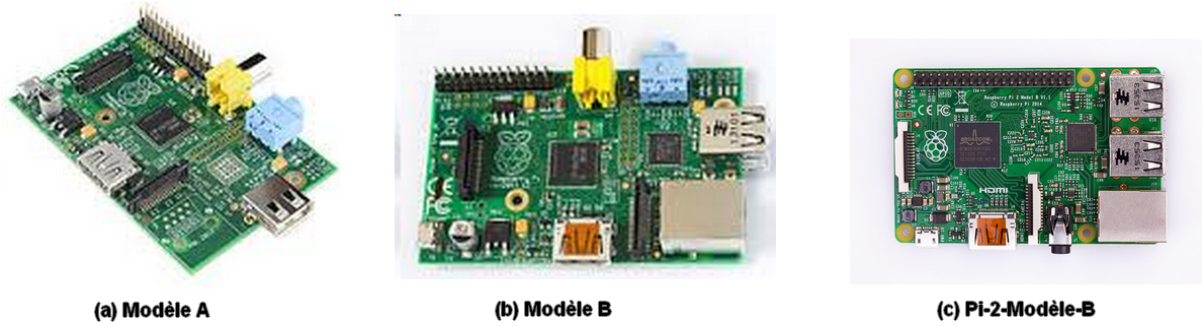


Figure II-3. Les différentes Modèles du Raspberry Pi.

Modèle	Processeur	RAM	Les interfaces USB2.0	Ethernet	Entrées/sorties
Modèle A	ARM-v6	256 Mo	1	0	GPIO 26 pts
Modèle B	ARM-v6	512 Mo	2	1	GPIO 26 pts
Pi-2 Modèle B	ARM-v7	1Go	4	1	GPIO 40 pts

Tableau II-1. Comparatif des spécifications des modèles du Raspberry Pi.

II.2.4 Les composantes du Raspberry Pi

Vous serez peut-être tenté de considérer le Raspberry Pi comme une carte à microcontrôleur comme Arduino, ou comme un substitut à votre ordinateur portable. En réalité, la structure du Raspberry Pi ressemble davantage à celle d'un appareil mobile, doté de plusieurs connecteurs permettant d'accéder facilement à ses nombreux ports et fonctions. La figure II-4 représente les différents composants de la carte.

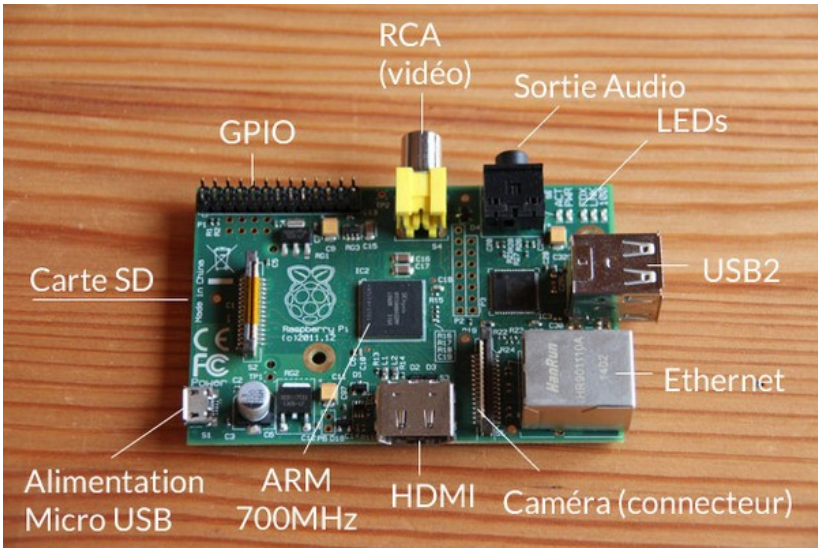


Figure II-4. Emplacement des ports et interfaces matérielles du Raspberry Pi.

II.2.4.1 Le processeur (Broadcom BCM 2835)

Le cœur du Raspberry Pi est équipé d'un processeur de même type que celui de l'iPhone 3G. La puce centrale est un System on Chip (SoC) cadencé à 700 MHz, utilisant l'architecture ARM11 conçue par la société anglaise ARM Holdings.

Figure II-5. Le processeur du Raspberry Pi.

II.2.4.2 Le support de carte SD (Secure Digital)

Comme vous pourrez rapidement le constater, le Raspberry Pi ne possède pas de disque dur, tout est stocké sur une carte SD.

Figure II-6. Le support de carte SD.

II.2.4.3 La prise USB

Utiliser pour connecter n'import quelle périphérique de type USB comme clavier, souris, carte réseau sans fil...etc.

Figure II-7. Les interfaces USB du Raspberry Pi.

II.2.4.4 La prise Ethernet

Raspberry Pi possède une prise Ethernet standard au format.

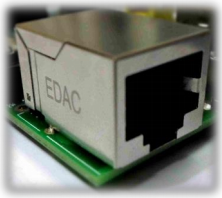


Figure II-8. La prise Ethernet RJ45.

II.2.4.5 La prise HDMI

Le port HDMI fournit les sorties vidéo et audio numériques et il supporte 14 résolutions vidéo différentes.

Figure II-9. La prise HDMI.

II.2.4.6 Les LED d'état

Le Raspberry Pi dispose de cinq LED qui renseignent sur l'activité de la carte (tableau II-2).

Figure II-10. Les LED d'état.

LED	Couleur	Commentaires
ACT	Vert	S'allume lors des accès à la carte SD.
PWR	Rouge	Témoin de l'alimentation 3.3 V.
FDX	Vert	S'allume lorsque le port Ethernet est en mode full duplex.
LNK	Vert	Témoin d'activité du port Ethernet.
100	Jaune	Indique que la connexion au réseau est en mode 100 Mbps.

Tableau II-2. Fonctionnalité de chaque LED d'état.

II.2.4.7 La sortie vidéo composite

C'est une prise standard de type RCA qui fournit les signaux vidéo composites. Le format composite a une résolution très basse par rapport à celle du HDMI.

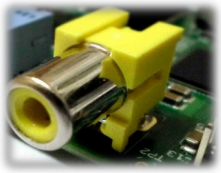


Figure II-11. La sortie vidéo.

II.2.4.8 Le connecteur d'alimentation

Le connecteur micro USB sert uniquement à alimenter la carte et il fonctionne comme un interrupteur.

Figure II-12. Connecteur d'alimentation.

II.2.4.9 Les broches d'entrées-sorties (GPIO)

Les ports GPIO (General Purpose Input/Output) très utilisés dans le monde des microcontrôleurs et de l'électronique embarquée.

Ces broches GPIO offre à la Raspberry Pi la possibilité de communiquer avec d'autres circuits électroniques.

Il existe plusieurs bibliothèques facilitent la programmation des connecteurs GPIO par exemple : wiringPi-langage C, Rpi.GPIO-Python, Pi4J-Java.

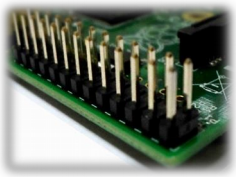


Figure II-13. Les broches GPIO.

II.2.4.10 Le connecteur Display Serial Interface (DSI)

Ce connecteur accepte un câble plat à 15 conducteurs qui peut communiquer avec un afficheur LCD ou OLED.

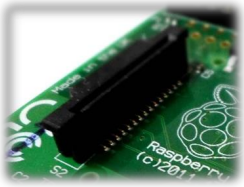


Figure II-14. Le connecteur DSI.

II.2.4.11 Le connecteur Camera Serial Interface (CSI)

Ce port permet de connecter directement à la carte un module caméra spécifique.

Figure II-15. Le connecteur CSI.

II.2.4.12 La sortie audio analogique

La carte comporte une prise jack standard de 3,5 mm qui fournit un signal sonore stéréo à haute impédance, destiné à des appareils comme des haut-parleurs amplifiés.

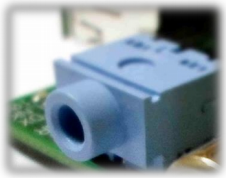


Figure II-16. La sortie audio analogique.

II.2.5 Processeur ARM Cortex du Raspberry Pi

Le processeur qui anime le système Raspberry Pi est un processeur multimédia **Broadcom BCM2835** de type **SoC** ((System-on-Chip), tout le système sur un seul circuit). Cela signifie que la plupart des composants du système (unité centrale, coprocesseur graphique, matériel audio et vidéo) sont réunis dans un seul composant, qui est physiquement caché sous le circuit mémoire de 256, 512 Mo ou 1 Go qui est implanté au centre de la plaque.

Le processeur BCM2835 ne se distingue pas seulement des processeurs pour PC de bureau par cette conception intégrée SoC. Plus important encore, il utilise un autre jeu d'instructions machine (ISA, Instruction Set Architecture) nommé **ARM.**

Elle n'est pas très répandue dans le monde des ordinateurs de bureau, mais elle l'est beaucoup plus dans l'univers des appareils nomades, de nombreux modèles de smartphones fonctionnent avec un processeur ARM. Cette architecture utilise un jeu d'instructions machine réduit (RISC). La faible consommation électrique des circuits ARM leur donne un net avantage par rapport aux processeurs des ordinateurs de bureau qui se basent sur un jeu d'instructions complexes (CISC).

C'est grâce à ce circuit ARM BCM2835 que le Raspberry Pi peut fonctionner avec une alimentation 5V 1A, et donc être alimenté par le port micro USB de la carte. Si vous cherchez,

vous ne trouverez aucun radiateur, car la consommation du circuit rend ce genre de dissipateur de chaleur inutile, même dans les périodes de forte charge du processeur.
En revanche, puisque le Raspberry Pi se base sur un jeu d'instructions machine différent, les programmes ne sont pas compatibles avec ceux des PC. En effet, les logiciels conçus pour les ordinateurs de bureau et portables ciblent l'architecture x86 qui est celle des processeurs AMD, Intel et VIA. Tous ces programmes ne fonctionnent donc pas sur le Raspberry Pi et son processeur ARM.

Le BCM2835 fait partie de la génération de processeurs ARM nommée ARM11, qui utilise la version de l'architecture du jeu d'instructions ARMv6. Il est important de le savoir, car ARMv6 est une architecture légère tout en étant puissante, mais légèrement inférieure à l'architecture ARMv7 adoptée par la famille de processeurs ARM Cortex. Les logiciels conçus pour ARMv7, comme ceux pour les x86, ne sont pas compatibles avec le processeur [14].

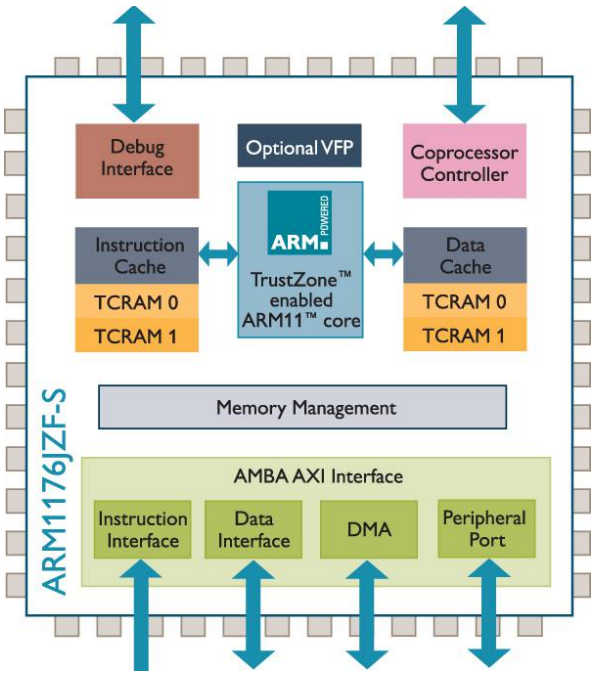


Figure II-17. Processeur ARM Cortex du Raspberry Pi.

II.3 Les différentes distributions Linux pour le Raspberry Pi

Comme nous l'avons dit précédemment, le Raspberry Pi fonctionne généralement avec un système d'exploitation basé sur Linux.

Techniquement parlant, Linux n'est qu'un noyau alors qu'un système d'exploitation est un ensemble de différents éléments (drivers, services et applications). Il existe de nombreuses distributions Linux, très diversifiées et qui évoluent au fil des années. Les distributions les plus courantes sur les ordinateurs de bureau sont Ubuntu, Debian, Fedora et Arch. Chacune d'entre elles est adaptée à des applications spécifiques et possède sa propre communauté d'utilisateurs. Comme le Pi est basé sur un SoC pour appareil mobile, il n'utilise pas les mêmes logiciels qu'un ordinateur de bureau.

Tandis que la plupart des ordinateurs offrent plusieurs gigaoctets de RAM et des centaines de gigaoctets de stockage, le Raspberry Pi est quant à lui plus limité dans ces deux domaines.

On trouve de plus en plus de distributions Linux développées ou adaptées spécifiquement pour le Raspberry Pi, les plus connues sont les suivantes.

II.3.1 Raspbian

C'est une distribution basée sur Debian Wheezy qui comporte plus de 35000 paquets. Par défaut elle est installée avec le bureau LXDE. C'est la distribution recommandée par la fondation Raspberry Pi.

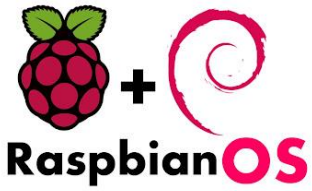


Figure II-18. Logo du Raspbian.

II.3.2 Arch Linux

La célèbre distribution pour le Raspberry Pi. Elle est réservée aux utilisateurs expérimentés de Linux. Elle ne propose aucun bureau graphique.

Figure II-19. Logo du Arch Linux.

II.3.3 Risc Os

Un système d'exploitation conçu par Acorn en 1988. Pauvre par le nombre de logiciels.

Figure II-20. Logo du Risc Os.

II.3.4 Pidora

Une version de Fedora pour les processeurs ARM.

Figure II-21. Logo du Pidora.

II.3.5 Raspberry Pi Linux version éducative d'Adafruit (Occidentalis)

La distribution d'Adafruit basée sur Raspbian contient un certain nombre d'outils et de drivers utiles pour apprendre l'électronique.

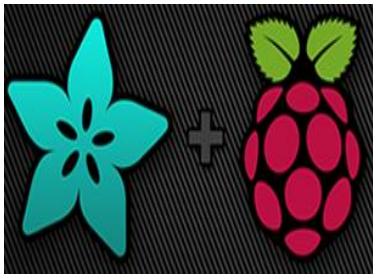


Figure II-22. Logo de la version éducative d'Adafruit.

II.3.6 NOOBS

Le programme d'installation NOOBS (New Out Of Box Software) permet de faire cela. Par défaut il contient six distributions officielles de la fondation Raspberry Pi (Raspbian, Arch, Pidora, RiscOS, openELEC et Raspbmc).

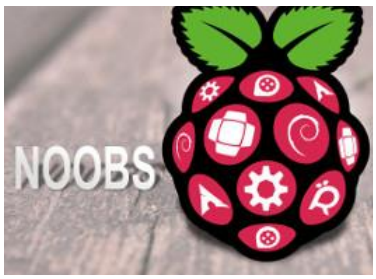


Figure II-23. Logo du NOOBS.

II.4 Préparation de la carte SD

II.4.1 Téléchargement du Raspbian

La première étape consiste à télécharger Raspbian à partir de la page de téléchargements http://www.raspberrypi.org/downloads, puis téléchargez l'image compressée (fichier.zip).

Figure II-24. Téléchargement du Raspbian à partir de son site officiel.

On décompresse le fichier.zip. Nous obtenons alors un fichier de 2 Go environ.

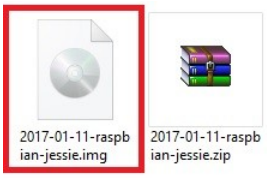


Figure II-25. L'image du Raspbian obtenu après la décompression.

En suite, on a vérifiez l'intégrité de notre image disque d'origine. Pour cela, nous pouvons utiliser un logiciel de signature numérique pour calculer la signature SHA (Secure Hash Algorithm) de l'image et on compare le résultat à l'empreinte de 40 caractères avec qui est dans la page de téléchargements.

II.4.2 Écrire une carte SD

II.4.2.1 Les périphériques nécéssaires

Une carte SD

Il faut utiliser carte d'une capacité d'au moins 4 Go et de classe 4 (ou plus) de préférence parce que les cartes de classe 4 sont capables de transférer au moins 4 Mo par seconde.
Une carte micro SD dans un adaptateur convient également.

Figure II-26. Carte SD.

Lecteur de carte micro SD pour port USB

Utiliser pour installer le système d’exploitation sur une carte SD.

Figure II-27. Lecteur de la carte SD.

II.4.2.2 L'installation du système Raspbian sur la carte SD

(1). Téléchargez et installez le programme Win32DiskImager https://launchpad.net/win32-image-writer.
(2). Insérez la carte SD dans son lecteur, puis sur le PC (a) et notez la lettre du lecteur qui lui a été assignée par l'explorateur de fichiers (b).

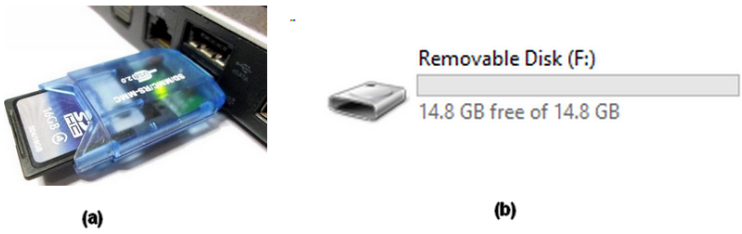


Figure II-28. Insertion le lecteur de carte sur PC.

(3). Lancez le programme Win32DiskImager et sélectionnez le fichier contenant l'image à installer (coleur jaune) et choisissez la lettre correspondant à votre carte SD (coleur rouge), puis cliquez sur Write.

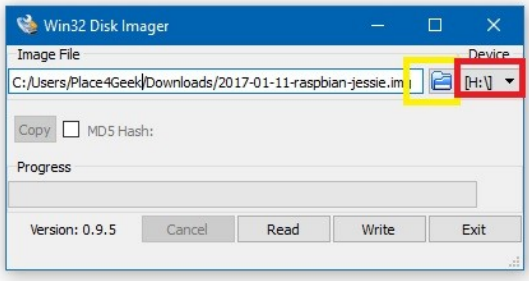


Figure II-29. Écriture de Raspbian sur une carte SD.

(4). Une fois la copie terminée, éjectez la carte et insérez-la dans votre Raspberry Pi.

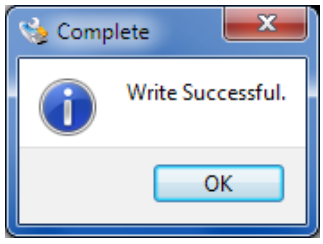


Figure II-30. La fenêtre qui affiche la fin d'écriture sur la carte.

II.5 Démarrage et la configuration du Raspberry Pi

II.5.1 Démarrage du Raspberry Pi

Suivez ces étapes pour démarrer votre Raspberry Pi pour la première fois.

1. Insérez la carte SD dans son support.

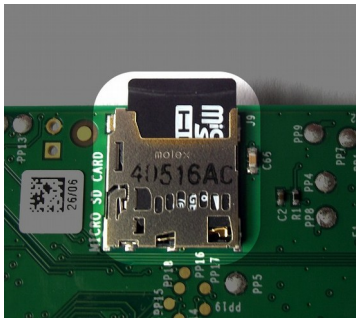


Figure II-31. Carte SD dans son support sur le Raspberry Pi.

2. Branchez un clavier USB et une souris, connectez la sortie HDMI à votre téléviseur au Raspberry Pi, et en suite l'alimentation. De manière générale, vérifiez que tous les autres éléments sont bien branchés avant de connecter l'alimentation.

Figure II-32. Différent éléments branché au Raspberry Pi.

Pour l'alimentation

Vous devez impérativement utiliser une fiche micro USB, capable de fournir au moins 700 mA de courant. Une alimentation de 1A est recommandée.

Enfin, Voilà notre système finale pour débuter à configurer le Raspberry Pi.

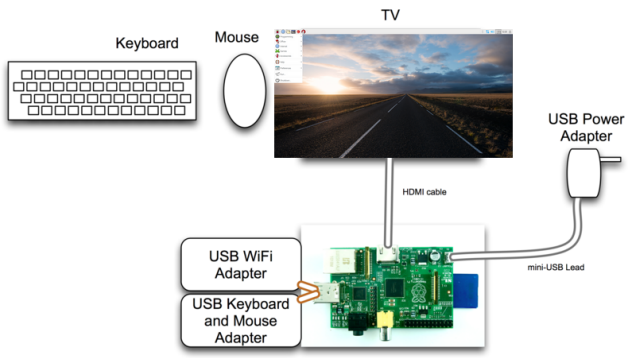


Figure II-33. Connexion de l'ensemble des éléments sur Raspberry Pi.

Si tout se passe correctement, vous verrez une liste de messages défiler à l'écran. Ces messages de log montrent tous les processus du système d'exploitation qui se lancent au démarrage de votre Raspberry Pi.

Au tout premier démarrage de notre Raspberry Pi, l'outil **raspi-config** se lancera automatiquement pour personnaliser notre carte (figure II-34).

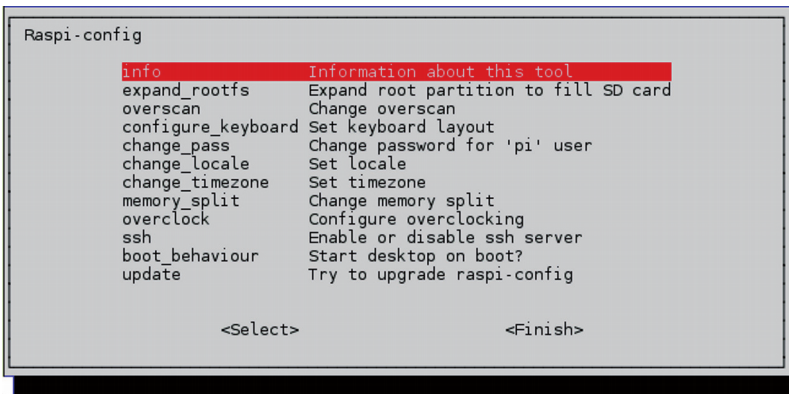


Figure II-34. Le menu principal de l'outil raspi-config.

II.5.2 La configuration du Raspberry Pi

Nous allons à présent passer en revue les différents paramètres de base et distinguer ceux qui sont essentiels de ceux qui pourront être réglés par la suite.

Dans l'outil **raspi-config**, utilisez les flèches de direction du clavier haut et bas pour naviguer dans la liste. La barre d'espacement permet de sélectionner une option et la touche Tab de passer d'un champ à un autre ou de déplacer le curseur en bas de l'écran pour quitter.

Les options disponibles sont les suivantes :

expand_rootfs : vous devez toujours choisir cette option, qui agrandit le système de fichiers Linux, afin d'utiliser l'intégralité d'une carte SD. Par exemple, si vous disposez d'une carte SD de 4 Go, Linux pourra accéder à 2 Go supplémentaires.

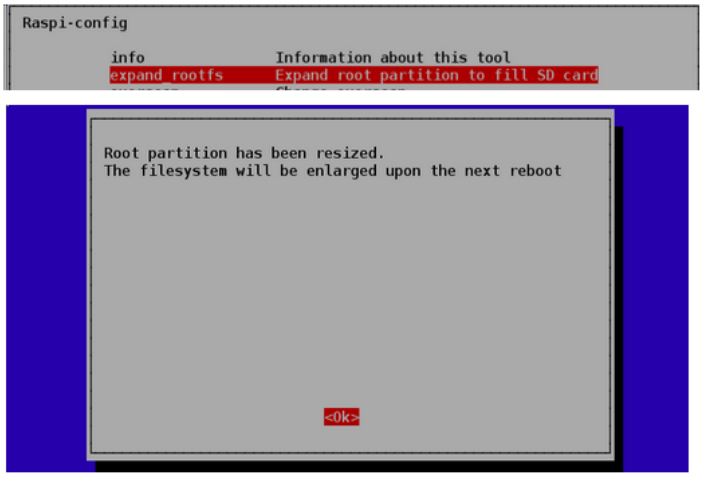


Figure II-35. L'option expand_rootfs de l'outil raspi-config.

Overscan : si vous possédez un moniteur haute définition, il est possible que du texte dépasse de votre écran. Pour résoudre ce problème, activez L'option overscan et modifiez les valeurs

pour que l'image s'adapte à la taille de votre écran. Entrez des valeurs positives si l'image sort de l'écran, des valeurs négatives si des marges noires apparaissent sur les côtés.
Keyboard : par défaut, les paramètres du clavier sont réglés pour un clavier britannique (QWERTY). Pour que les bonnes lettres soient saisies lorsque vous tapez sur les touches correspondantes, vous devez choisir le type de clavier approprié. Heureusement, Linux supporte énormément de langues et de dispositions de touches.
Password : nous vous conseillons de modifier le mot de passe par défaut (raspberry) pour le rendre plus fiable.
Change_locale : pour activer notre langue et nos préférences d'encodage des caractères. Le paramètre par défaut est l'anglais (en) britannique (UK) avec un encodage de caractères en UTF-8.
Change_timezone : modifiez cette option pour sélectionner le fuseau horaire correspondant au lieu où vous vous trouvez.
memory_split : cette option vous permet de changer la quantité de mémoire allouée à l'affichage.
Overclock : vous avez la possibilité de faire tourner le processeur à des vitesses supérieures à 700 MHz. Pour votre premier démarrage, conservez les paramètres par défaut. Si par la suite vous devez modifier ce paramètre, essayez les valeurs Medium ou Modest. Le mode Turbo permet de monter à 1GHz avec un dissipateur.
ssh (Secure Shell) : cette option active l'accès au shell sécurisé, ce qui vous permet de vous connecter à votre Raspberry Pi en passant par le réseau.
boot_behaviour : cette option est sélectionnée par défaut et active le démarrage automatique de l'environnement de bureau graphique. Si vous sélectionnez No, le Pi restera en mode texte. Vous pourrez toujours lancer l'interface graphique manuellement, comme suit :
raspberrypi login : pi
Password : raspberry (ou le mot de passe que vous venez de paramétrer).
pi@raspberrypi ~ $ startx
Quand vous êtes dans l'interface graphique, il n'y a plus de ligne de commande.
Vous pouvez en obtenir une ou plusieurs en lançant un programme de terminal.
Pour cela, cliquez sur le menu Bureau en bas à gauche et choisissez Accessoires>LXTerminal.
Update : si vous êtes connecté à Internet, cette option vous permettra de mettre à jour l'utilitaire de configuration.

Une fois la configuration terminée, sélectionnez **Finish** (accédé avec la touche Tab). Selon les options que vous aurez modifiées, il se peut que l'outil raspi-config redémarre le système. Sinon, vous serez redirigé vers la ligne de commande sur laquelle vous saisirez :
pi@raspberrypi ~ $ sudo reboot
afin que votre Raspberry Pi redémarre et prenne en compte vos nouveaux paramètres.

II.5.3 Éteindre le Raspberry Pi

Il n'existe pas de bouton d'alimentation sur le Raspberry Pi, quoique les nouvelles cartes soient équipées d'une broche permettant d'accueillir un bouton de réinitialisation. Pour éteindre correctement le système d'exploitation, allez dans le menu Logout de l'interface graphique et sélectionnez shutdown.
Vous pouvez aussi éteindre le Raspberry Pi depuis la ligne de commande en tapant :
pi@raspberrypi ~ $ sudo shutdown -h now
ou bien une version plus courte :
pi@raspberrypi ~ $ sudo Ralt

Veillez bien à arrêter le système d'exploitation comme indiqué ici et à ne pas débrancher sauvagement votre Raspberry Pi. Si vous coupez le courant sans arrêter le système correctement, vous risquez d'endommager la carte SD [15].

II.6 Linux

Pour profiter au mieux de toutes les possibilités offertes par notre Raspberry Pi, il est souhaitable de connaître un peu Linux.

Linux est un système d'exploitation open source. Ce logiciel a été écrit dans le cadre d'un projet communautaire pour ceux qui recherchaient une alternative au monopole de Microsoft Windows et d'Apple OS X. C'est un système d'exploitation complet, basé sur les mêmes concepts robustes d'Unix qui est apparu au début de l'informatique. Il a de nombreux partisans fidèles et serviables et s'est transformé en un système d'exploitation puissant et facile à utiliser.

II.6.1 Brève histoire

En 1984, Richard Matthew Stallman, chercheur en informatique du MIT quitte son poste et se consacre à l'écriture d'un système d'exploitation libre du nom de GNU.

Le projet GNU arrive en 1991 avec de très nombreux outils libres, mais il lui manque un élément central, le **noyau**. Cet élément est essentiel car il gère la mémoire, le microprocesseur, les périphériques comme le clavier, la souris, les disques durs...etc. C'est à cette époque qu'un étudiant finlandais, Linus Torvalds, commence à développer un noyau et demande aux personnes intéressées d'y contribuer.

Le système d'exploitation actuellement connu est donc un assemblage des outils GNU fonctionnant sur un noyau Linux, on parle donc de GNU/Linux avec le slash, « / » pour « GNU sur Linux ».

Puisque, Debian « jessie » est recommandé par la fondation Raspberry Pi avec sa version dédiée Raspbian (Raspberry & Debian) nous allons travailler avec celui-ci.

II.6.2 Raspbian

Raspbian est un système d'exploitation libre basé sur la distribution GNU/Linux Debian, et optimisé pour le plus petit ordinateur du monde, qui est le Raspberry Pi.

Raspbian ne fournit pas simplement un système d'exploitation basique, il est aussi livré avec plus de 35000 paquets, c'est-à-dire des logiciels précompilés livrés dans un format optimisé, pour une installation facile sur notre Raspberry Pi via les gestionnaires de paquets.

II.6.2.1 Découvrir l'interface graphique

Sous GNU/Linux, il existe un grand nombre de GUI dont les plus connus sont GNOME et KDE. C'est derniers sont très gourmands en ressources (mémoire RAM, processeur ...) et sont donc peu adaptés à un environnement embarqué.

La distribution Raspbian est livrée avec LXDE (Lightweight X11 Desktop Environment), un environnement de bureau graphique simplifié qui tourne sur des ordinateurs Unix et Linux depuis les années 1980. Une partie des outils visibles sur le Bureau et dans le menu (l'éditeur de texte Leafpad et le shell LXTerminal, par exemple) sont fournis avec LXDE.

Les différents outils que vous serez amené à utiliser fréquemment sont les suivants :

Le gestionnaire des fichiers (File Manager)

Si vous préférez déplacer vos fichiers sans passer par la ligne de commande, sélectionnez File Manager dans le menu Accessories. Vous pourrez alors naviguer dans le système de fichiers grâce à des icônes et des dossiers, comme vous en avez probablement l'habitude.

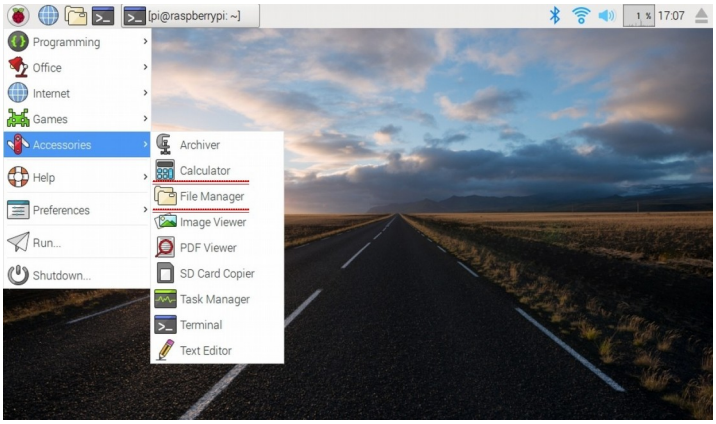


Figure II-36. Le gestionnaire des fichiers de l'interface graphique du Raspbian.

Le navigateur web

Le navigateur web par défaut, Midori, a été élaboré de manière à pouvoir fonctionner avec peu de ressources. On trouve aussi navigateur web comme Dillo et NetSurf.

Figure II-37. Différentes navigateur web sur Raspbian.

II.6.2.2 Les fichiers et le système de fichiers

Le tableau II-3 liste certains des répertoires les plus importants dans le système de fichiers de Raspbian. La plupart d'entre eux suivent le standard Linux pour leur structure. Quelques-uns sont spécifiques au Raspberry Pi. Le répertoire /sys permet d'accéder à tout le matériel de notre Raspberry Pi.

Répertoire	Description
/	La « racine », c'est l'entrée du labyrinthe du système de fichiers.
/bin	Programmes et commandes pour tous les utilisateurs.
/boot	Tous les fichiers nécessaires au démarrage du noyau.

/dev	Fichiers spéciaux qui représentent les périphériques installés. Des fichiers virtuels qui permettent de communiquer avec des périphériques, comme les ports série, l'écran...etc.
/etc	Fichiers de configuration du système.
/etc/init.d	Scripts pour démarrer les services du système.
/etc/X11	Fichiers de configuration pour le mode graphique X11.
/home	Répertoires personnels des utilisateurs.
/home/pi	Répertoire personnel pour l'utilisateur Raspberry Pi.
/lib	Bibliothèques partagées et modules du noyau.
/media	Points de montage pour les supports amovibles (c'est ici qu'apparaîtra notre clé USB Flash, par exemple).
/proc	Répertoire virtuel contenant des informations sur les processus en cours et le système d'exploitation.
/sbin	Programmes pour l'administration du système.
/sys	Répertoire spécial contenant des fichiers qui contrôlent le matériel.
/tmp	Emplacement dans lequel les programmes peuvent créer des fichiers temporaires (ils sont effacés à chaque redémarrage).
/usr	Programmes et données accessibles par tous les utilisateurs.
/usr/bin	La plupart des programmes du système d'exploitation se trouvent ici.
/usr/src	Linux est open source, voici donc les codes sources.
/var/log	Archive des messages du système (logs).

Tableau II-3. Certains des répertoires les plus importants dans le système fichiers de Raspbian.

II.6.2.3 Le shell

Beaucoup de tâches nécessitent l'utilisation de la ligne de commande.

Le programme LXTerminal vous permet d'accéder à cette ligne de commande et au shell. Le shell par défaut sur Raspbian, ainsi que sur la majorité des systèmes Linux, s'appelle Bourne-Again Shell (ou bash). Il existe une alternative appelée dash.

Figure II-38. LXTerminal.

Notre répertoire actuel s'affichera devant l'invite de commande, aussi appelée le prompt. Sous Linux, notre répertoire personnel (home) est aussi représenté avec le caractère ~ pour faire plus court. À l'ouverture de LXTerminal, nous nous trouverons dans notre répertoire personnel et notre prompt ressemblera à ceci :

pi@raspberry ~ $

Voici une explication de ce prompt

pi ❶ @raspberry ❷ ~ ❸ $ ❹

❶ Notre nom d'utilisateur, pi, suivi du caractère arobase (@).

❷ Le nom de notre ordinateur (raspberry est le nom par défaut).

❸ Le répertoire courant (current working directory) du shell. Lorsque nous démarrons une session, nous nous trouverons toujours dans notre répertoire personnel (~).

❹ Le prompt du shell (ou invite de commande). Tout texte que vous saisirez apparaîtra à sa droite.

Appuyez sur la touche Entrée après chacune de nos commandes pour les exécuter.

Dès que nous arrivons dans un répertoire, on utilise la commande ls (List everything) pour afficher tous les fichiers et sous-répertoires :

Pour créer un nouveau répertoire, utilisez **mkdir (make a directory)**.

```
pi@raspberrypi ~ $ mkdir hello
pi@raspberrypi ~ $ ls
blink11.py     face          lightSensor.py  python_games  test.jpeg
blink13.py     fm            motion.py       Scratch       test-opencv.py
blinkpin11.py  hello         motor1.py       scratchgpio   test.py
Desktop        h.py          mot.py          spi.py        WebIOPi-0.6.0
Documents      inputRead.py  ocr_pi.png      test.avi      WebIOPi-0.6.0.tar.gz
pi@raspberrypi ~ $
```

La commande **cd (change directory)** vous permet de naviguer dans le système de fichiers.

```
pi@raspberrypi ~ $ cd hello
pi@raspberrypi ~/hello $ pwd
/home/pi/hello
```

Comme nous allons voir la commande **pwd (Print Working Directory)** donne la position courant.

II.6.2.4 Droit d'accès sous Linux

Linux est un système mutli-utilisateur qui permet de gérer les permissions d'accès aux fichiers. Chaque utilisateur a un identifiant, un nombre unique qui l'identifie. Les utilisateurs appartiennent également à un ou plusieurs groupes. Les groupes peuvent être employés pour limiter l'accès à un certain nombre de personnes.

Les permissions d'accès pour un fichier peuvent être positionnés par **propriétaire**, **groupe**, et pour les **autres** sur la base de permissions en lecture (**r)**, écriture (**w**) et exécution (**x**). Les droits d'accès sont attribués aux fichiers ou aux répertoires.

Il y a trois catégories de droits : Lecture (**r**ead), écriture (**w**rite), exécution (e**x**ecute).
Pour les répertoires, il correspond à une interdiction d'accès au contenu de ce répertoire.
Ces droits sont définis pour 3 types d'utilisateurs :
. Le propriétaire du fichier (**u**=user).
. Le groupe auquel appartient le fichier (**g**=group).
. Tous les autres utilisateurs (**o**=other).
. **a**=all signifie user+group+other.

Pour la modification des droits d'accès

Les droits d'accès ne peuvent être modifiés que par le propriétaire ou l'administrateur.
La commande **chmod** permet de modifier les droits d'accès. Elle peut être utilisée de deux façons différentes (symbolique et octal).
Pour pouvoir changer les droits on doit spécifier :
Les droits (**r=read**, **w=write**, **x=execute**).
A qui s'appliquent ces droits (u=user, g=group, o=other,a=all).
Le ou les fichiers/répertoires dont on veut changer les droits.
Pour ajouter des droits l'opérateur + est utilisé.
Pour enlever des droitsl'opérateurr – est utilisé.

- **Méthode symbolique**

Voici quelques exemples qui vous montrent comment combiner ces arguments :
chmod u+rwx,g-rwx,o-rwx wibble.txt ❶
chmod g+wx wobble.txt ❷
chmod -xw,+r wubble.txt ❸
❶ Autorise uniquement l'utilisateur à lire, écrire et exécuter.
❷ Ajoute une permission d'écriture et d'exécution pour les membres du groupe.
❸ Permission de lecture uniquement pour tout le monde.

- **Méthode octal**

Les arguments sont souvent condensés en notation octale par exemple :
chmod 744 wabble.txt
744 signifie « tout le monde peut lire le fichier mais seul le propriétaire peut le modifier ou d'exécuter ». Comme il y a peu de combinaisons d'arguments utiles ou sûres, on n'a besoin de mémoriser que quelques nombres [16].

II.6.2.5 Le répertoire /etc

Le répertoire **/etc** contient tous les fichiers de configuration du système et les scripts de démarrage.

Lorsque vous avez lancé le programme de configuration raspi-config au premier démarrage, vous avez modifié des valeurs dans différents fichiers du répertoire **/etc**. Pour éditer ces fichiers, utilisez la commande sudo pour obtenir les droits du super utilisateur root. Ensuite, utilisez un éditeur de texte comme nano et lancez-le avec sudo :
pi@raspberrypi ~ $ sudo nano /etc/hosts

II.6.2.6 Réglage de la date et de l'heure

Les ordinateurs portables et les ordinateurs de bureau standards sont équipés de circuits et d'une pile (une petite batterie rechargeable ou une pile bouton au lithium) pour sauvegarder et rafraîchir l'heure et la date. Le Raspberry Pi n'en a pas, mais Raspbian est configuré pour synchroniser l'heure et la date automatiquement quand il est branché à un réseau, en communiquant avec un serveur NTP (Network Time Protocol, le protocole de synchronisation de l'heure).

Il est important que notre Raspberry Pi soit à l'heure, surtout pour certaines applications.
Pour régler l'heure et la date manuellement, utilisez le programme date :
$ sudo date --set="Sun Fab 14 1:55:16 EDT 2017".

II.6.2.7 Installation de nouveaux logiciels

Raspbian est une distribution plutôt minimaliste et vous aurez rapidement besoin de télécharger et d'installer de nouveaux programmes. La ligne de commande, est le moyen le plus flexible et le plus rapide pour installer un logiciel.

Le programme **apt-get** avec l'argument install va télécharger et installer des logiciels dont vous connaissez déjà le nom. apt-get va aussi télécharger les modules nécessaires à votre paquet pour que vous n'ayez pas besoin de chercher partout les dépendances. Les logiciels doivent être installés par l'administrateur, donc il faut toujours employer **sudo**. Par exemple, la commande suivante installe Tkinter (bibliothèque sous Python pour créer des interfaces graphiques).
pi@raspberrypi ~ $ sudo apt-get install python-tk [17].

II.7 Les broches d'entrées-sorties sur Raspberry Pi

Le Raspberry Pi a une double rangée de broches sur l'un de ses côtés. Ces broches, que l'on appelle connecteur GPIO (General Purpose Input/Output, entrées/ sorties à usage général), permettent de connecter du matériel électronique au Raspberry Pi sans passer par le port USB.

Les broches GPIO du Raspberry Pi vous permettent de contrôler d'autres composants électroniques ainsi que des interfaces telles que des LED, des moteurs et des relais. Ces diverses interfaces sont généralement regroupées sous le terme de « sorties ». Pour ce qui est des « entrées », notre Raspberry Pi peut lire et interpréter l'état de boutons, d'interrupteurs, de capteurs de température, de lumière, de mouvement ou de proximité, etc. la liste est très longue.

En utilisant les interfaces numériques asynchrones et synchrones (SPI et I2C), les applications potentielles sont encore plus nombreuses. Nous allons ici nous intéresser aux applications les plus élémentaires.

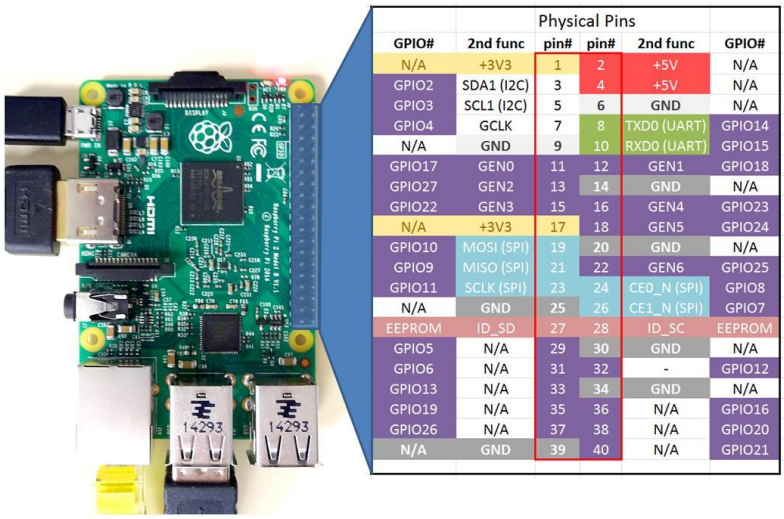


Figure II-39. Les broches GPIO du Raspberry Pi.

Le principe de fonctionnement des broches GPIO

– Un connecteur GPIO peut être configuré en entrée (réception d'un signal) ou sortie (émission d'un signal).
– Un connecteur GPIO ne peut traiter que des signaux numériques, 1 ou 0, une connexion sur un support analogique nécessitera un CAN/CNA.
– Un connecteur GPIO est généralement alimenté en 3.3 Vcc et ne peut émettre que des courants de faible intensité, allant de 3 mA à 50 mA.
– Certains connecteurs peuvent être reconfigurés en liaison série, bus **I2C** ou **SPI** :
Pour **I2C** (Inter-Integrated Circuit) est un bus de données. La connexion est réalisée par l'intermédiaire de deux lignes :
SDA (Serial Data Line) : ligne de données bidirectionnelle,
SCL (Serial Clock Line) : ligne d'horloge de synchronisation bidirectionnelle.
Il ne faut également pas oublier la masse qui doit être commune aux équipements.
Pour **SPI** (Serial Peripheral Interface) c'est un bus de données série synchrone en mode Full-duplex.

Les circuits communiquent selon un schéma maître-esclaves, où le maître s'occupe totalement de la communication.

Le bus SPI utilise quatre signaux logiques :
● SCLK — Serial Clock, Horloge (généré par le maître).
● MOSI — Master Output, Slave Input (généré par le maître).
● MISO — Master Input, Slave Output (généré par l'esclave).
● SS — Slave Select, Actif à l'état bas (généré par le maître).

Il existe un grand nombre de langages de programmation pour controller les broches d'entrées-sorties du Raspberry Pi. En effet, Python permet d'accéder aux broches d'entrées-sorties et d'automatiser la lecture et le contrôle des GPIO.

II.8 Programmation des GPIO en Python

II.8.1 Langage de programmation Python

Python est un langage interprété, ce qui veut dire que vous pouvez écrire un programme ou un script et l'exécuter directement sans le compiler en code machine. Les langages interprétés sont souvent un peu plus rapides à écrire que d'autres, et ils fournissent quelques avantages supplémentaires.

Avec Python, par exemple, il n'est pas nécessaire d'indiquer à votre ordinateur si une variable est un nombre, une liste ou une chaîne de caractères. L'interpréteur devine les types de données lors de l'exécution d'un script. L'interpréteur de Python est capable de fonctionner de deux façons différentes : comme un shell interactif pour exécuter les commandes individuelles, ou comme un programme en ligne de commande pour exécuter les scripts indépendamment. L'environnement intégré de développement (IDE) pour Python fourni avec le Raspberry Pi s'appelle **IDLE** (figure II-40).

Figure II-40. L'environnement de développement intégré IDLE pour Python.

Vous pouvez choisir n'importe quel éditeur de texte pour commencer à coder, ici nous utilisons **IDLE 3**. Pour le lancer, double-cliquez sur l'icône IDLE 3 après quelques secondes, la fenêtre d'IDLE apparaît avec son shell interactif.

Le triple chevron (>>>) correspond à l'invite de commande, ou prompt interactif. Quand le prompt est visible, cela signifie que l'interpréteur attend vos commandes.

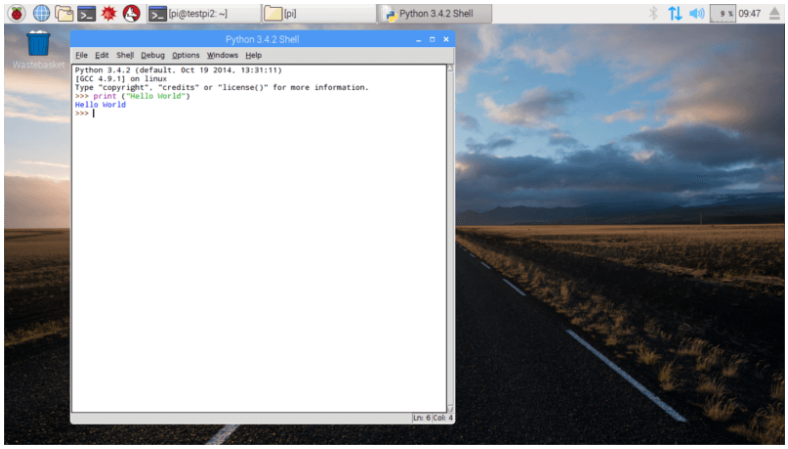


Figure II-41. Le shell interactif d'IDLE.

Saisissez le code suivant, appuyez ensuite sur la touche Entrée.

```
>>> print "Hello World"
Hello World
>>>
```

Python exécutera alors cette commande et vous verrez le résultat dans la fenêtre du shell.
Dans la mesure où Python est un langage interprété, vous pouvez utiliser le shell comme une calculatrice pour tester les déclarations ou les calculs :

```
>>> 1+1
2
```

```
>>> 2*2-1
3
```

Le code que vous saisissez dans le shell interactif est comme un programme que vous exécutez ligne par ligne. Vous pouvez définir vos variables ou importer des modules.
Pour créer vos variables, utilisez l'opérateur d'affectation (=) :

```
>>> x=2
>>> print x
2
```

Importer le module math par exemple :
La commande import math rend toutes les fonctions mathématiques de Python disponibles pour votre programme.

>>> import math
>>> (1 + math.sqrt(5)) / 2
1.618033988749895

L'interpréteur de Python est pratique pour tester des déclarations ou des opérations simples, mais vous aurez souvent besoin d'exécuter nos scripts Python comme des applications autonomes. Pour cela, il vous faut un éditeur plus classique et non un shell. Pour créer un nouveau programme Python, sélectionnez File>New Window afin d'ouvrir une nouvelle fenêtre d'édition de code (figure II-42).

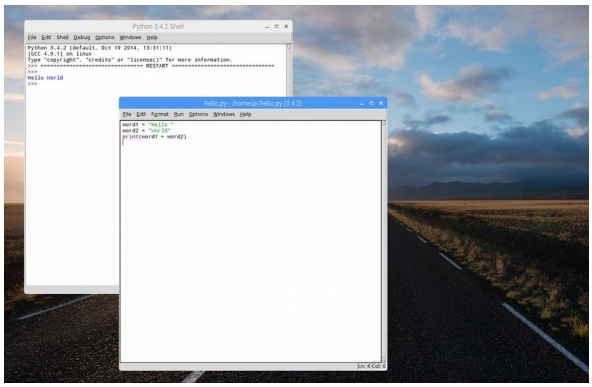


Figure II-42. Le shell interactif d'IDLE (à gauche) et une fenêtre de l'éditeur (à droite).

Si vous tapez une ligne de code et sélectionnez le menu Run>Run Module, un avertissement s'affichera : « Source Must Be Saved OK To Save? » (soit« Le code source doit être sauvegardé, voulez-vous le sauvegarder ? »).

Vous devez enregistrer votre script (par exemple, dans votre répertoire personnel /home/pi sous le nom BonjourPi.py) pour qu'il puisse s'exécuter dans le shell.

Parfois, vous aurez peut-être envie de travailler avec un autre environnement qu'IDLE. Pour exécuter un script en ligne de commande, ouvrez LXTerminal et saisissez le code suivant :

pi@raspberrypi ~ $ sudo python BonjourPi.py

II.8.2 Installation du module Python GPIO

Il y a de nombreux modules disponibles pour Python, et certains ont été créés spécifiquement pour le Raspberry Pi, comme la bibliothèque **RPi.GPIO** pour contrôler les broches GPIO.

Assurez-vous que le Raspberry Pi est correctement connecté à Internet, puis lancez les commandes suivantes pour mettre à jour la liste des packages et démarrer l'installation du module raspberry-gpio-python :

pi@raspberrypi ~ $ sudo apt-get update

pi@raspberrypi ~ $ sudo apt-get install python-rpi.gpio

Lancez l'interpréteur Python à partir d'un terminal en tant qu'utilisateur root. Puisque le module raspberry-gpio-python nécessite les droits du compte root pour lire et contrôler les broches, vous devez lancer l'interpréteur Python avec la commande sudo.

pi@raspberrypi ~ $ sudo python

Au prompt >>>, importez le module en tapant la commande suivante :

>>> import RPi.GPIO as GPIO

II.8.3 Faire clignoter une LED via interface graphique sous Python

Le moyen le plus facile d'utiliser les broches GPIO en sortie est de les connecter à des LED (diodes électroluminescentes).

Nous allons maintenant voir comment créer une simple interface graphique pour faire clignoter une LED à l'aide d'un module présent par défaut dans Python : **Tkinter**.

II.8.3.1 Présentation de Tkinter

Tkinter (**Tk interface**) est un module intégré à la bibliothèque standard de Python. Il offre un moyen de créer des interfaces graphiques via Python.

II.8.3.2 Montage

On va utiliser les éléments suivants pour faire allumer une LED.

• Une carte Raspberry Pi connecté à avec un téléviseur ou moniteur.

• Une plaque d'essai.

• Une résistance 1kΩ.

• Une LED.

• Des fils de connexion.

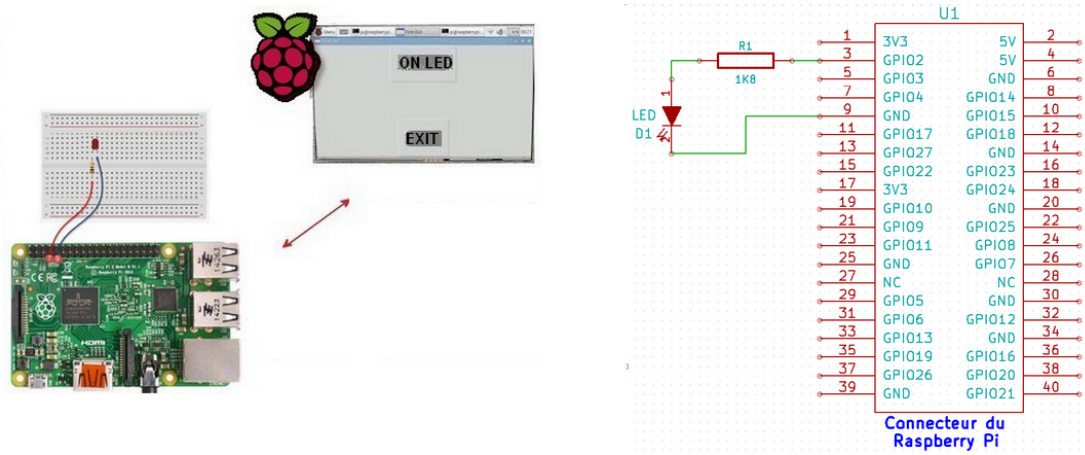


Figure II-43. Commander LED avec simple GUI python (à gauche) et circuit éléctronique montage (à droite).

II.8.3.3 Code en Python

1.Ouvrez le gestionnaire de fichiers (File Manager) en cliquant sur son icône dans la barre de menus.
2.Assurez-vous que vous êtes bien dans votre répertoire personnel, qui se trouve par défaut dans /home/pi. Si ce n'est pas le cas, cliquez sur l'icône représentant une maison.
3.Créez un fichier vide dans votre répertoire /home/pi. Pour ce faire, effectuez un clic droit dans la fenêtre du répertoire personnel, sélectionnez Create New..., puis double-cliquez sur Blank File.
Nommez ce nouveau fichier blinkpin11.py.

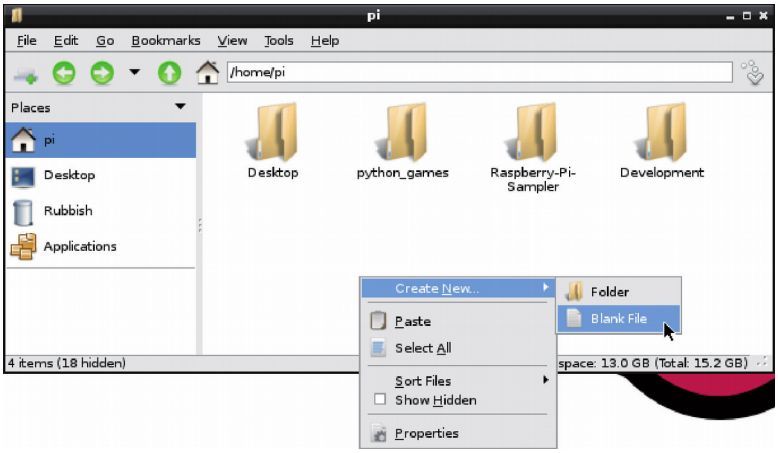


Figure II-44. Création d'un nouveau fichier dans le répertoire home.

4. Double-cliquez sur ce fichier pour l'ouvrir avec Leafpad, l'éditeur de texte par défaut en mode graphique.
5. Saisissez le code source suivant dans l'éditeur et enregistrez le fichier [18] :

```
from Tkinter import *
import tkFont
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.OUT)
GPIO.output(11, GPIO.LOW)

interface = Tk()

notreFont = tkFont.Font(family = 'Helvetica', size = 36, weight = 'bold')

def On_Off_LED():
        print("button LED pressed")
        if GPIO.input(11) :
                GPIO.output(11,GPIO.LOW)
                Buttonled["text"] = "ON LED"
        else:
                GPIO.output(11,GPIO.HIGH)
                Buttonled["text"] = "OFF LED"

def interface_exit():
        print("Button EXIT pressed")
        GPIO.cleanup()
        interface.quit()

interface.title("Allumer LED")
interface.geometry('800x480')

Buttonexit = Button(interface, text = "Exit", font = notreFont, command = interface_exit, height =2 , width = 6)
Buttonexit.pack(side = BOTTOM)

Buttonled = Button(interface, text = "ON LED", font = notreFont, command = On_Off_LED, height = 2, width =8 )
Buttonled.pack()

mainloop()
```

Explication du code

❶ **import Tkinter et tkFont** : on commence par importer **Tkinter**, et par la suite la bibliothèque **tkFont** qui a pour but de spécifier notre police de caractère des bouttons.
❷ **import RPi.GPIO as GPIO** : importe le code nécessaire pour le contrôle des GPIO.
❸ **GPIO.setmode(GPIO.BOARD)** : utilise la numérotation des broches de la carte Raspberry Pi.
❹ **GPIO.setup(11, GPIO.OUT)** : configure la broche 11 en sortie.
❺ **interface= Tk()** : on crée ensuite un objet de la classe **Tk**. La plupart du temps, cet objet sera la fenêtre principale de notre interface.
❻ **def On_Off_led()** : on crée une méthode On_Off_led, qui est appelée quand on clique sur le bouton ON LED ou OFF LED. Elle ne prend aucun paramètre.
GPIO.output(11, GPIO.LOW) : pour éteindre la LED.

GPIO.output(11,GPIO.HIGH) : pour allumer la LED.
❼ **interface.title ("Allumer LED")** : pour donner le titre de l'interface.
interface.geometry('800x480') : indique les dimensions de notre interface en longeur et en largeur.
❽ On appelle la méthode **pack** de notre Buttonexit qui permet de positionner l'objet dans notre fenêtre et, par conséquent, de l'afficher. En précisant side="Bottom", on demande à ce que le widget Buttonexit soit placé en bas de la fenêtre.
Le dernier argument qui est important ici a pour nom **command** et a pour valeur la méthode interface_exit de notre fenêtre. Il s'agit de l'action liée à un clic sur le bouton. Ici, c'est la méthode interface_exit de notre fenêtre racine qui est appelée.
Ainsi, quand vous cliquez sur le bouton EXIT, la fenêtre se ferme.

Même principe pour le widget Buttonled, mais si nous cliquons sur ce bouton, par exemple ON LED, on remarque que la LED clignote, et le bouton change son libelé et devient OFF LED pour éteindre cette LED.

❾ Enfin, on appelle la méthode **mainloop()** de notre fenêtre racine. Cette méthode ne retourne que lorsqu'on ferme la fenêtre.

6.Ouvrez LXTerminal et lancez les commandes suivantes pour vous assurer que le répertoire de travail est bien /home/pi. Exécutez ensuite le script :

pi@raspberrypi ~ $ cd ~

```
pi@raspberrypi ~ $ sudo python blinkpin11.py
```

7. L'interface graphique apparu, et le programme fonctionne correctement.

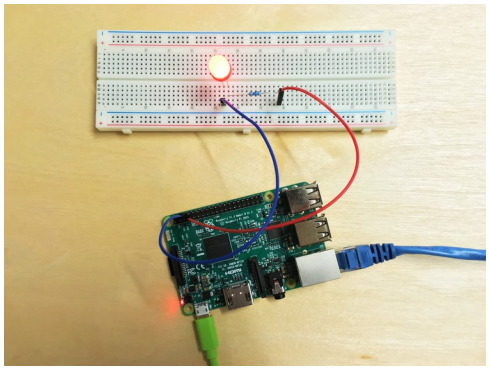


Figure II-46. Allumer et éteindre une LED via une interface graphique Python.

II.9 Conclusion

Comme nous venons de le voir, le Raspberry Pi est un mini PC qui possède des entrées/sorties (GPIO) puissantes permettant une connexion avec le monde physique par l'intermédiaire de capteurs et d'actionneurs. Le Raspberry Pi constitue une liaison concrète entre le monde de l'électronique et celui de l'informatique, le port GPIO est un outil fort pratique, et simple à interfacer grâce au langage Python.

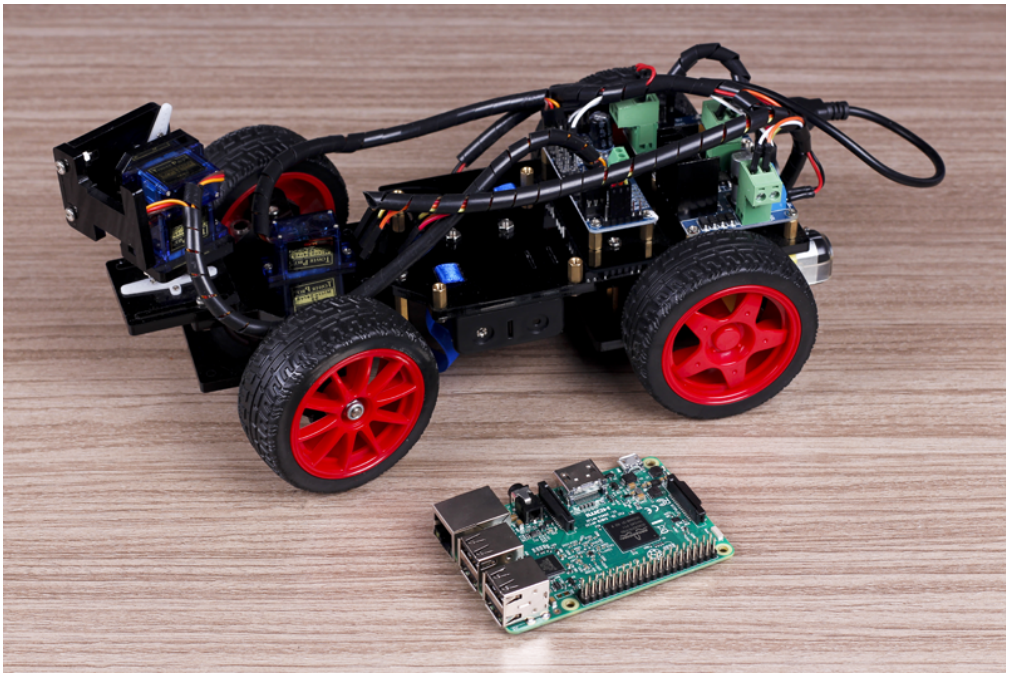
Il fait ainsi du Raspberry Pi une cible idéale à embarquer et programmer en Python afin de réaliser :

• de la domotique (station météo, pilotage de volets roulants, centrale d'alarme…).
• de la robotique (robot suiveur, bras motorisé…).

Maintenant que nous avons installé notre Raspberry Pi et qu'il est prêt à l'emploi, nous pouvons commencer à explorer ses caractéristiques dans le domaine de la robotique.

Chapitre III

Commande à distance d'un robot mobile



III.1 Introduction

La robotique est au service de l'homme dans de nombreux domaines tels que l'industrie, médecine, domotique, militaire, etc.

Dans ce chapitre, nous allons apprendre à utiliser et programmer notre Raspberry Pi dans le cadre d'un environnement robotique. Dans un premier temps, nous allons voir une simple présentation concernant les composants du robot, puis dans un second temps, nous allons voir en détails la réalisation électronique, et enfin nous présenterons les différentes procédures, programmes et les explications pour contrôler le robot via une application cliente desktop, ensuite par une application Android qui constitue l'objectif de notre travail à travers le développement d'un site web sous le framework Django, et de plus via une page web PHP depuis un navigateur.

L'acquisition et l'échange de données seront faites par un protocole de communication sans fil WiFi. Cela se fait en embarquant au robot une webcam capable de faire la transmission vidéo en temps réel, et permet aussi de prendre des images de haute qualité en utilisant la bibliothèque OpenCV.

III.2 Description du kit de robot

Cette partie ne fut que très rapidement abordée. En effet, au vu des composants disponibles, le châssis est livré en pièces détachées qui doivent être assemblées. Des instructions de montage **détaillées** sont incluses dans le kit, celui conçu par la société **Sunfounder** (figure III.1).

La plate-forme utilisée est très simple capable de porter les composants du système, composée de deux moteurs à courant continu (CC), quatre roues, trois servomoteurs qui jouent le rôle des actionneurs, d'autres modules que nous allons voir par la suite, et toutes les pièces nécessaires pour assembler le châssis. Il comprend également un boîtier où peuvent se loger deux piles. Ce qui signifie que l'ensemble du projet peut être réalisé sans aucune soudure (l'outil le plus difficile à utiliser ne sera donc qu'un tournevis).

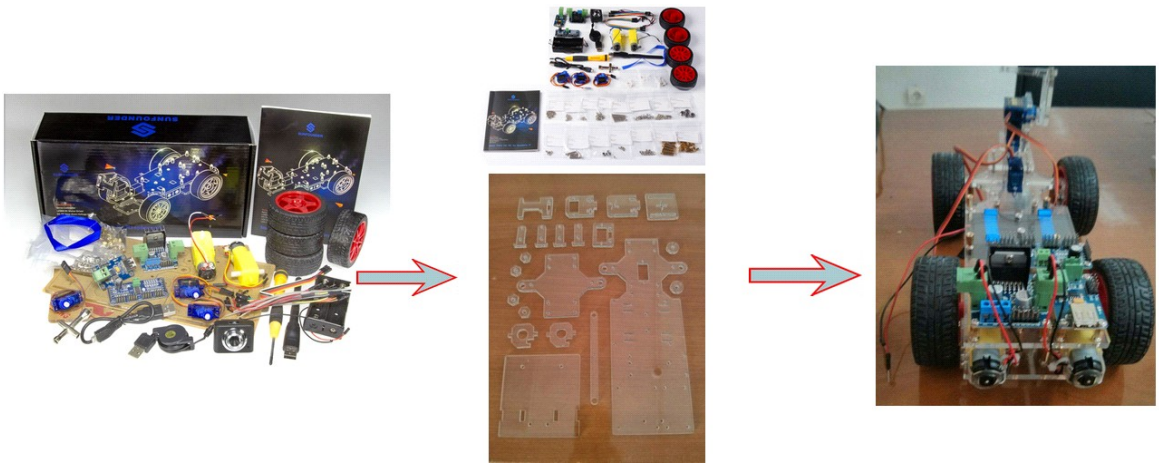


Figure III-1. Le robot avant et après l'assemblage.

III.3 Partie électronique

La partie électronique a pris une place très importante dans notre projet car elle fait le lien entre la commande informatique et les composants en lien direct avec l'extérieur (moteurs CC, servomoteurs...).

Tout d'abord, le cerveau du robot est le Raspberry Pi qui est fixè à l'aide de vis et boulons au milieu de la plate-forme (châssis).

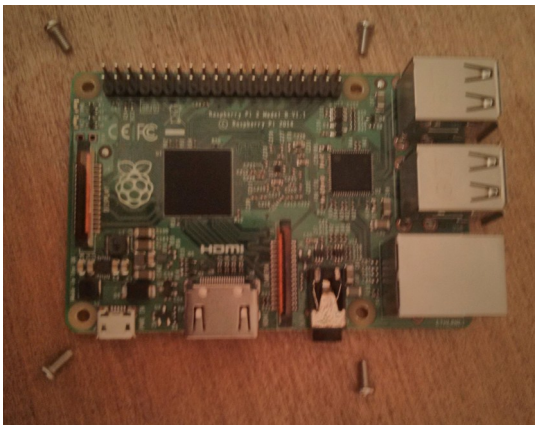


Figure III-2. Raspberry Pi 2 modèle B.

Dans cette partie, nous détaillons les montages électriques que nous avons pu réaliser étape par étape ainsi que le rôle de chaque module électronique utilisé pour bien apprendre comment le robot fonctionne.

III.3.1 Première étape

Pour assurer le déplacement du robot, on a utilisé deux moteurs CC.
Dans notre projet, on se limite au contrôle des moteurs à courant continu les plus ordinaires.

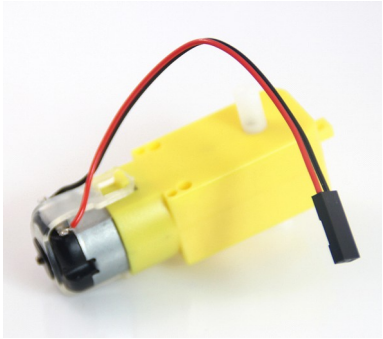


Figure III-3. Moteur à courant continu.

Le module L298N à été choisi pour commander les moteurs. Pour cela, deux techniques sont utilisées :
- La commande du sens de rotation par un pont en H.
- La commande de puissance (et donc de la vitesse) par la modulation d'amplitude du signal (PWM ou Pulse Width Modulation).

III.3.1.1 Le pont en H

Le pont en H est une structure électronique servant à contrôler la polarité aux bornes d'un dipôle.

Il est composé de quatre éléments de commutation généralement disposés schématiquement en une forme de H comme l'indique la figure III-4. Les commutateurs peuvent être des relais, des transistors, ou autres éléments de commutation en fonction de l'application visée [19].

a) Principe de fonctionnement du pont en H

Les interrupteurs fonctionnent deux par deux. Le A est associé au D et le B est associé au C. Dans le schéma ci-dessous, rien ne se passe car tous les interrupteurs sont ouverts (ils ne laissent pas passer le courant), donc le moteur est arrêté.

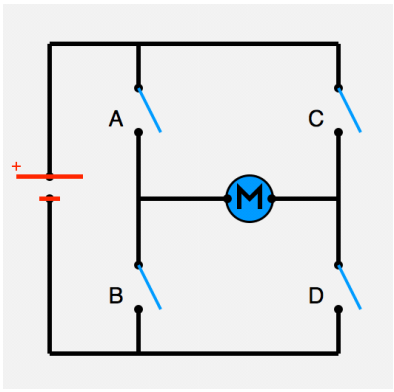


Figure III-4. Le pont en H.

Voyons maintenant ce qui arrive lorsqu'on actionne en même temps les interrupteurs A et D (schéma de gauche), ou les interrupteurs B et C (schéma de droite) selon la figure III-5.
Sur le schéma de gauche, les interrupteurs A et D sont fermés, donc le courant entre par la patte gauche du moteur et sort par sa droite et par conséquent le moteur tourne.
Sur le schéma de droite, les interrupteurs B et C sont fermés, donc le courant entre par la patte droite du moteur et sort par sa gauche ce qui fait tourner le moteur dans le sens inverse.

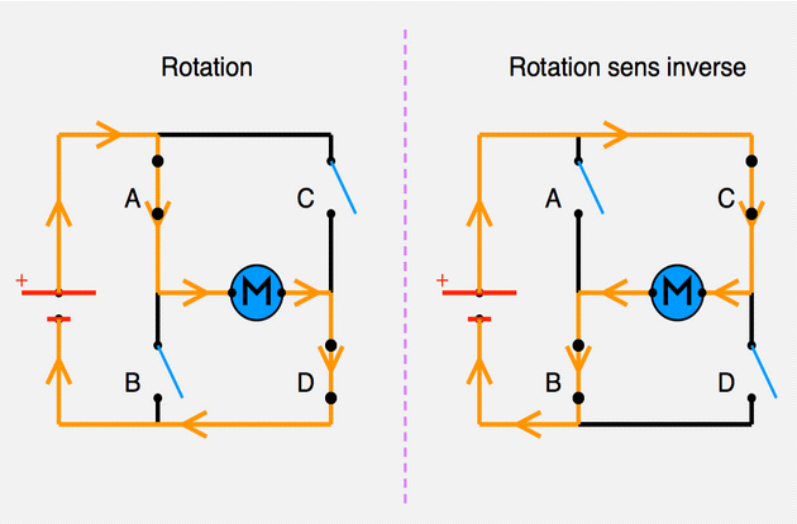


Figure III-5. Sens du courant en fonction de l'état des interrupteurs d'un pont en H.

b) La technique PWM

PWM est l'acronyme anglais de "Pulse Width Modulation" que l'on traduit par "modulation par largeur d'impulsions" qui est une technique utilisée pour contrôler la puissance envoyée à un périphérique. Nous l'utiliserons pour contrôler la quantité d'énergie alimentant le moteur et par conséquent sa vitesse de rotation.

Exemple d'utilisation

Le graphique ci-dessous montre le signal PWM tel qu'il est envoyé par la broche PWM du Raspberry Pi.

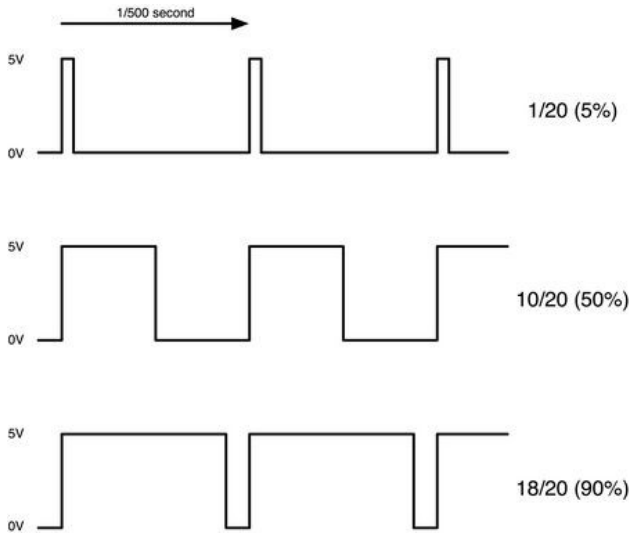


Figure III-6. Signal PWM envoyé par la broche PWM du Raspberry Pi.

A chaque 1/500 ième de seconde, la sortie PWM produit une impulsion. La longueur de cette impulsion (dans le temps) contrôle la quantité d'énergie qui alimente le moteur. Comme vous pouvez le constater sur le graphique, la longueur de l'impulsion peut varier de 0 à 100%.

Sans impulsion, le moteur ne fonctionne pas, une courte impulsion le fera tourner lentement. Si l'impulsion est active pendant cinquante pour cent du cycle, le moteur recevra la moitié de la puissance qu'il recevrait avec des impulsions maximales (constante dans le temps) [20].

III.3.1.2 Présentation du module L298N

Le L298N intègre deux ponts en H de 2A maximum (avec dissipateur). Ils doivent être protégés contre les courants inverse grâce à des diodes de roues libres. Il dispose de deux broches d'activation qui permettent de désactiver l'alimentation de chaque pont en H.

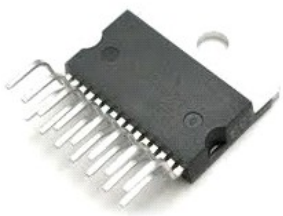


Figure III-7. Circuit intégré L298N.

Ce composant a besoin de deux alimentations pour fonctionner, l'alimentation (+VSS) jusqu'à 7V utilisée pour alimenter les portes logiques, et l'alimentation (+Vs) jusqu'à 46V utilisée pour alimenter les moteurs.

Nous utilisons un module déjà monté, il se compose du circuit intégré L298N avec dissipateur, d'un régulateur 5V (pour la partie logique), des diodes de protection (diodes de roue libre) contre les courants inverses, de deux connecteurs pour le branchement des moteurs.

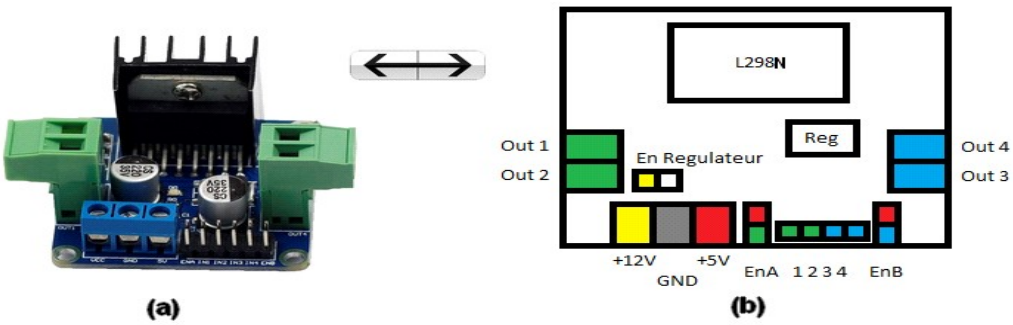


Figure III-8. Module L298N (a) et leur broches de connexion (b).

III.3.1.3 Description du module L298N

En se basant sur la figure III-8 (b).
- En couleur vert : pont en H (A).
  La sortie Out1 est commandée par l'entrée 1 (IN1).
  La sortie Out2 est commandée par l'entrée 2 (IN2).
- En couleur Bleu : pont en H (B).
  La sortie Out3 est commandée par l'entrée 3 (IN3).
  La sortie Out4 est commandée par l'entrée 4 (IN4).
- En jaune : 12V = Vs du L298N.
- En gris : GND.
- En Rouge : 5 V = VSS du L298N.

Le jumper en régulateur permet d'alimenter le régulateur afin de créer le signal VSS nécessaire au L298N grâce au 12V. Si le jumper est mis il suffit d'alimenter +12 et GND.
Les deux jumper EnA et EnB permettent d'activer les ponts en H [23].
Le montage électrique réalisé entre le module L298N et les moteurs CC est apparu comme suit :

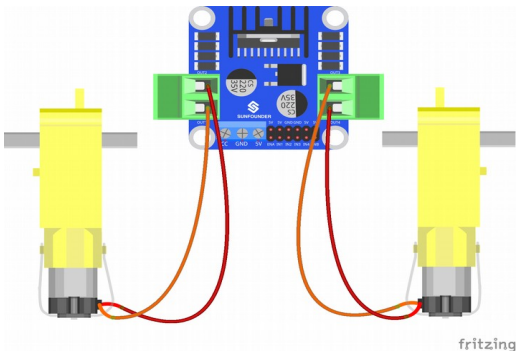


Figure III-9. Module L298N Commande deux moteurs CC séparés.

III.3.2 Deuxième étape

Cette étape est basée sur le tableau III-1, consiste à connecter le module L298N avec les broches 11, 12, 13 et 15 du Raspberry Pi.

GPIO du Raspberry Pi	Module L298N
Pin11	IN1
Pin12	IN2
Pin13	IN3
Pin15	IN4

Tableau III-1. Le branchement du module L298N avec les GPIO (11, 12, 13, 15) du Raspberry Pi.

Les entrées IN1, IN2, IN3 et IN4 du module L298N sont utilisées pour commander le sens de la rotation du moteur 1 et du moteur 2 respectivement.
Schéma de branchement :

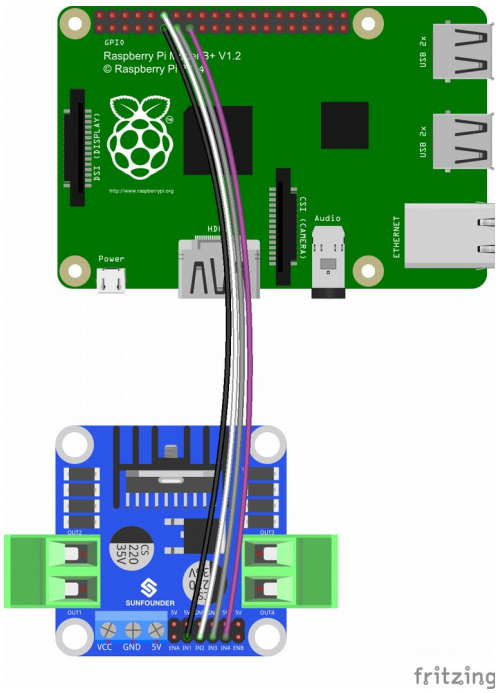


Figure III-10. Raccordement électrique du module L298N avec les GPIO du Raspberry Pi.

III.3.3 Troisième étape

Il est vraiment pratique dans notre cas pour piloter des sorties PWM comme les servomoteurs d'utiliser un module puissant et très utilisé dans le domaine de la robotique qui est le **contrôleur servo-PWM**.

Ce module permet de contrôler jusqu'à 16 servomoteurs (ou LEDs) en PWM, il peut être monté en série pour obtenir jusqu'à 62 x 16 canaux avec seulement 2 fils d'interface.

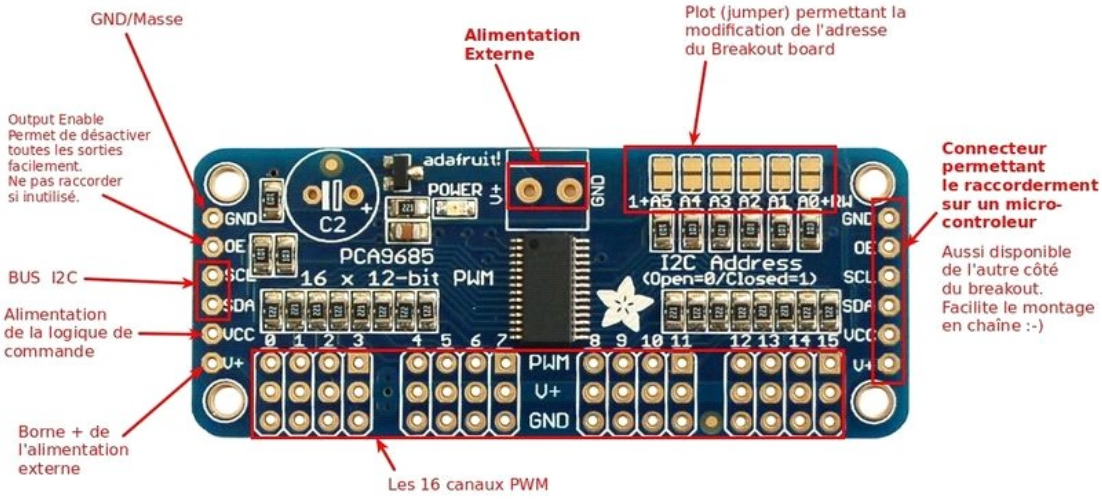


Figure III-11. Contrôleur servo-PWM et ses interfaces.

Remarque :

Ce module est aussi connu sous son appellation d'origine "16-Channel 12-bit PWM/Servo Driver - I2C interface - PCA9685".

III.3.3.1 Connecteur de commande du contrôleur servo-PWM

Le connecteur de commande comporte les broches suivantes :

Module PWM/Servo	Raspberry Pi
GND	GND/Masse.
OE	Permet de désactiver toutes les sorties PWM à l'aide d'un seul signal.
SDA	S'utilise avec SDA pour établir une communication I2C avec le Raspberry Pi.
SCL	S'utilise avec SCL pour établir une communication I2C avec le Raspberry Pi.
V+	Correspond à l'entrée V+ du bornier d'alimentation disponible sur le module.

Tableau III-2. Tableau illustre les connecteurs de commande du contrôleur servo PWM.

III.3.3.2 Alimentation externe du contrôleur servo-PWM

Le bornier bleu sur le module permet de raccorder une source d'alimentation externe utilisée pour fournir la puissance nécessaire à nos servomoteurs.

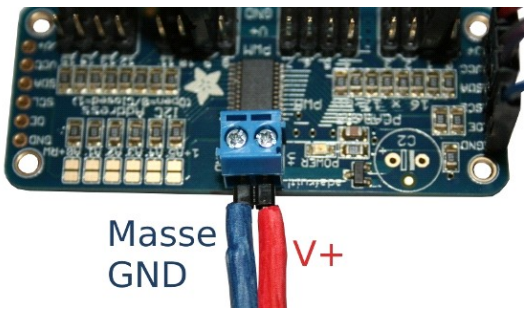


Figure III-12. Alimentation externe du contrôleur servo-PWM.

Selon la fiche produite, vous pouvez y raccorder une source de tension allant jusqu'à 6 volts (ce qui est le maximum en usage pour les servomoteurs) [21].

Le module contrôleur servo-PWM se raccorde sur un Raspberry Pi comme indiqué sur le tableau suivant :

GPIO du Raspberry Pi	contrôleur PWM
Pin 2	VCC
Pin 3	SDA
Pin 5	SCL
Pin 6 (GND)	GND

Tableau III-3. Tableau indique le raccordement entre les pins du contrôleur PWM et les GPIO du Raspberry Pi.

Le montage suivant montre ce raccordement en image :

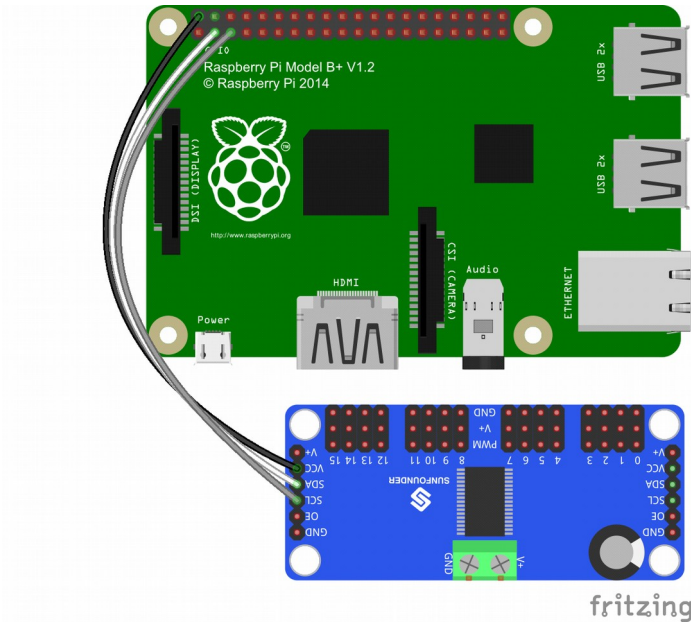


Figure III-13. Raccordement du module contrôleur servo-PWM sur le Raspberry Pi.

III.3.4 Quatrième étape

L'image ci-dessous montre comment brancher correctement le module L298N avec le module contrôleur servo-PMW.

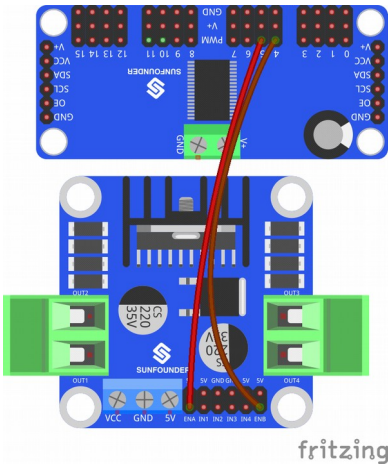


Figure III-14. Schéma de branchement du module L298N avec contrôleur servo-PMW.

Les broches ENA et ENB sont directement reliées aux sorties (PWM) du contrôleur servo-PWM pour commander la vitesse du robot par un signal PWM.

III.3.5 Cinquième étape

Les servomoteurs sont très utilisés en robotique car ils sont à la base de nombreux mécanisme d'actionneurs, ils consomment souvent beaucoup de courant et demandent une tension souvent très bien régulée ce qui nécessite des dispositions particulières.

Dans notre cas, on utilise les servomoteurs pour diriger le robot dans les deux sens gauche et droite comme indique la figure III-15 et rendant positionner le webcam selon les deux axes X et Y.

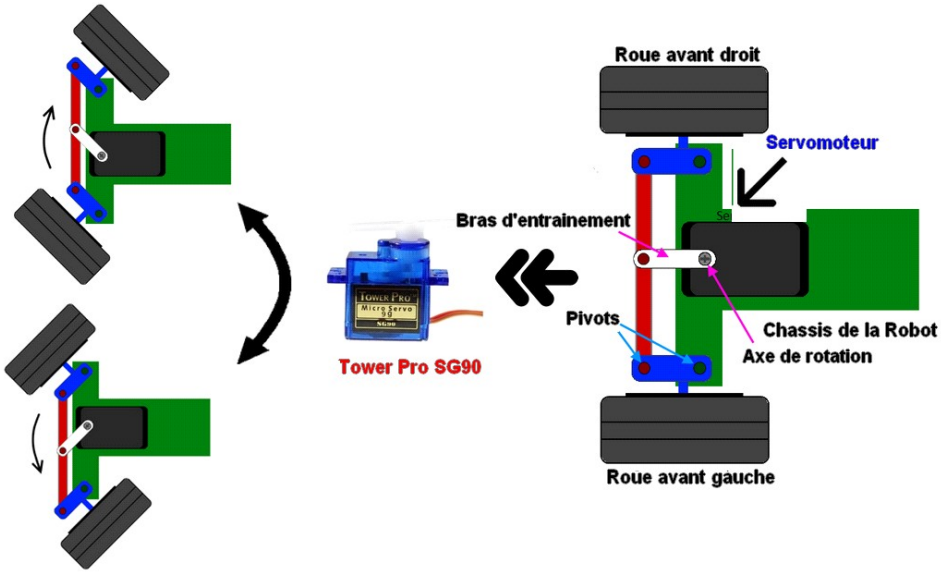


Figure III-15. Servomoteur modèle Tower Pro SG90 et son utilisation pour diriger le robot.

Le servomoteur comprend les éléments suivants :

• Un moteur électrique de petite taille.
• Un réducteur en sortie de ce moteur pour avoir moins de vitesse et plus de couple.

• Un capteur : un potentiomètre qui produit une tension variable en fonction de l'angle de l'axe de sortie.
• Un asservissement électronique pour contrôler la position de cet axe de sortie.

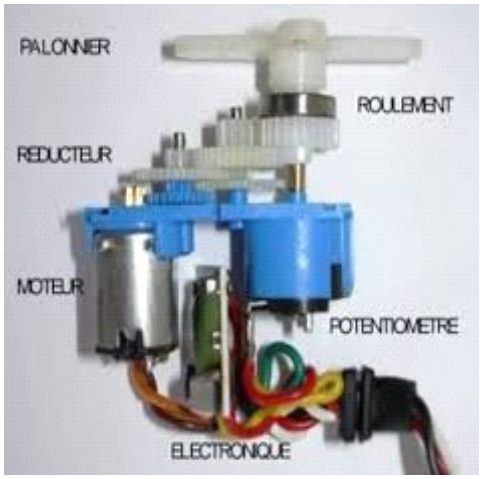


Figure III-16. Les composantes d'un servomoteur.

Un servomoteur a la capacité de tourner d'un angle de 0° à un angle de 180°.

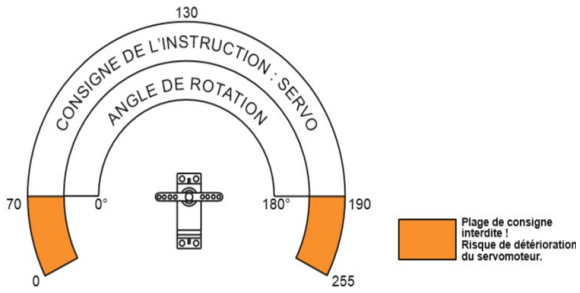


Figure III-17. Les angles de rotation d'un servomoteur.

Le servomoteur est équipé d'une prise de type Graupner à 3 fils :

Marron	masse
Rouge	+ 5v
Orange	commande

Tableau III-4. La correspondance des trois fils d'un servomoteur.

Ce type de servomoteur peut être branché directement sur un contrôleur servo-PMW. Le branchement pourra être le suivant :

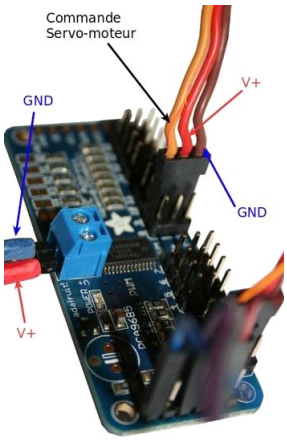


Figure III-18. Servomoteur branché sur un contrôleur servo-PMW.

Le tableau suivant indique le mode de branchement entre les servmoteurs et contrôleur servo-PMW.

Servomoteurs	Contrôleur servo-PMW
Bottom Servo	CH14
Top Servo	CH15
Steering Servo	CH0

Tableau III-5. Raccordement entre les servmoteurs et les cannaux du contrôleur servo-PMW.

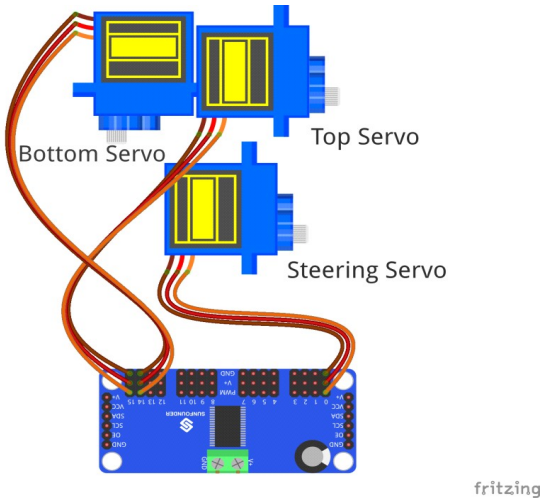


Figure III-19. Montage de branchement des servomoteurs avec contrôleur servo-PMW.

Principe de commande des servomoteurs

Les servomoteurs sont très souvent pilotés par un signal PWM.
Le fonctionnement des signaux PWM est assez simple, nous allons envoyer des impulsions de taille variable qui vont correspondre à un certain angle. La durée d'un signal créneau (l'impulsion) doit être comprise entre 1 et 2 ms qui se traduit par une valeur d'angle (0 et 180°).

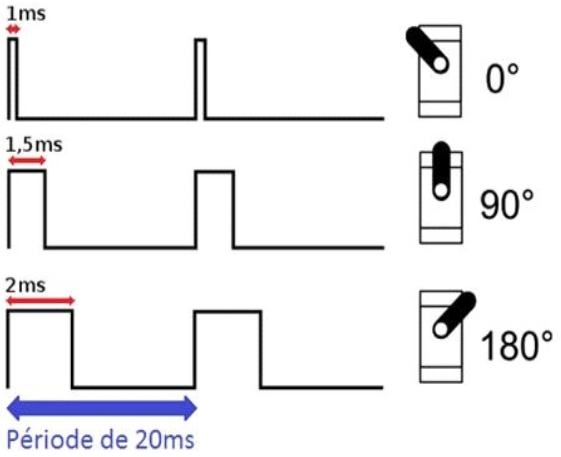


Figure III-20. Le pilotage d'un servomoteur par des impulsions PMW.

Comme on peut le voir sur la figure ci-dessus, la durée entre deux impulsions doit être de 20 ms maximum (ce qui correspond à une fréquence de 50Hz, F = 1 / T = 1 / 0.02 = 50Hz).

III.3.6 Sixième étape

Nous parlons dans cette partie sur le convertisseur de tension qui permet de transformer un courant continu (CC) venant d'une batterie de 7,4V en une autre tension CC réglable 5V pour alimenter le système électronique du robot. La puce a une tension d'entrée allant de 4,5 V à 40 V et génère une tension de sortie de 5V avec un courant aussi élevé que 2A.

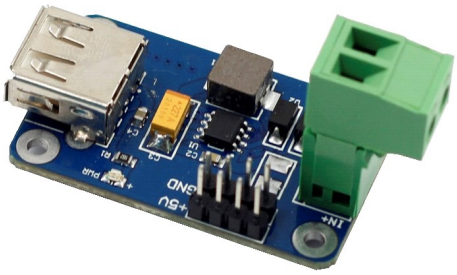


Figure III-21. Le module convertisseur de tension CC-CC.

Voici le montage qui illustre le raccordement entre le module contrôleur servo-PMW et le convertisseur :

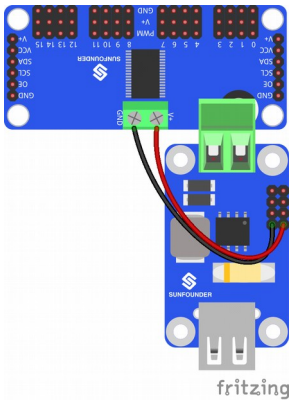


Figure III-22. Schéma de branchement du module contrôleur servo-PMW avec le convertisseur de tension.

III.3.7 Septième étape

Pour que notre système électronique fonctionne, on a besoin de l'energie. Pour cela, nous allons utiliser une alimentation autonome qui est deux piles de type Li-ion 18650 3.7 V.

Figure III-23. Batteries du robot.

La borne positive de l'alimentation (fil en rouge) doit être reliée à la prise droite du convertisseur, et la borne négative (fil en noir) doit être reliée à la prise gauche. Puis, on branche les deux fils (GND et VCC) insérés dans le module L298N dans les deux prises du convertisseur, un dans chacun, VCC avec la borne positive de l'alimentation et GND avec la borne négative. Ceci est indiqué sur le montage de la figure suivante :

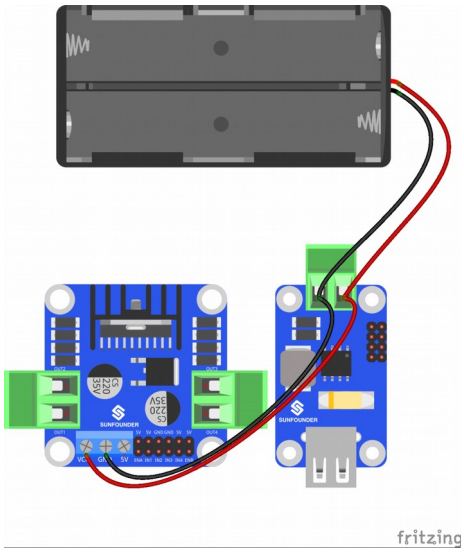


Figure III-24. Schéma de branchement du module convertisseur de tension avec le support de batterie et le module L298N.

L'ensemble du câblage final doit être comme ceci :

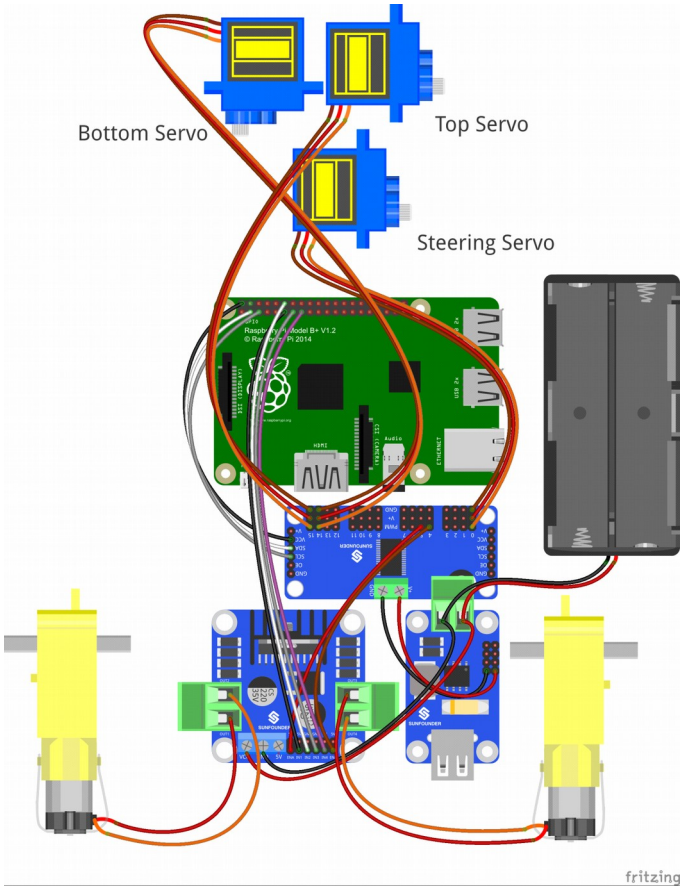


Figure III-25. Schéma de branchement final des circuits électroniques du robot.

Enfin, nous allons intégrer la webcam, USB WiFi adaptateur au Raspberry Pi de notre robot. L'ensemble du robot est apparu comme indiqué sur la figure ci-dessous :

Figure III-26. Assemblage complet du robot.

Nous avons parlé de la partie électronique ici, il faut s'attaquer maintenant à la partie informatique.

III.3 Partie informatique

Le principal objectif de la partie informatique était de pouvoir implémenter au sein du Raspberry Pi les programmes qui par la suite permettrait au robot de pouvoir agir comme nous le souhaitions.

III.3.1 Connexion à distance du robot via SSH

Pour pouvoir se connecter à distance avec notre robot, il faut le connecter à un réseau local et faire appel au protocole SSH.

Activation du SSH et utilisation de PuTTY

Secure Shell SSH est un protocole de réseau crypté pour initier des sessions shell textuelles sur des machines distantes de manière sécurisée. Cela permet à un utilisateur d'exécuter des commandes sur l'invite de commande d'une machine sans qu'il soit physiquement présent à proximité de la machine.

Pour pouvoir se connecter via notre ordinateur, il faut activer «ssh» sur le Raspberry Pi. Pour cela, on ouvre le terminal pui on entre la commande suivante pour démarrer raspi-config :
sudo raspi-config.

Une fois lancé, il faut aller sur «Advanced Options» et valider :

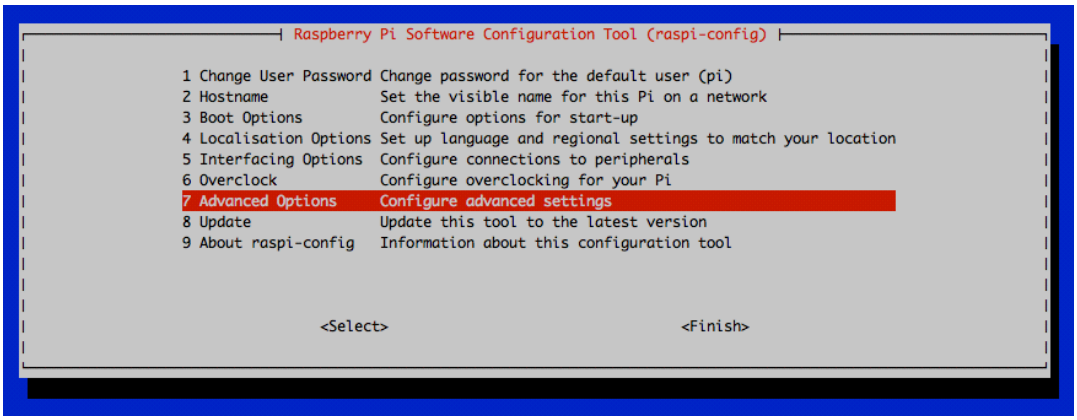


Figure III-27. Advanced Options de l'outil raspi-config.

Ensuite, on utilise la flèche de direction du clavier bas jusqu'à l'option ssh et on valide avec la touche entrée.

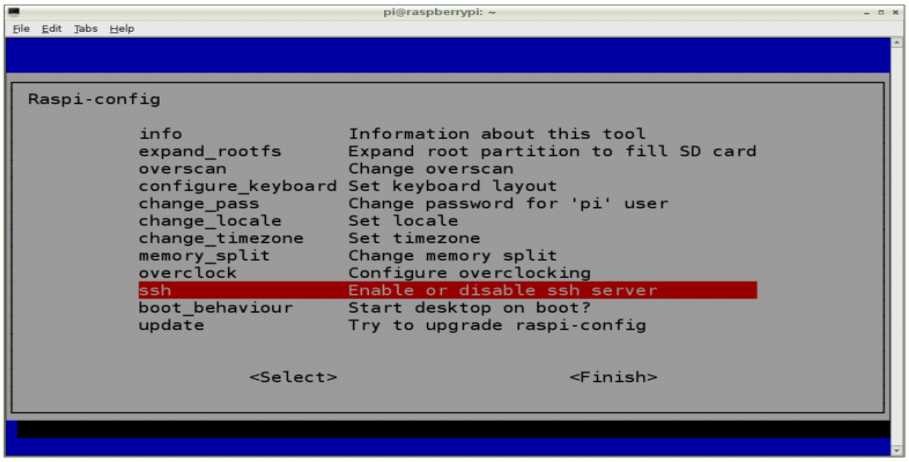


Figure III-28. L'option ssh de l'outil Raspi-config.

Après avoir choisi «Enable/Disable ssh», il faut demander à l'activer (Enable) :

Figure III-29. L'activation d'une connexion SSH.

Après validation, ssh est activé, il ne reste plus qu'à installer un logiciel permettant de l'utiliser sur l'ordinateur sous Windows. Ce programme s'appelle PuTTY. Le paramétrage se fait de la façon suivante (connexion SSH, port 22) :

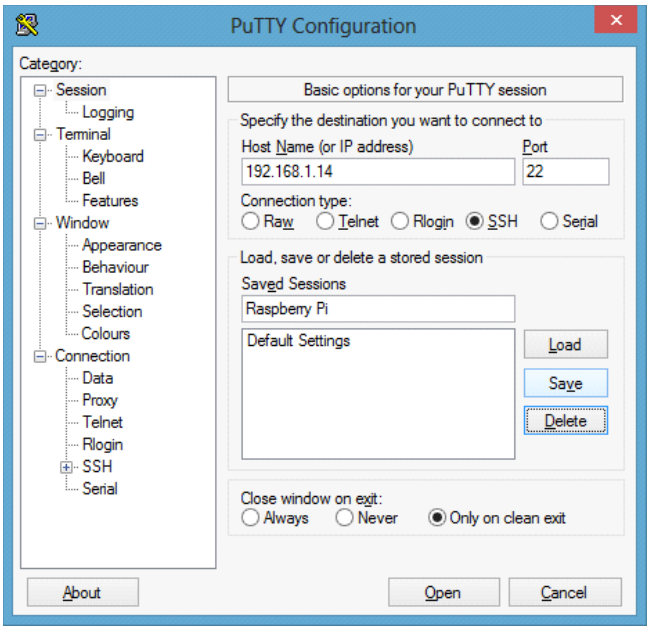


Figure III-30. Configuration de PuTTY pour établir une connexion SSH.

III.3.2 Communication entre les différentes modules du robot et le Raspberry Pi

C'est bien beau d'avoir utiliser les servomoteurs, contrôleur servo-PMW...etc, mais c'est un peu inutile si l'on n'est pas capable de communiquer les informations acquises au reste du robot, depuis d'un module à un autre, ou un composant au microprocesseur, ou aussi depuis le robot vers un ordinateur, il faut pouvoir communiquer les informations. Pour cela, il faut tout d'abord configuré le protocole I2C pour que notre circuit fonctionne correctement.

III.3.2.1 Configuration du protocole I2C sur le Raspberry Pi

Le bus I2C (ou I²C), pour Inter Integrated Circuit, a été développé en 1982 par Philips et permet de connecter divers équipements électroniques entre eux. Il dispose d'un système d'adressage permettant de connecter de nombreux périphériques I2C sur les mêmes câbles, ce qui signifie ici que l'on utilisera un nombre restreint de GPIO, quelque soit le nombre de périphériques I2C installés.

La première chose à faire sera de se connecter en ssh sur le Raspberry Pi pour faire les installations nécessaires depuis le terminal.

Installation des outils I2C

Nous allons maintenant installer des outils qui nous permettront de vérifier le bon fonctionnement de nos périphériques I2C, via les commandes suivantes :

sudo apt-get install python-smbus.

sudo apt-get install i2c-tools.

III.3.2.2 Installation du module noyau I2C

Par défaut, le support de l'I2C n'est pas activé dans le noyau Linux de Raspbian. Pour pouvoir le faire, comme précédemment nous allons utiliser l'utilitaire raspi-config, que l'on lancera en tapant la commande : **sudo raspi-config**

Cela lancera un menu, il faudra donc faire bas jusqu'à arriver sur le menu Advanced Options et appuyer sur entrée.
Cela ouvrira un nouveau menu, dans lequel on sélectionnera l'option I2C et confirmer avec entrée :

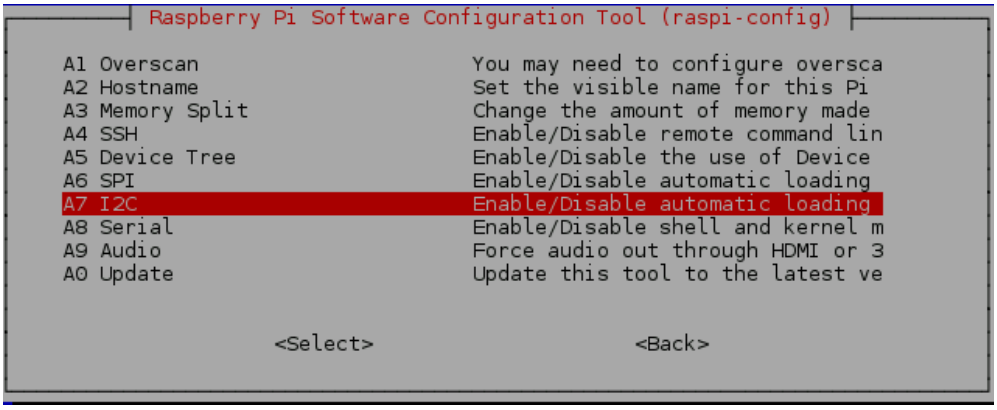


Figure III-31. L'option I2C après le choix Advanced Options.

Cela ouvrira une boite de dialogue demandant si l'on veut activer l'I2C, il faut répondre Oui :

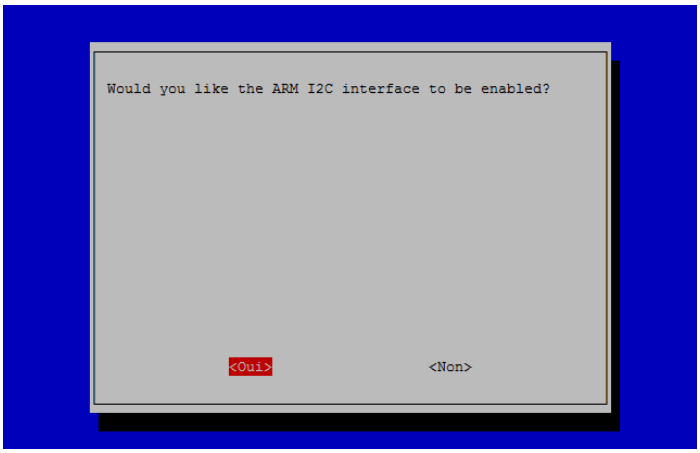


Figure III-32. Boite de dialogue demandant si l'on veut activer l'I2C.

Dès lors, une boite de dialogue devrait vous proposer de redémarrer le Raspberry Pi, ce à quoi il faut répondre oui. Si ce n'est pas le cas, il suffira de redémarrer en tapant la commande suivante : **sudo reboot**.

Notre projet consiste à commander notre robot à distance depuis un réseau local. Pour ce fait, on a besoin d'une communication sans fil afin de transmettre les ordres de pilotage de robot vers la carte Raspberry Pi qui s'occupe dans la suite par le déplacement du robot.

III.3.3 Communication sans fil

Dans cette partie importante, on décrira la communication sans fil qui utilise les ondes (signaux) électromagnétiques pour transmettre des données en utilisant l'air comme canal de transfert. Elle consiste à envoyer ces signaux via une bande de fréquence déterminée, et selon des protocoles déterminés WiFi, Bluetooth, ZigBee...ect. Le choix de la technologie WiFi est convenable pour notre situation.

III.3.3.1 Configuration du WiFi sur le Raspberry Pi via la ligne de commande

La technologie WiFi (correspondant à la famille des standards IEEE 802.11) qui équipe aujourd'hui tous les nouveaux smartphones, et est utilisée principalement par les réseaux locaux sans fil. Le WiFi permet l'accès à Internet avec des vitesses de transfert atteignant facilement des dizaines de méga bits par seconde. Il est caractérisé par une consommation relativement importante, et donc une autonomie faible pour les équipements qui l'embarquent.

Figure III-33. Logo de la technologie WiFi.

Afin de contrôler le robot via le WiFi, nous avons utilisé le module USB WiFi adaptateur.

Figure III-34. Module USB WiFi adaptateur.

Une fois la clé WiFi inséré dans un des ports USBs du Raspberry Pi, il faut procéder à l'installation. Pour cela, il faut éditer le fichier de configuration réseau : **sudo nano /etc/network/interfaces**.

Une fois le fichier ouvert, la configuration par défaut doit être modifiée comme ci-dessous, en

remplaçant **NOM_RESEAU_WIFI** par le nom du réseau concerné et **CLE_RESEAU_WIFI** par la clé de sécurité réseau :

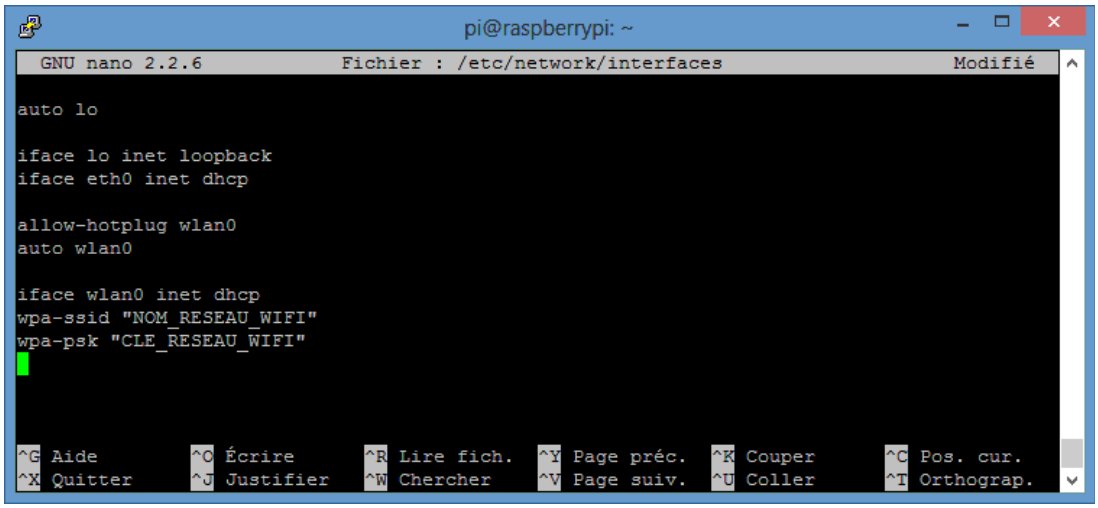


Figure III-35. Fichier de configuration des interfaces réseau.

Une fois cela fait, il faudra faire Ctrl + x pour quitter, on vous demandera si vous voulez sauvegarder, appuyer sur la touche O puis sur la touche entrée. Après avoir redémarré (**sudo reboot**), la clé WiFi s'allume, et le réseau fonctionne [22].

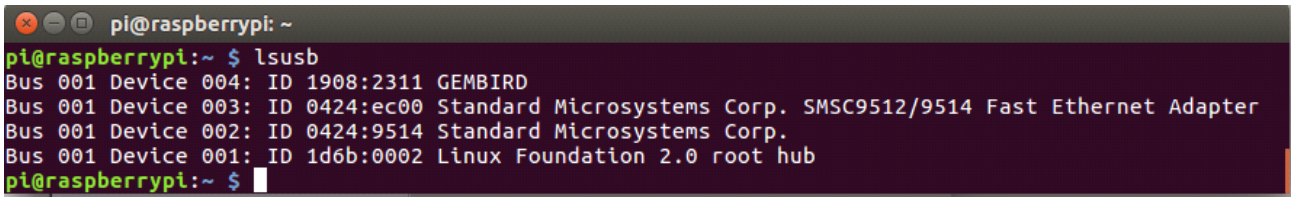
On a utilisé le système d'exploitation Windows pour faire les configurations de base sur notre Raspberry Pi, dans ce qui suit nous allons basé sur le système d'exploitation **Ubuntu 16.04** pour exécuter le reste des tâches et les programmes à cause de sa flexibilité.

III.3.4 Visualisation du flux vidéo de la webcam embarquée dans le robot

Nous avons testé différentes solutions pour la diffusion d'une vidéo en streaming dont le but est de trouver la solution la moins consommatrice en ressources (mémoire, CPU…) en gardant une bonne qualité. Ainsi, nous avons retenu la solution **mjpg-streamer**, celui-ci permet de pouvoir afficher le flux vidéo dans une page HTML par le simple biais d'une balise image en spécifiant le port de l'application. Le logiciel a fonctionné sans difficulté sur le système embarqué. Cependant, afin de bénéficier d'une vidéo fluide sans délai apparent, il a été nécessaire d'imposer un certain nombre d'images par seconde (framerate).

Concernant les performances, la page officielle sur Sourceforge annonce une consommation de 10% seulement pour un CPU tournant à 200 MHz pour une vidéo de 960x720 pixels.

Via ssh, on établit une connexion avec Raspberry Pi, et on tape la commande **lsusb** :



C'est le **GEMBIRD** qui représente notre webcam USB, car il est imprimé sur l'écran, il indique que le système a reconnu la webcam.

Vérifions si le driver de la webcam fonctionne normalement en tapant : **ls /dev/vid***

```
pi@raspberrypi ~ $ ls /dev/vid*
/dev/video0
```

Donc, le driver est dans l'état normal.

Installation du logiciel mjpg-streamer

Tout d'abord, nous avons besoin de récupérer le code source mjpg-streamer et l'enregistrer dans notre dossier sous le nom RobotPi2 dans /home/pi du Raspberry Pi.
Dans le répertoire où l'archive a été sauvegardée, on décompresse par la commande :
tar xvzf mjpg-streamer-r63.tar.gz

Dans un premier temps, on installe les dépendances de **mjpg-streamer** ainsi que subversion :

sudo apt-get install subversion

Ensuite, nous allons installer quelques outils nécessaires au bon fonctionnement de **mjpeg-streamer.**

sudo apt-get install libv41-dev
sudo apt-get install libjpeg8
sudo apt-get install imagemagick

Pour compiler mjpg-streamer, nous avons besoin des bibliothèques libjpeg8-dev que nous allons installer : **sudo apt-get install libjpeg8-dev**

Compilation pour l'installation de mjpg-streamer :

```
pi@raspberrypi: ~/RobotPi2/mjpg-streamer/mjpg-streamer
pi@raspberrypi:~ $ cd RobotPi2
pi@raspberrypi:~/RobotPi2 $ ls
django  djangoserver  i2cHelper.py  mjpg-streamer  server
pi@raspberrypi:~/RobotPi2 $ cd mjpg-streamer
pi@raspberrypi:~/RobotPi2/mjpg-streamer $ cd mjpg-streamer
pi@raspberrypi:~/RobotPi2/mjpg-streamer/mjpg-streamer $ sudo make USE_LIBV4L2=true clean all
```

```
pi@raspberrypi: ~/RobotPi2/mjpg-streamer/mjpg-streamer
pi@raspberrypi:~/RobotPi2/mjpg-streamer/mjpg-streamer $ sudo make DESTDIR=/usr install
install --mode=755 mjpg_streamer /usr/bin
install --mode=644 input_uvc.so output_file.so output_udp.so output_http.so input_testpicture.so
input_file.so /usr/lib/
install --mode=755 -d /usr/www
install --mode=644 -D www/* /usr/www
pi@raspberrypi:~/RobotPi2/mjpg-streamer/mjpg-streamer $
```

Pour tester, on exécuter la commande suivante : **sudo sh start.sh**

```
pi@raspberrypi:~ $ cd RobotPi2
pi@raspberrypi:~/RobotPi2 $ cd mjpg-streamer
pi@raspberrypi:~/RobotPi2/mjpg-streamer $ cd mjpg-streamer
pi@raspberrypi:~/RobotPi2/mjpg-streamer/mjpg-streamer $ sudo sh start.sh
MJPG Streamer Version: svn rev: 3:172M
 i: Using V4L2 device.: /dev/video0
 i: Desired Resolution: 640 x 480
 i: Frames Per Second.: 5
 i: Format............: YUV
 i: JPEG Quality......: 80
 o: www-folder-path...: ./www/
 o: HTTP TCP port.....: 8080
 o: username:password.: disabled
 o: commands.........: enabled
```

Par défaut, mjpg-streamer lancera un serveur web sur le port 8080/TCP auquel vous pouvez accéder avec un navigateur en utilisant l'adresse : **http://adress IP_du_Raspberry Pi:8080**.

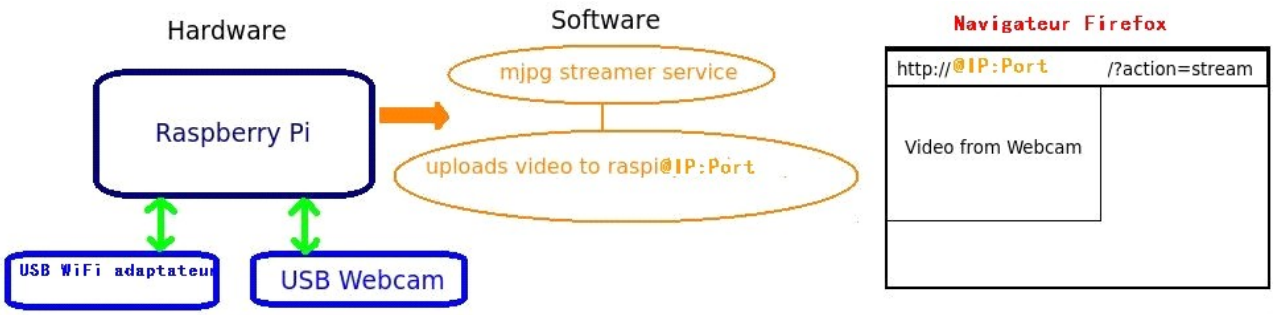


Figure III-36. Le principe de fonctionnement du logiciel mjpg-streamer.

Nous allons voir l'interface de démonstration de mjpg-streamer comme suit :

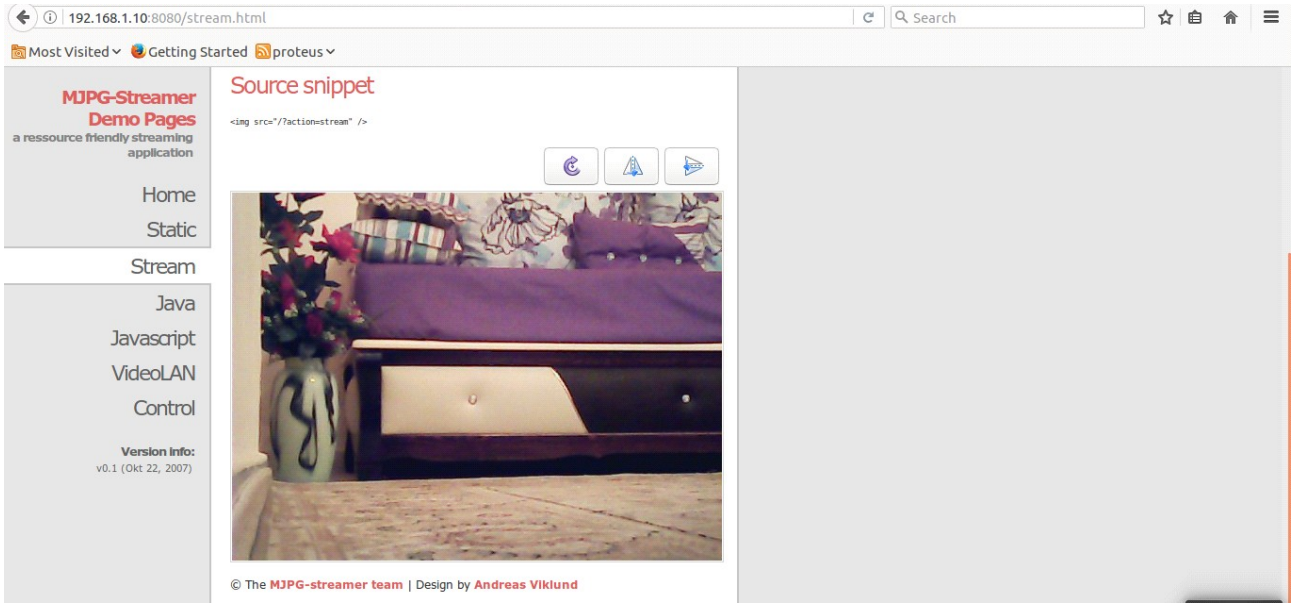


Figure III-37. Test du logiciel mjpg-streamer afin de récupérer le flux vidéo de la webcam.

Voilà, le robot est maintenant équipé pour faire un système de vidéo-streaming efficace.

Afin de réaliser le pilotage à distance du robot et pour des raisons de portabilité, nous choisissons de réaliser des applications web. Le gros avantage de ce type d'applications est qu'elle sera utilisable aussi bien depuis le navigateur d'un ordinateur que depuis celui d'un smartphone.

L'idée est de pouvoir accéder à notre Raspberry à distance via WiFi pour contrôler les GPIOs. Donc, on a besoin de rendre notre carte comme un serveur web capable d'héberger un ou plusieurs sites Internet pour assurer la communication avec un client et répondre à ses demandes grâce au protocole réseau http.

On s'intéresse aussi de commander notre robot via une application desktop en utilisant les sockets.

Voilà une aperçu général pour ce qui nous avons souhaitons faire dans ce chapitre selon l'architecture client-serveur suivante :

Figure III-38. Présentation des différentes applications qui sert à commander le robot.

III.3.5 Commander le robot via une application desktop sous Python

Des machines distantes peuvent communiquer entre elles grâce à leur adresse IP. Une machine cliente (c'est à dire qui demande un service) contacte une machine serveur qui répondra à sa demande, ceci est désigné par le terme client/serveur qui décrit une relation entre deux applications logicielles.

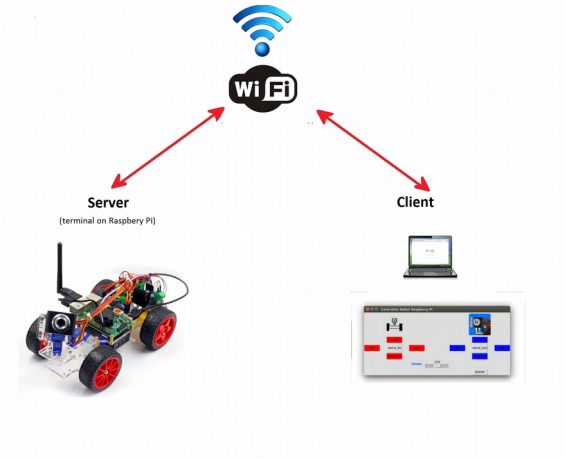


Figure III-39. Commander le robot via une application desktop.

Dans cette partie, nous allons donc apprendre à faire communiquer deux applications client et serveur. Pour réaliser ce besoin, on utilise des **sockets.** L'idée c'est que le client fasse une demande de type (adresse IP, port) puis de créer un lien entre ce port et notre programme.

Pour créer une connexion côté serveur ou client, on utilise le module **socket** et la classe **socket** de ce module.

III.3.5.1 Côté serveur (Raspberry Pi)

N'importe quel ordinateur peut devenir un serveur à condition qu'un programme écoutant les requêtes des clients et fonctionne en permanence pour répondre à chaque fois. Pour cela, on va rendre notre Raspberry Pi comme serveur.
Il faut tout d'abord installer les outils nécessaires en tapant les commandes :

sudo apt-get install python-dev
sudo apt-get install python-smbus

Socket serveur :

Le fabricant de la société **Sunfounder** a déjà codé les librairies pour contrôler les servomoteurs et les moteurs en Python (import video_dir, import car_dir et import motor).
Le code source concernant la partie de programmation socket est :

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
import video_dir
import car_dir
import motor
from socket import *
from time import ctime
busnum = 1
HOST = ''
PORT = 21567
BUFSIZ = 1024
ADDR = (HOST, PORT)
tcpSerSock = socket(AF_INET, SOCK_STREAM)
tcpSerSock.bind(ADDR)
tcpSerSock.listen(5)
```

Explication du code source :

- Commençons donc, par importer notre module socket.

- **Construire notre socket**

Pour établir une connexion TCP, on appelle le constructeur socket qui prend les deux paramètres suivants dans l'ordre :

socket.AF_INET : la famille d'adresses, ici ce sont des adresses Internet.

socket.SOCK_STREAM : le type du socket, SOCK_STREAM pour le protocole TCP.

- **Connecter le socket**

Ensuite, nous connectons notre socket. Pour une connexion serveur qui va attendre des connexions de clients, on utilise la méthode **bind**. Elle prend un paramètre : le tuple (HOST, PORT), pour que notre serveur écoute sur un port, il faut le configurer en conséquence. Donc, dans notre cas, le nom de l'hôte sera vide et le port sera celui que vous voulez, entre 1024 et 65535.

- **Faire écouter notre socket**

Notre socket est prêt à écouter sur le port 21567 mais il n'écoute pas encore. On va avant tout lui préciser le nombre maximum de connexions qu'il peut recevoir sur ce port sans les accepter. On utilise pour cela la méthode **listen**. On lui passe généralement 5 en paramètre.

- **Accepter une connexion venant du client**

Maintenant, on va accepter une connexion parce que jusqu'à la aucune connexion ne s'est encore présentée mais la méthode **accept** que nous allons utiliser va bloquer le programme tant qu'aucun client ne s'est connecté. Voilà la ligne de programme pour cette méthode :

tcpCliSock, addr = tcpSerSock.accept()

Il est important de noter que la méthode accept renvoie deux informations : le socket connecté qui vient de se créer, celui qui va nous permettre de dialoguer avec notre client tout juste connecté; un tuple représentant l'adresse IP et le port de connexion du client.

III.3.5.2 Côté client (PC Ubuntu 16.04)

Socket client :

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import *
from socket import *   # Importer les modules necessaires
window = Tk()   # Creation d'un Fenêtre
window.title('Controller Robot Raspberry Pi') # Titre de L'application

HOST = '192.168.1.5'    # serveur (Raspberry Pi) @ip
PORT = 21567
SIZBUF = 1024            # Taille du buffer
ADDR = (HOST, PORT)
tcpCliSock = socket(AF_INET, SOCK_STREAM)   # Creation d'un socket
tcpCliSock.connect(ADDR)                    # Se connecter avec le serveur (Raspberry Pi)
```

- On commence par construire notre socket de la même façon :

from socket import *

tcpClisock = socket(AF_INET, SOCK_STREAM)

- **Connecter le client**

Pour se connecter à notre serveur, le socket client utilise la méthode **connect**. Elle prend en paramètre un tuple, comme bind, contenant l'adresse IP (HOST) et le numéro du port identifiant le serveur auquel on veut se connecter.

Le numéro du port sur lequel on veut se connecter c'est 21567. Vu que nos deux applications Python ne sont pas sur la même machine, donc HOST va être l'adresse IP du Raspberry Pi.
Et voilà, notre serveur et notre client sont connectés.

- **Faire communiquer nos sockets**

On utilise les méthodes **send** pour envoyer et **recv** pour recevoir. La méthode **send** vous renvoie le nombre de caractères envoyés. La méthode **recv** prend en paramètre le nombre de caractères à lire. Généralement, on lui passe la valeur 1024.

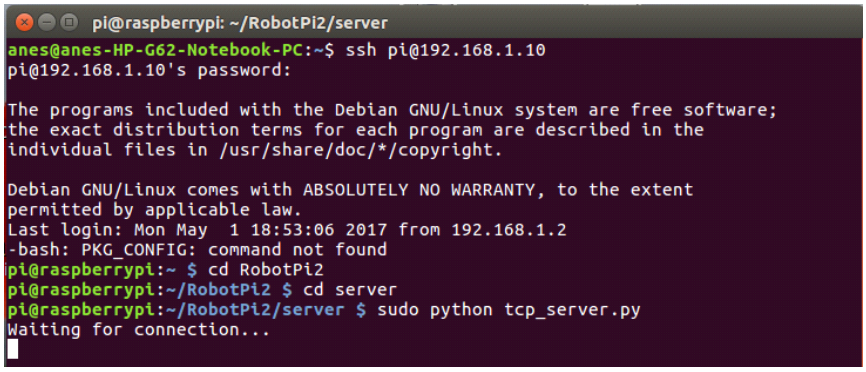
- **Fermer la connexion**

Pour fermer une connexion, le socket serveur ou client utilise la méthode **close**.

Côté serveur : **tcpSerSock.close()**.

Côté client : **tcpCliSock.close()**.

Exécution du serveur :



Le programme du serveur sur le Raspberry Pi fonctionnera et attendra le client pour se connecter.

Exécution du programme client de l'application desktop :

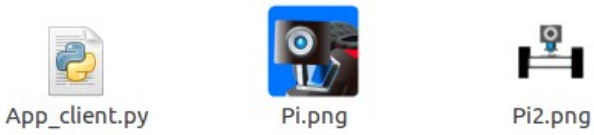


Figure III-40. Répertoire client de l'application desktop.

```
anes@anes-HP-G62-Notebook-PC: ~/Desktop/RobotPi/client
anes@anes-HP-G62-Notebook-PC:~$ cd /home/anes/Desktop
anes@anes-HP-G62-Notebook-PC:~/Desktop$ cd RobotPi
anes@anes-HP-G62-Notebook-PC:~/Desktop/RobotPi$ cd client
anes@anes-HP-G62-Notebook-PC:~/Desktop/RobotPi/client$ sudo python App_client.py
[sudo] password for anes:
sendData = speed200
```

L'interface de notre application apparaît comme suit :

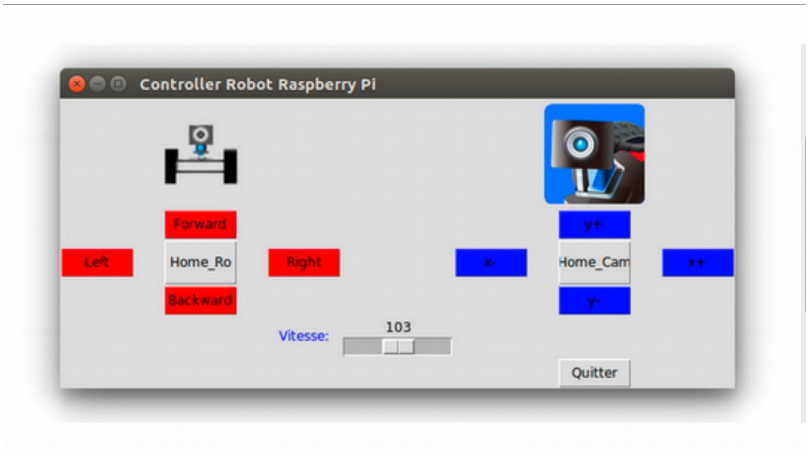


Figure III-41.L'interface de l'application desktop.

III.3.6 Commander le robot via une application Android

En 2014, le nombre d'accès au web avec un appareil mobile a dépassé ceux effectués avec un ordinateur de bureau, c'est pour ça, on a besoin de plus en plus de rendre les applications des appareil mobiles de transmission sans fil accessibles au web ou d'accéder à Internet de façon générale.

Dans ce cadre, nous allons voir comment commander notre robot via un appareil Android à l'intermidiare d'un site web Django, ce site s'exécute au niveau du Raspberry Pi (côté serveur).

Pour pouvoir créer l'application, **App Inventor2** a été utilisé et permet à la fois aux débutants de commencer à créer leur propre application pour smartphone, mais permet aussi aux plus confirmés de manipuler des fonctionnalités plus complexes.

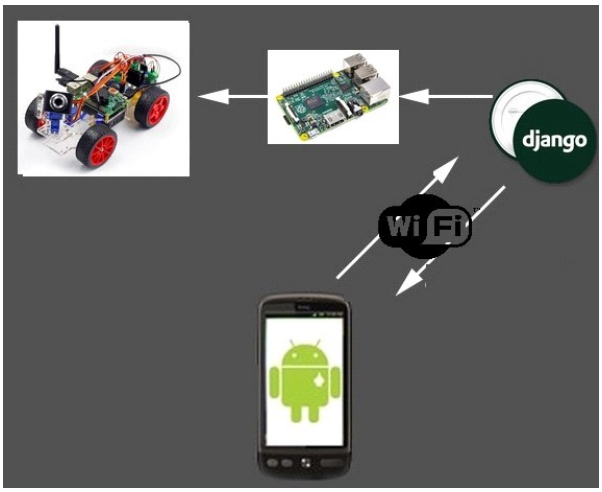


Figure III-42. Commander le robot via application Android.

III.3.6.1 Partie serveur (site Web Django sous Raspberry Pi)

Raspberry Pi comme serveur web

Qu'est-ce qu'un serveur ?

« Un **serveur** est un dispositif informatique matériel ou logiciel qui offre des services à différents clients». En effet, le serveur est un programme qui rend service à ses clients qui lui envoient des requêtes et ses clients ce sont par exemple nos navigateurs web. Quand vous voulez aller sur **www.google.com,** votre navigateur va d'abord envoyer une requête à un serveur DNS qui, lui, offre comme service de traduire www.google.com en adresse IP à laquelle votre navigateur peut alors envoyer une autre requête à un serveur web qui aura pour service de lui envoyer la page qu'il demande,c'est-à-dire la page d'accueil de www.google.com.

Dans ce contexe, on a besoin d'un serveur web visant à héberger notre site, qui devront être accessibles depuis l'extérieur afin de commander notre robot via un navigateur ou application Android. Pour cela, on a utilisé la programmation Python web sous Django, mais c'est quoi Django ?

Django est un **framework web** écrit en Python, qui se veut complet tout en facilitant la création d'applications web riches.

Figure III-43. Le logo de Django.

Aujourd'hui, Django est devenu très populaire et est utilisé par des sociétés du monde entier, telles qu'Instagram, Pinterest, et même la NASA.

Figure III-44. Logos d'Instagram, de la NASA et de Pinterest.

Avant de commencer la conception de notre site, nous allons tout d'abord voir et comprendre le fonctionnement de ce framework, ce qui vous sera grandement utile lorsque vous commencerez à créer n'importe quel site sous Django.

a) Le fonctionnement de Django

Lorsque nous parlons de frameworks qui fournissent une interface graphique à l'utilisateur (soit une page web, comme ici avec Django, soit l'interface d'une application graphique classique, comme celle de votre traitement de texte par exemple), nous parlons souvent de l'architecture **MVC**. Comme son nom l'indique, l'architecture **M**odèle-**V**ue-**C**ontrôleur est composée de trois entités distinctes, chacune ayant son propre rôle à remplir. Tout d'abord, le **M**odèle représente une information enregistrée quelque part, le plus souvent dans une base de données. Ensuite la **V**ue qui est, comme son nom l'indique, la visualisation de l'information. C'est la seule chose que l'utilisateur peut voir. Finalement, le **C**ontrôleur prend en charge tous les événements de l'utilisateur (accès à une page, soumission d'un formulaire, etc.). Lors de l'appel d'une page, c'est le **C**ontrôleur qui est chargé en premier, afin de savoir ce qu'il est nécessaire d'afficher.

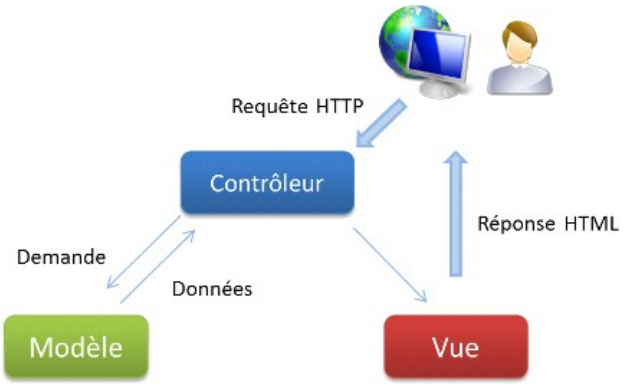


Figure III-45. Schéma de l'architecture MVC.

L'architecture utilisée par Django diffère légèrement de l'architecture **MVC** classique. En effet, la « magie » de Django réside dans le fait qu'il **gère lui-même la partie contrôleur** (gestion des requêtes du client, des droits sur les actions…). Ainsi, nous parlons plutôt de framework utilisant l'architecture **MVT** : **M**odèle-**V**ue-**T**emplate. Cette architecture reprend les définitions de modèle et de vue que nous avons vu, et on introduit une nouvelle : le **Template**. Un template est un fichier HTML. Il sera récupéré par la vue et envoyé au visiteur. Cependant, avant d'être envoyé, il sera analysé et exécuté par le framework, comme s'il s'agissait d'un fichier avec du code.

Maintenant que nous avons vu l'architecture de fonctionnement de Django, il est temps de passer à son installation. Via pip, il suffit de demander l'installation de Django avec la commande suivante : **sudo pip install Django**

L'outil de gestion fourni avec Django se nomme **django-admin.py** et il n'est accessible qu'en

ligne de commandes. En plus de l'architecture MVT, Django introduit le développement d'un site web sous forme de projet.

Nous appelons notre projet **djangoserver** et créons-le grâce à la commande suivante :
django-admin.py startproject djangoserver

À la création du projet, Django déploie un ensemble de fichiers, facilitant à la fois la structuration du projet et sa configuration.

Dans le dossier principal djangoserver, nous retrouvons deux éléments : un fichier **manage.py** et un autre sous-dossier nommé également djangoserver.
Le sous-dossier contient quatre fichiers Python, à savoir **settings.py, urls.py, wsgi.py** et **__init__.py**.

settings.py contiendra la configuration de notre projet, Il est nécessaire de modifier le settings.py afin de configurer le projet selon nos besoins.
Tandis que **urls.py** rassemblera toutes les URL de notre site web et la liste des fonctions à appeler pour chaque URL. Voilà notre code source concernant le script urls.py :

```
1 from django.conf.urls import url
2 from django.contrib import admin
3 import views
4
5 urlpatterns = [
6     url(r'^admin/', admin.site.urls),
7     url(r'^robot/avancer', views.robot_avancer),
8     url(r'^robot/arriere', views.robot_arriere),
9     url(r'^robot/areter', views.robot_areter),
10    url(r'^webcam/haut', views.webcam_haut),
11    url(r'^webcam/bas', views.webcam_bas),
12    url(r'^webcam/droite', views.webcam_droite),
13    url(r'^webcam/gauche', views.webcam_gauche),
14    url(r'^webcam/postion_initial', views.webcam_postion_initial),
15    url(r'^moteur/change/vitesse/(\d{1,3})', views.moteur_change_vitesse),
16   url(r'^django_website_client/', views.django_website_client),
17 ]
```

Le minimum requis pour obtenir une page web avec Django est une vue, associée à une URL.

Une vue est une fonction placée dans le fichier views.py d'une application. Cette fonction doit toujours renvoyer un objet **HttpResponse**.

Pour être accessible, une vue doit être liée à une ou plusieurs URL dans les fichiers urls.py du projet.

Les URL sont désignées par des expressions régulières, permettant la gestion d'arguments qui peuvent être passés à la vue pour rendre l'affichage différent selon l'URL visitée.

Voilà une partie de code source (**views.py**) qui explique cette relation entre les URL définie au niveau de script **urls.py**.

```
def robot_avancer(request):
        motor.forward()
        return HttpResponse("robot avancer")

def robot_arriere(request):
        motor.backward()
        return HttpResponse("robot arriere")

def robot_areter(request):
        motor.ctrl(0)
        return HttpResponse("robot areter")

def webcam_haut(request):
        video_dir.move_increase_y()
        return HttpResponse("webcam haut")

def webcam_bas(request):
        video_dir.move_decrease_y()
        return HttpResponse("webcam_bas")

def webcam_gauche(request):
        video_dir.move_increase_x()
        return HttpResponse("webcam gauche")

def webcam_droite(request):
        video_dir.move_decrease_x()
        return HttpResponse("webcam droite")

def webcam_postion_initial(request):
        video_dir.home_x_y()
        return HttpResponse("webcam_postion_initial")

def django_website_client(request):
        return render(request, 'djano_website_client.html')
```

L'administration de projet s'effectue via le script **manage.py** soit dans le cas des mises à jour du bases de données après chaque modification effectué de notre site en utilisant la commande **python manage.py migrate**, ou bien si nous voulons partager notre projet avec d'autres utilisateurs, nous pouvons choisir l'adresse IP 0.0.0.0 c'est à dire rendre disponible notre serveur sur tout le réseau local en tapant la commande : **python manage.py runserver 0.0.0.0:8000**.

b) Conception du site web Django

Notre site est composé de trois dossiers et 9 scripts Python web Django selon la figure ci-dessous tel que le dossier djangoserver qui comprend les scripts Python **settings.py, urls.py, wsgi.py** et **__init__.py** et les bibliothèques fournies par le fabricant de la société Sunfounder, ainsi le dossier js_Bootstrap qui assure le design de notre site et le dernier dossier Template contient la page HTML qui fournit à l'utilisateur la page web pour contrôler le robot.

Le code source se trouve dans le répertoire /home/pi/djangoserver de la carte Raspberry Pi.

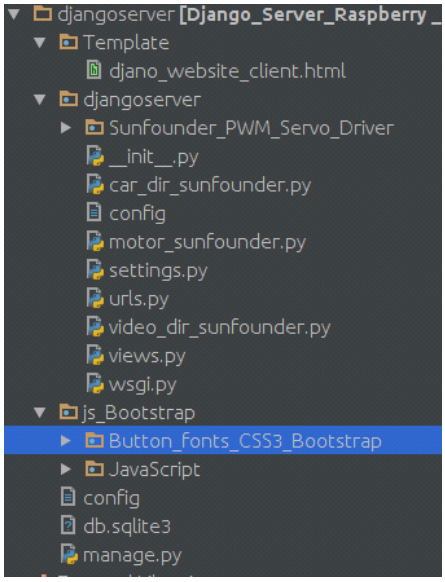


Figure III-46. Arborescence du projet.

Notre serveur web est maintenant totalement configuré et opérationnel.

Exécution du serveur web Django :

```
pi@raspberrypi: ~/RobotPi2/djangoserver
pi@raspberrypi:~/RobotPi2 $ cd djangoserver
pi@raspberrypi:~/RobotPi2/djangoserver $ python manage.py runserver 0.0.0.0:8000
Performing system checks...
 o: www-folder-path...: ./www/
 o: HTTP TCP port.....: 8080
 o: username:password.: disabled
 o: commands..........: enabled
System check identified no issues (0 silenced).
May 14, 2017 - 20:35:29
Django version 1.10.6, using settings 'djangoserver.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
```

Cette console donne des informations, des logs (une historique indiquant quelle page a été accédée, quand et par qui) et les exceptions de Python lancées en cas d'erreur lors du développement. Par défaut, l'accès au site de développement se fait via l'adresse http://localhost:8000.

De même, il est possible de spécifier une adresse IP différente de 127.0.0.1 (pour l'accessibilité depuis le réseau local par exemple dans notre cas).

III.3.6.2 Partie Client (depuis un navigateur)

Pour que notre site devient dynamique, on a besoin de connaître quatre langages : le HTML (pour le squelette de la page), le CSS (le style de la page), et le JavaScript (pour gérer les interactions avec l'utilisateur).

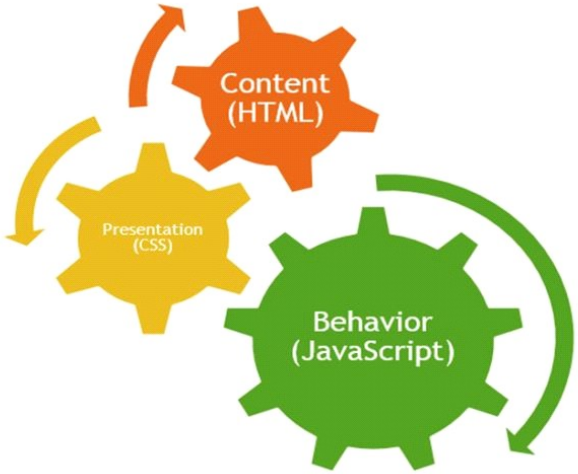


Figure III-47. Langages de développement web côté client.

La partie cliente a été réalisée principalement en HTML5, CSS et Javascript. L'appel aux différents fichiers a été géré directement dans le HTML ainsi que les fichiers CSS permettent d'écrire l'affichage et les feuilles de style utilisés pour la page principale. Le Javascript quant à lui permet d'ajouter de l'interaction coté client et notamment la connectivité. Le Framework Javascript jQuery a été utilisé afin de gagner du temps dans la réalisation de certaines interactions.

Voici une représentation de l'application depuis un navigateur :

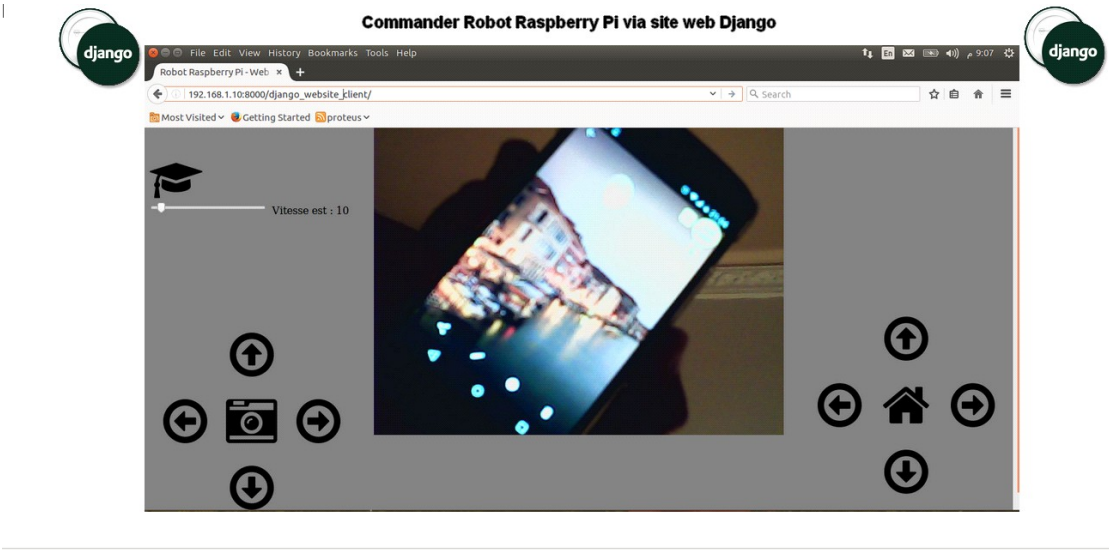


Figure III-48. Interface web Django.

L'interface propose sur la partie droite les bouttons qui permettre de piloter le robot selon les différentes directions, ensuite sur la partie gauche se trouve les bouttons pour contrôler notre webcam et au centre de la page vidéo streaming en temps réel .

Ce site est accessible que sur un réseau local. Autrement dit, seule une personne connectée sur notre réseau Wi-Fi ou via un câble Ethernet peuvent accèder à notre site.

Lorsque un utilisateur appelle cette page web de notre site, le framework se charge, via les règles de routage URL définies, d'exécuter la vue correspondante. Cette dernière récupère les données des modèles et génère un rendu HTML à partir du template et de ces données.

Une fois la page générée, l'appel fait chemin arrière, et le serveur renvoie le résultat au navigateur de l'utilisateur [23].

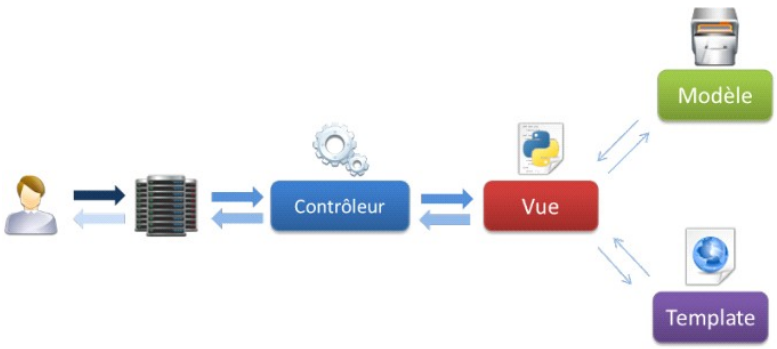


Figure III-49. Schéma d'exécution d'une requête.

III.3.6.3 Partie Client (Développement de l'application Android sous App Inventor2)

Nous allons présenter dans cette partie étape par étape la conception de notre application Android que l'on a nommée **RobotPi** servant à contrôler notre robot par l'intermidiare d'un site web sous Django.

Maintenant, nous allons réaliser l'application Android grâce à App Inventor développé par Google et basé sur une interface graphique similaire à scratch. C'est un logiciel en ligne qui permet de créer des applications pour appareils Android (smartphone ou tablette). L'outil est gratuit et permet de développer sa créativité et ses compétences en programmation. Ainsi, vous pourrez réaliser vos applications, mais aussi vos objets connectés via une carte Raspberry Pi ou Arduino par exemple. Pour mettre l'application dans l'App Inventor, tu dois travailler avec deux environnements, le concepteur et l'éditeur de blocs.

a) App Inventor Designer

La fenêtre de création, ou simplement « Designer » est où vous étendez le regard et ressent de votre application et spécifiez quelles fonctionnalités il devrait avoir. Vous choisissez des choses pour l'utilisateur interface choses comme les boîtes de boutons, images et texte et des fonctionnalités comme la synthèse vocale, capteurs et GPS.

Figure III-50. Mode designer de l'App Inventor.

Conception de l'application Android

En mode «designer» :

Dans cette partie, nous allons voir les différentes interfaces de notre application Android pour contrôler le robot en ajoutant des boutons, des composants...etc.

Interface d'authentification :

Dans cette interface, nous avons deux champs (username et password) alors lorsqu'on clique sur le bouton " Login " on appelle une méthode **Log_in** pour faire une vérification avec les informations introduites et celles enregistrées dans la base de données de notre application.

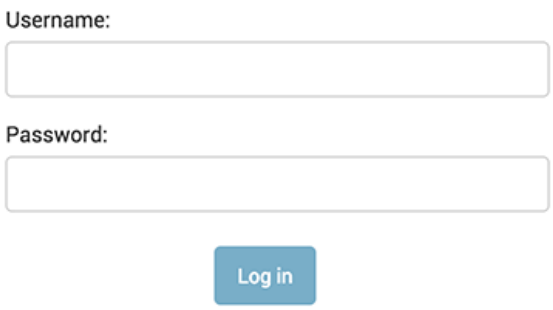


Figure III-51. Interface d'authentification de l'application Android.

Interface d'accueil :

Cette interface est apparue après avoir eu une authentification réussie. Elle contient un simple champ qui demande à l'utilisateur d'introduire l'adresse IP valable du serveur, puis un bouton rouge pour accéder à l'interface de base qui sert à contrôler notre robot.

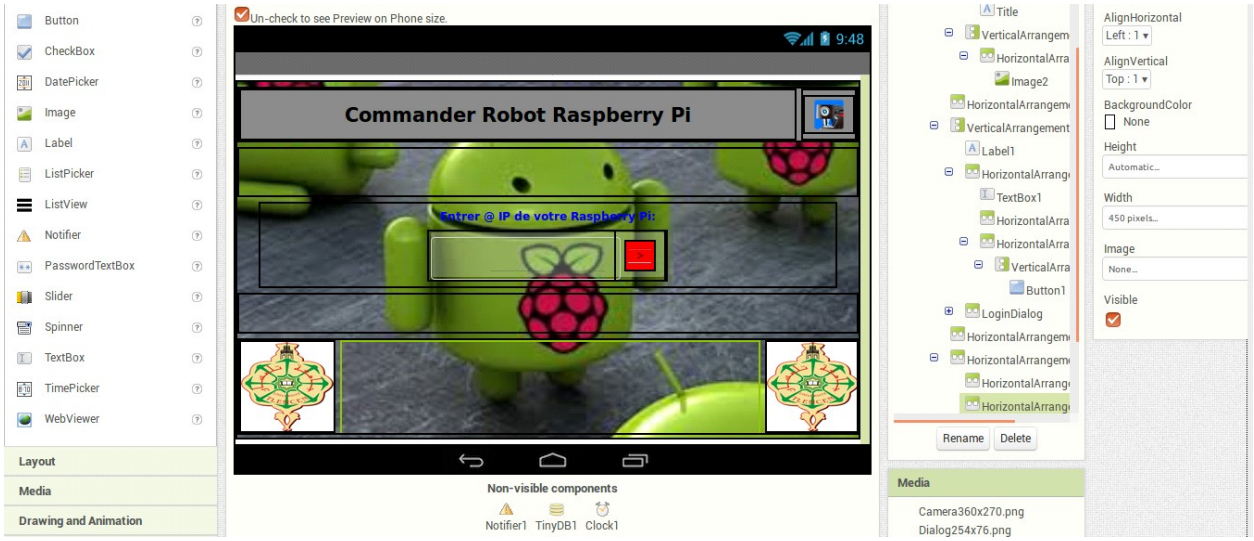


Figure III-52. Vue de l'interface d'accueil.

Interface de contrôle du robot :

Enfin, sur cette interface se trouve les bouttons directionnels, "en rouge", pour faire avancer et reculer le robot, aussi on utilise les capteurs de position d'un smartphone pour contrôler la direction du robot (permettre d'orienter les deux roues d'avant du robot suivant l'inclinaison du téléphone). On ajoute cinq bouttons "en blue" qui permettent de tourner la webcam selon les axes X et Y.
Un simple scale pour changer la vitesse du robot, et on s'interesse aussi pour le streaming vidéo obtenu par un serveur mjpeg-streamer.

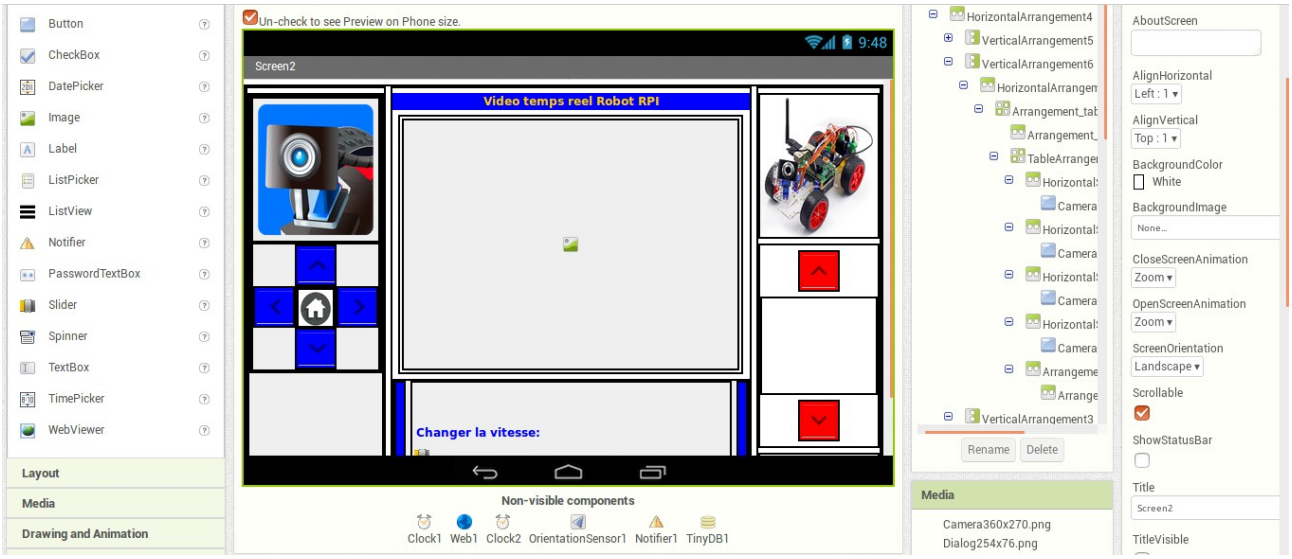


Figure III-53. Interface de contrôle du robot.

b) Éditeur de blocs App Inventor

En mode «blocks» :

Après les étapes précédentes, il faut connecter tous les éléments afin d'obtenir un programme fiable et opérationnel. Grâce à AppInventor, nous n'avons pas besoin d'avoir de solides connaissances en Java, les fonctions sont déjà préparées et l'utilisateur n'a besoin que de les rassembler pour créer son application. Voici toutes les étapes de cette conception qui permet à concevoir un programme pour contrôler notre robot.

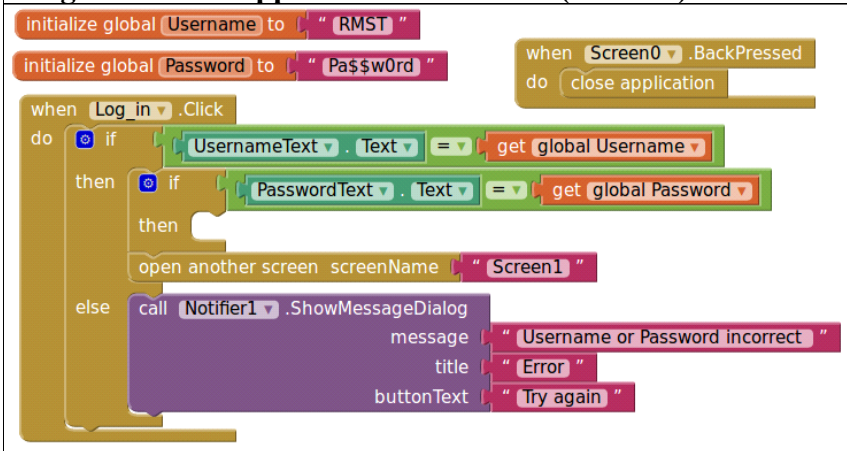
Programme de l'application ANDROID (Screen0)	Commentaires
	Initialiser le nom d'utilisateur avec "RMST"et le mot de passe avec "Pa$$w0rd". Quand on clique sur le bouton "Login", si le nom d'utilisateur et le mot de passe saisis dans les zones de texte correspondent avec ceux initialisés, on ouvre une nouvelle interface "Screen1". Si ce n'est pas le cas, une fenêtre de notification s'ouvre indiquant un message d'erreur. Si on appuie sur "BackPressed" du smartphone, on quitte l'application.

Tableau III-6. Programmes de l'interface d'authentification.

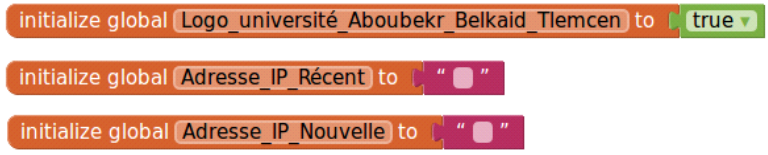
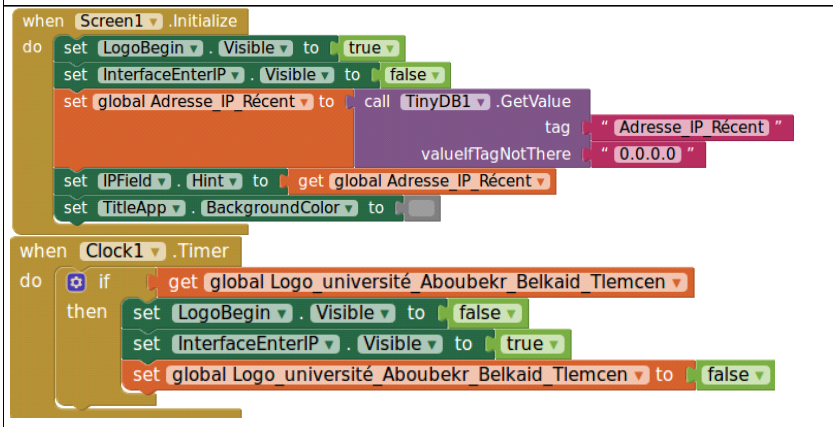
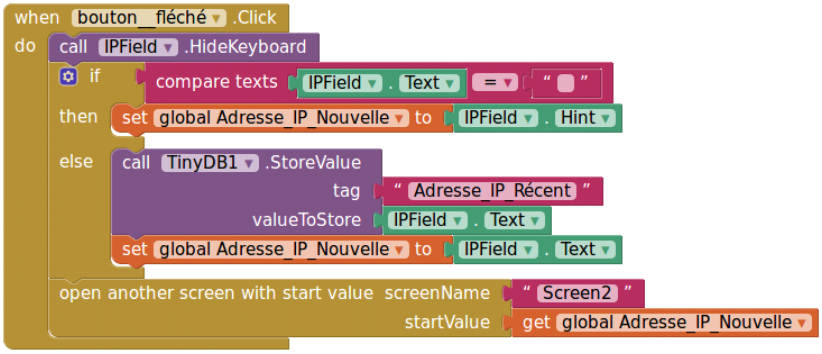
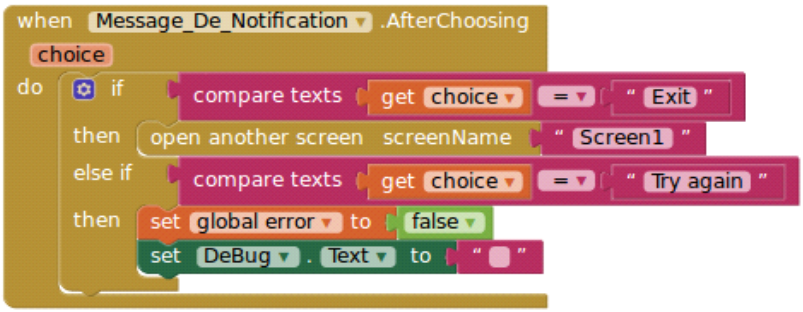
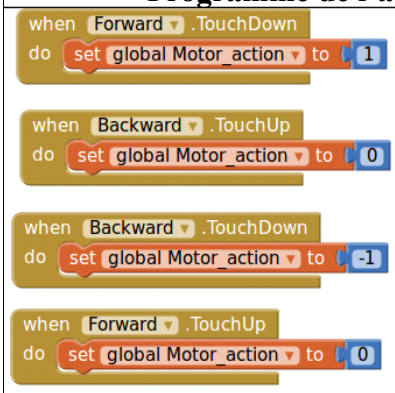
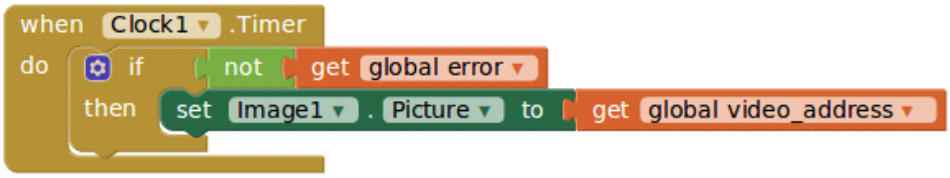
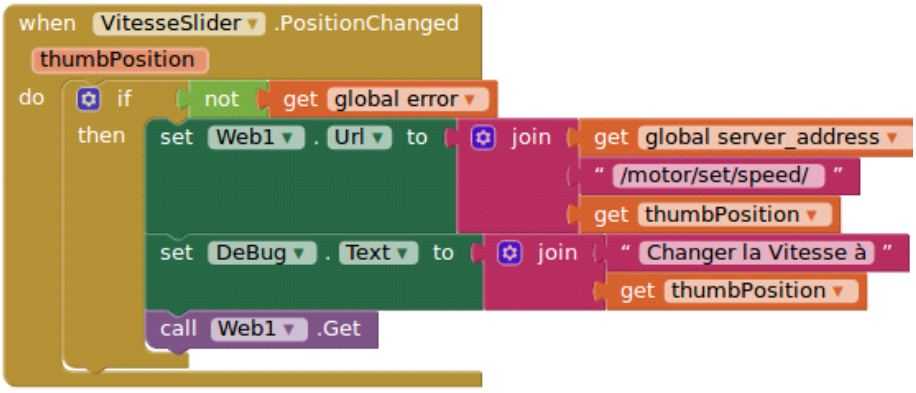
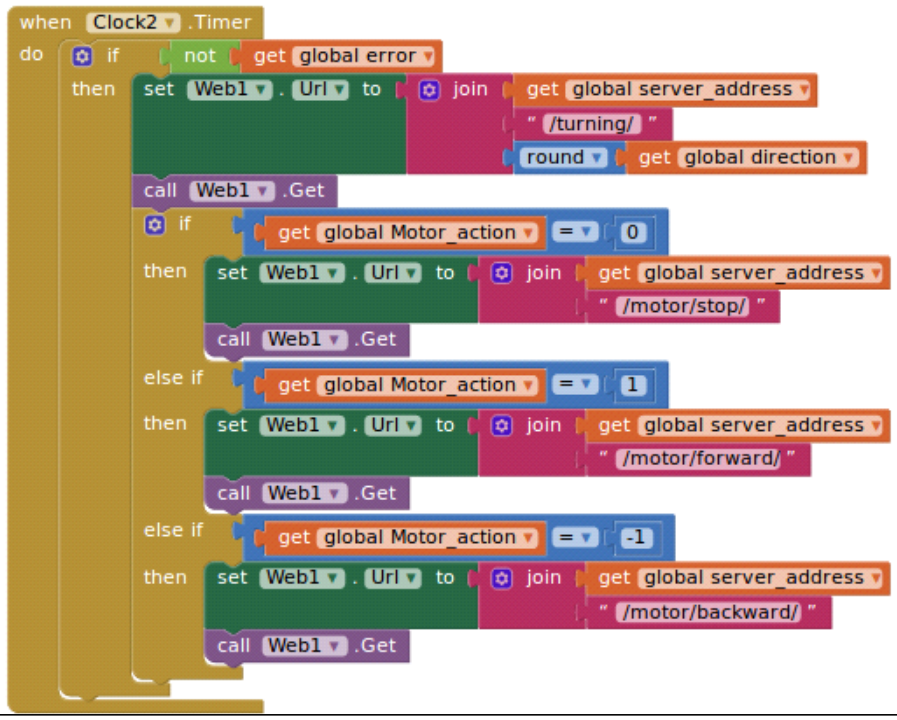
Programme de l'application ANDROID (Screen1)	Commentaires
	Initialisation de la variable globale logo avec une variable booléenne true et les adresses IP précédente et nouvelle avec des chaînes de caractères vides.
	Si on l'authentification est réussie, l'interface d'accueil s'initialise avec le logo de notre université Aboubekr Belkaid dans un laps de temps très court (1000 ms) contrôlée par le Clock1, après cela l'interface de la figure III-52 apparaît en demandant à l'utilisateur d'introduire l'adresse IP du serveur.
	Ensuite, on clique sur le bouton rouge selon que l'adresse IP soit valide ou non on peut permuter à l'interface de contrôle du robot.
	Mais si l'adresse IP est faux, un message de notification apparaît n'autorisant pas de commander le robot, ce message contient deux boutons **Exit** et **Try again.** Selon le choix de l'utilisateur, si le choix est **Exit,** on retourne à l'interface d'accueil, d'autre part si le choix est **Try again,** l'utilisateur réessaie une autre fois

Tableau III-7. Programmes de l'interface d'accueil.

Programme de l'application ANDROID (Screen2)	Commentaires
	Envoi au moteur du robot des données spécifiques, ceci est associé lorsqu'on clique sur les deux boutons rouges qui sont situées dans gauche de cette interface (figure III-53).
	S'il n' y a pas d'erreur concernant l'adresse du serveur webcam mjpg-streamer, le flux vidéo temps réel sera affiché au niveau de Image1.
	Lorsque on pose notre pouce sur le slider pour changer la vitesse de notre robot, la requête est envoyée au serveur.
	Quand on clique sur les boutons pour contrôler notre webcam selon l'axe X (gauche ou droite) ou bien selon l'axe Y (haut ou bas) des commandes sera exécuté pour cela.

	Ce Clock2 a pour fonction de déclencher périodiquement ce traitement qui permet de piloter notre robot.
	Ce sont des variables globales qui permet d'appeler dans n'importe quel bloc du Screen 2.

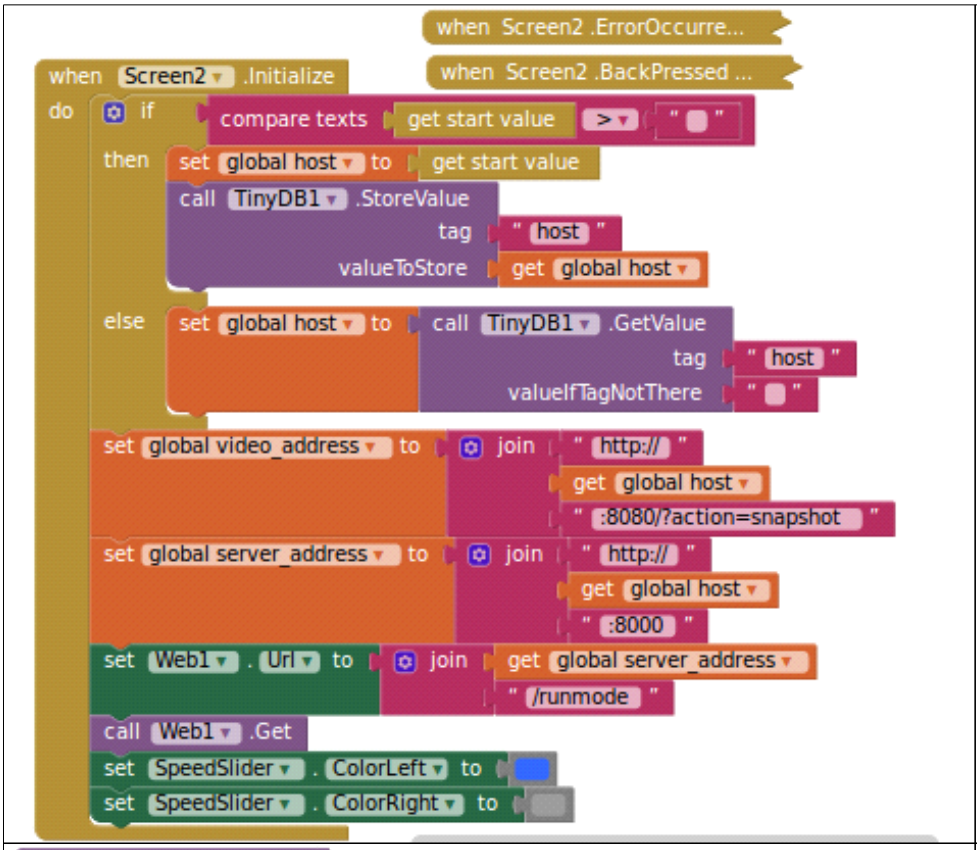
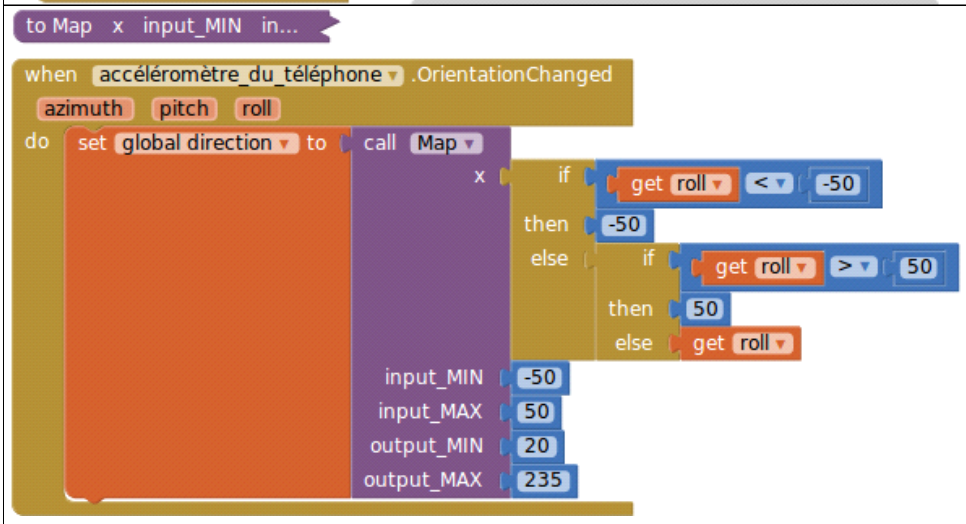
<table>
<tr>
<td> 
</td>
<td>On utilise le protocole http pour communiquer avec le serveur web Django (via le port 8000) et le serveur mjpg-streamer pour la vidéo (via le port 8080) si l'adresse IP de ce serveur (host) est valide.</td>
</tr>
<tr>
<td> 
</td>
<td>Fonction qui détecte la moindre variation du mouvement du téléphone, et selon l'inclinaison de notre téléphone le robot va tourner à la gauche ou la droite.</td>
</tr>
</table>

Tableau III-8. Programmes de l'interface de contrôle du robot.

Enfin, nous allons illustrer ici les différentes interfaces graphiques de notre application Android.

Figure III-54. Les différentes interfaces de notre application Android RobotPi.

III.3.7 Contrôler le robot via une page web PHP

III.3.7.1 Le Raspberry Pi comme un serveur web Apache

La mise en place d'un serveur web sur le Raspberry Pi peut s'avérer très intéressante. Elle permet notamment de se familiariser avec l'installation et la configuration d'un serveur Linux.

Le but de ce projet est de rendre notre Raspberry Pi un serveur web capable d'héberger plusieurs sites Internet et assurer donc la communication avec un client et répondre à ses demandes grâce au protocole réseau http.

Dans notre cas, Apache sert de serveur web, la base de données est en MySQL et le langage de script en PHP.

Le schéma suivant illustre comment s'effectuent les échanges entre le client et notre serveur web :

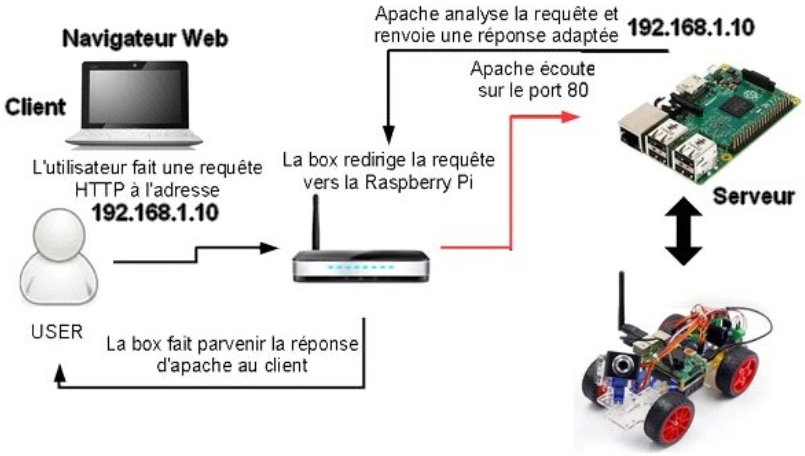


Figure III-55. Commander le robot via page web PHP.

Pour faire fonctionner un serveur web sur le Raspberry Pi, il faut installer plusieurs choses :
Le serveur HTTP Apache 2 : c'est lui qui permettra de gérer les requêtes http sur le système.
PHP 5 : L'interpréteur du très répandu langage php utilisé pour concevoir les sites web dynamiques.
Libapache2-mod-php5 : système permettant d'utiliser php avec Apache.

Vient ensuite le moment de télécharger les applications. Tout d'abord, nous allons installer Apache.

III.3.7.2 Installation du serveur Apache

Apache est le serveur web le plus utilisé dans le monde, très documenté visant son installation, son utilisation et sa sécurisation. Apache va nous permettre d'héberger des fichiers au format HTML afin qu'un navigateur web puisse les lire. Avant de débuter l'installation, il faut commencer par mettre à jour les paquets et faire la mise à jour du système avec les commandes suivantes :

sudo apt-get update
sudo apt-get upgrade

Une fois le Raspberry Pi est mis à jour, nous allons installer le serveur Apache par la commande suivante :

```
pi@raspberrypi:~ $ sudo apt-get install apache2
```

Après son exécution, nous donnons les droits d'accès adéquats au dossier racine de notre serveur à Apache qui vous permettra de facilement administrer les sites. Pour cela, lancez les commandes suivantes :

sudo chown -R pi:www-data /var/www/html/
sudo chmod -R 770 /var/www/html/

Une fois l'installation terminée, nous pouvons tester qu'Apache fonctionne correctement en nous rendant sur l'adresse du Raspberry Pi.
Pour cela, il faut accéder au Raspberry depuis ssh. Il suffit d'ouvrir le navigateur web de notre PC (par exemple Mozilla), et d'aller à l'adresse «http://192.168.1.10». Alors, on obtient une page avec un message du genre «**It works!**» et plein d'autre texte, ce qui indique qu'Apache fonctionne.
Apache utilise le répertoire «/var/www/html» comme racine pour notre site, donc, Apache cherche le fichier dans «/var/www/html».

Pour communiquer avec le serveur web Apache, on a besoin d'un langage de programmation serveur comme PHP.

III.3.7.3 Installation de PHP sur le Raspberry Pi

PHP est principalement utilisé pour rendre un site dynamique via un serveur HTTP, c'est-à-dire que l'utilisateur envoie des informations au serveur qui lui renvoie les résultats modifiés en fonction de ces infos.

PHP est tout d'abord un langage script côté serveur. En des termes plus simples, il s'agit d'un langage qui nécessite d'être exécuté sur un serveur pour que ce qu'il est censé afficher le soit correctement (contrairement au HTML qui peut être lu sans serveur). Mais il s'agit également d'un interpréteur, car PHP est un langage interprété (en opposition au langage compilé). Nous allons donc installer cet interpréteur sur notre serveur afin qu'il puisse renvoyer à notre utilisateur le résultat de cette interprétation plutôt qu'un bout de code sans aucune signification.
Pour l'installer, il vous suffit de taper cette commande dans un terminal :

```
pi@raspberrypi:/etc/apache2 $ sudo apt-get install php5 libapache2-mod-php5 --fix-missing
```

Pour savoir si PHP fonctionne correctement, on a créé un fichier «index.php» dans le répertoire «/var/www/html» avec cette ligne de commande :

```
pi@raspberrypi: ~
pi@raspberrypi:~ $ echo "<?php phpinfo(); ?>" > /var/www/html/index.php
```

À partir de là, le fonctionnement est le même que pour la vérification d'Apache. Cela montre que PHP fonctionne correctement [24].

PHP Version 5.4.4-14+deb7u7

System	Linux ajaniserveur 3.2.0-4-amd64 #1 SMP Debian 3.2.54-2 x86_64
Build Date	Dec 12 2013 08:42:50
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-gd.ini, /etc/php5/apache2/conf.d/20-mysql.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5/apache2/conf.d/20-pdo_mysql.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend Extension Build	API220100525,NTS

Figure III-56. Vérification de fonctionnement de PHP sur le navigateur web.

III.3.7.4 Une base de données MySQL pour notre serveur

Nous allons besoin de stocker les informations des comptes de nos utilisateurs qui vont utiliser notre site web pour contrôler notre robot. Pour cela, on utilise le plus souvent des bases de données. Donc, nous allons installer MySQL qui est un **SGBDR** (Système de Gestion de Bases de Données Relationnelles), logiciel permettant la **gestion** de quantités massives de données. Ce SGBDR libre utilise le langage SQL.

Passons donc à l'installation de celui-ci:

Pour ce fait, nous allons tout simplement **installer le serveur MySQL** mais aussi un package qui permettra de faire le lien entre PHP et notre base de données.

En tapant donc cette commande dans notre terminal :

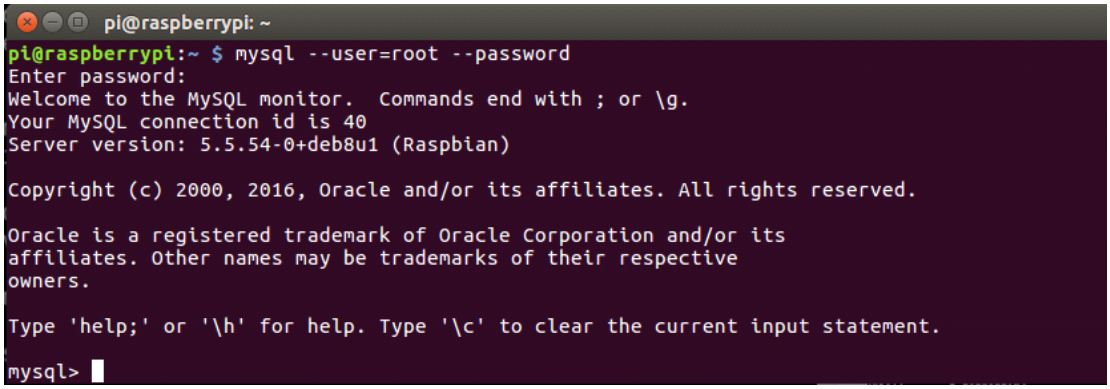
sudo aptitude install mysql-server php5-mysql

Au cours de l'installation de MySQL, plusieurs informations vont vous être demandées (dont un mot de passe pour le compte administrateur MySQL).

Cette fois, pour vérifier le bon fonctionnement de MySQL, nous n'auront besoin que d'un terminal :

mysql --user=root --password= notre mot de passe

Si tout se passe bien, vous verrez apparaître quelque chose du type :



III.3.7.5 Les différentes étapes pour conception de notre site web PHP

Nous allons héberger quatre scripts PHP dans notre serveur Apache dans le dossier «/var/www/html/RobotRPi».

1. L'utilisateur ne doit normalement pas appeler cette page directement, mais il y a une règle d'or lors de la création des sites web et vous devez toujours vous rappeler ceci : «**Ne faites jamais confiance à l'utilisateur**. En d'autres termes, ne pensez jamais que l'utilisateur va toujours faire ce que vous pensez qu'il va faire. Nous donc ajoutons quelques sécurités au début du code PHP pour m'assurer que l'utilisateur passe un nom l'utilisateur et mot de passe correcte pour accéder à notre site. Donc, notre site est géré par l'administrateur, celui qui va créer des comptes pour tous les utilisateurs qui veulent contrôler notre robot via le web.

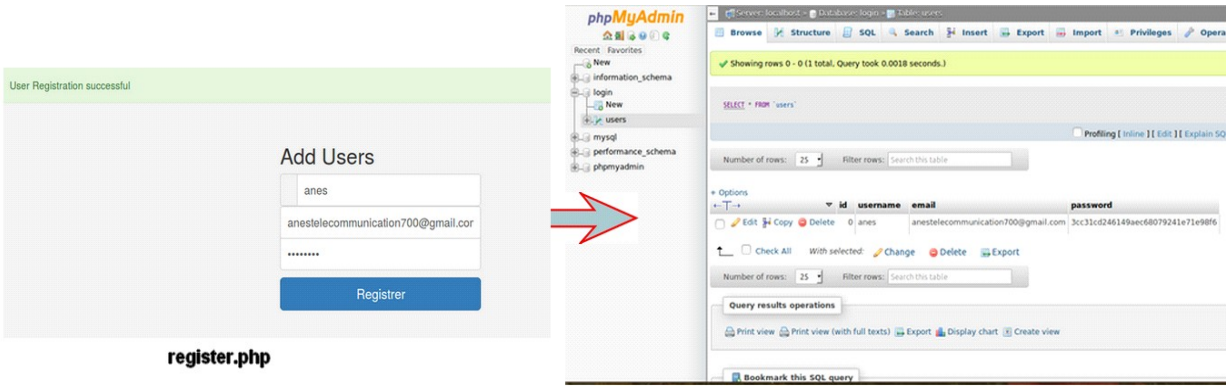


Figure III-57. La page register.php de l'administrateur.

Code source de la page register.php

```
Terminal File Edit View Search Terminal Help
 GNU nano 2.2.6                File: register.php                Modified

<?php
require_once('connect.php');
if(isset($_POST) & !empty($_POST)){
    $username= mysqli_real_escape_string($connection, $_POST['username']);
    $email=mysqli_real_escape_string($connection, $_POST['email']);
    $password= md5($_POST['password']);

    $sql = "INSERT INTO `users` (username, email, password) VALUES ('$username' , '$email' ,'$password')";
    $result= mysqli_query($connection, $sql);
    if($result){
      $smsg= "User Registration successful";
}else{
      $fmsg= "User Registration failed";
}
}
?>
<!DOCTYPE html>
<html>
<head>
<title> Users Registration </title>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
<link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
<div class="container">
        <?php if(isset($smsg)){ ?><div class="alert alert-success" role="alert"> <?php echo $smsg; ?> </div>
          <?php }?>
          <?php if(isset($fmsg)){ ?><div class="alert alert-success" role="alert"> <?php echo $fmsg; ?> </div>
          <?php }?>
 <form class="form-signin" method="POST">
    <h2 class="form-signin-heading">Add Users</h2>
        <div class="input-group">
          <span class="input-group-addon" id="basic-addon1"></span>
              <input type="text" name="username" class="form-control" placeholder="Username" required>
        </div>
         <label for="inputEmail" class="sr-only">Email address</label>
        <input type="Email" name="email" id="inputEmail" class="form-control" placeholder="Email address" required autofocus>
        <label for="inputPassword" class="sr-only">Password</label>
        <input type="password" name="password" id="inputPassword" class="form-control" placeholder="Password" required>
        <button class="btn btn-lg btn-primary btn-block" type="submit">Registrer</button>
    </form>
 </div>
</body>
</html>>
```

On a utilisé la fonction de hachage md5 pour hacher les mots de passe des utilisateurs qui seront enregistrés dans la base de données ou utilisés pour l'authentification des comptes.

2. Création de la base de données :

Nous allons créer une base de données nommée **login** en exécutant la requête suivante sur le SGBD MySQL : create database 'login' ;

3. Création de la table users :

Nous allons donc créer une table users contient quatre champs " id, username, email, password " dans notre base de données pour que l'administrateur puisse ajouter les utilisateurs qui auraient les droits pour contrôler le robot.

```
CREATE TABLE `users` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`username` varchar(255) NOT NULL,
`email` varchar(255) NOT NULL,
`password` varchar(255) NOT NULL,
PRIMARY KEY (`id`),
UNIQUE KEY `username` (`username`)
);
```

4. Script connect.php :

Ce script php est chargé d'établir une connexion avec notre base de données "login" à distance qui se trouve sur le serveur MySQL.

```
pi@raspberrypi: /var/www/html/RobotRPi
  GNU nano 2.2.6              File: connect.php

<?php
$connection = mysqli_connect('localhost', 'root', 'Pa$$w0rd');
if(!$connection){
    die("Database Connection Failed" .mysqli_error($connection));
}
$select_db = mysqli_select_db($connection, 'login');
if(!$select_db){
    die("Database Selection Failed" .mysqli_error($connection));
}
?>
```

5. Démarrage du site web (index.php)

Nous avons d'abord besoin d'un fichier «**index.php**» (l'extension est **.php** et non pas **.html** car nous allons utiliser du code PHP, elle permet au serveur de savoir qu'il y a du code PHP à exécuter avant d'envoyer la page générée). Ce fichier index.php est la page HTML que les utilisateurs verront en premier en accédant à notre site. Le processus est indiqué par la figure suivant :

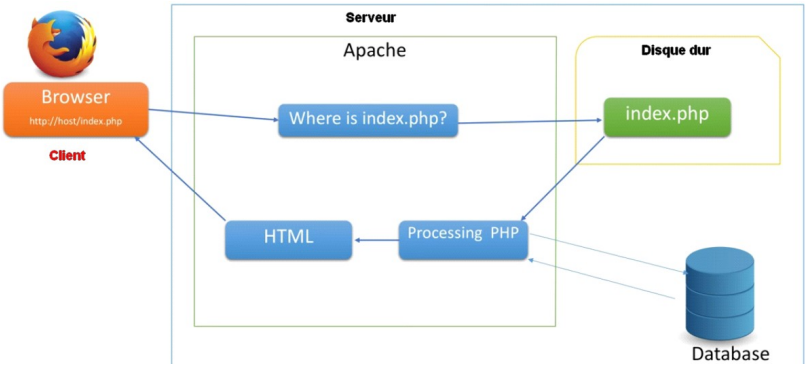


Figure III-58. Fonctionnement du site web PHP.

6. Script d'authentification des utilisateurs (login.php)

Il serait intéressent également de mettre en place un système d'authentification afin de sécuriser l'accès à l'application web qui sert à contrôler notre robot.

Au niveau de cette page web, on fait une authentification avec un nom d'utilisateur et un mot de passe c.à.d. on interroge notre base de données de tester le couple (username, password) ; s'il est

correct on accède à la page commander.php par contre, si ce n'est pas le cas, un message d'alert apparaît.

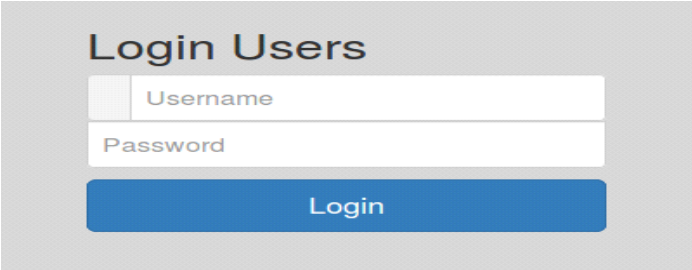


Figure III-59. La page login.

7. Page de contrôle du roobt (commander.php)

Si l'utilisateur légitime déjà introduit le bon username et password, la page web **commnder.php** sera apparue pour contrôler le robot et de visualiser la vidéo de la webcam.

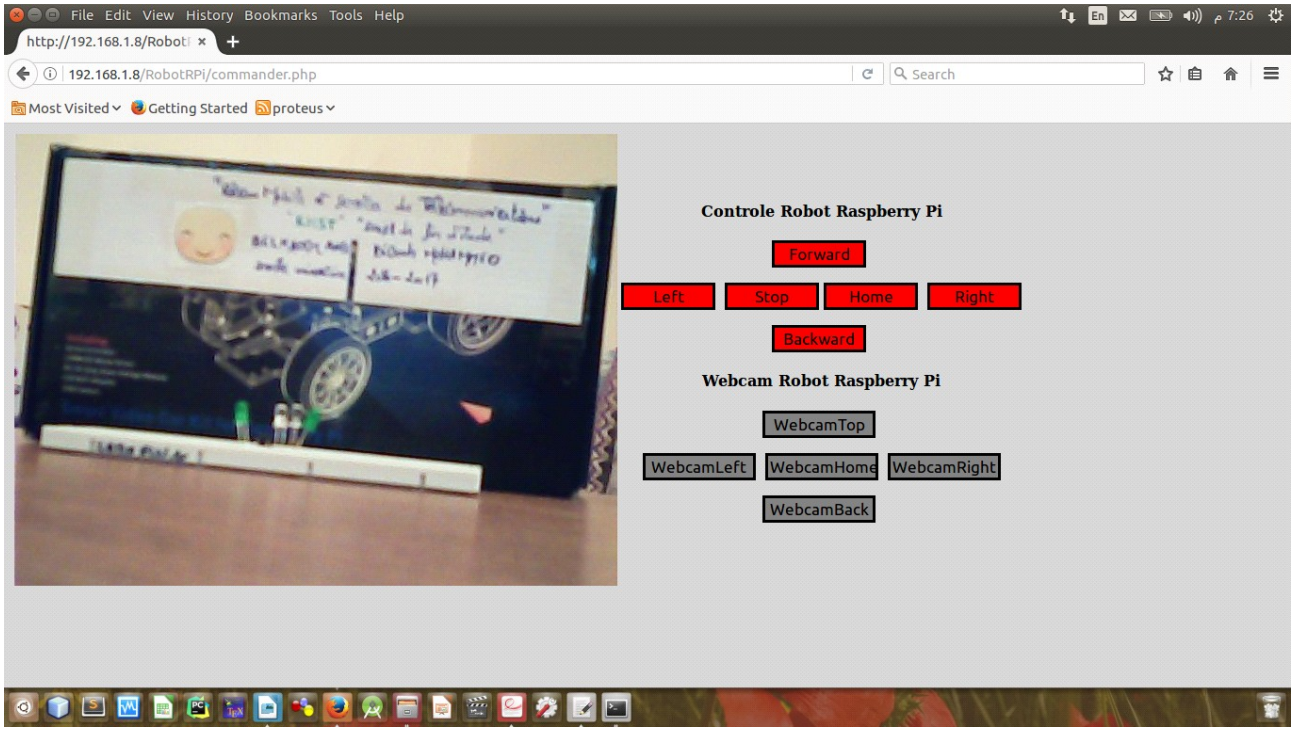


Figure III-60. Interface web PHP.

Il faut tout d'abord que Python soit installé sur le serveur et que l'utilisateur **www-data** a le **droit** de lancer les scripts Pythons sinon ceux-ci ne se lançerons pas par sécurité. Pour cela, éditez le fichier **/etc/sudoers** via la commande : **sudo nano /etc/sudoers.**
Maintenant au bas de ce fichier, entrons les deux lignes suivantes :

pi ALL = (ALL) NOPASSWD : ALL
www-data ALL= (ALL) NOPASSWD : ALL
Une fois terminé, nous sauvegardons et nous quittons en appuyant sur CTRL + X puis sur Y.

Fonction system() en php qui va lancer les scripts python :

PHP est capable d'exécuter un script python grâce à la fonction prédéfinie en PHP system() ou exec(). Mais avant cela, il faut donner les droits d'exécution des scripts python par la commande suivante : **sudo chmod +x script.py**

III.4 Application de capture d'images via le robot

Le streaming vidéo a bien évolué et est maintenant possible même pour des images très haute définition, qui contiennent beaucoup d'informations. Cela veut dire que vous pouvez, au moins, toujours voir ce que le robot est en train de faire. Vous pouvez aussi traiter les images sur un ordinateur à distance, en utilisant le flux vidéo venant du robot.

Notre objectif est d'écrire via la bibliothèque OpenCV et Tkinter avec Python une application pour capturer des images depuis notre flux vidéo lorsque le robot est en train de faire exploration de l'environnement.

Tout d'abord, il faut installer OpenCV en tapant sur le terminal du Raspberry les commandes :

sudo apt-get install python-opencv
pip install imutils
sudo apt-get install python-imaging-tk

Figure III-61. Repertoires et script Python de notre application.

Ce script Python est utilisé seulement pour initialiser à la fois le VideoStream et la class ImagesRobot (cette classe est incluse dans le dossier RobotPi sous forme de script photoapp.py).

```
from __future__ import print_function
from RobotPi.photoapp import ImagesRobot
from imutils.video import VideoStream
import argparse
import time


ap = argparse.ArgumentParser()
ap.add_argument("-o", "--output", required=True,
        help="path to output directory to store snapshots")
ap.add_argument("-p", "--picamera", type=int, default=-1,
        help="whether or not the Raspberry Pi camera should be used")
args = vars(ap.parse_args())


print("[INFO] webcam Robot...")
videostream = VideoStream(usePiCamera=args["picamera"] > 0).start()
time.sleep(2.0)


pba = ImagesRobot(videostream, args["output"])
pba.root.mainloop()
```

Exécution du script capture_image.py :

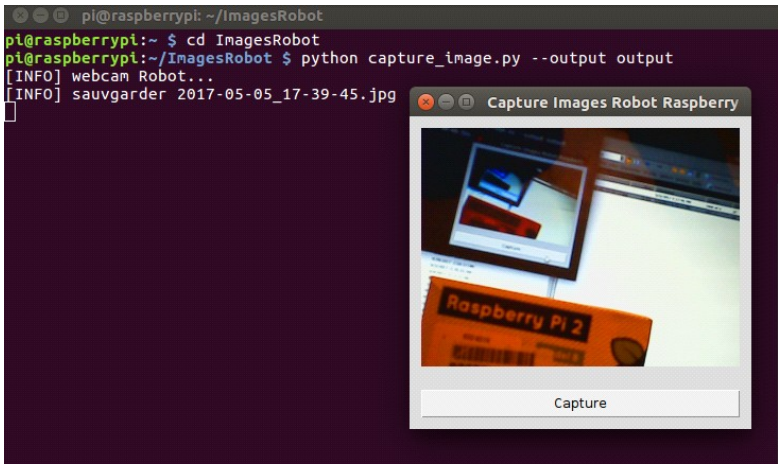


Figure III-62. Application de capture images.

Cette interface comporte deux éléments. Le premier, en bas, qui est le bouton de capture. Chaque fois que ce bouton est cliqué, la trame actuelle lue à partir du flux vidéo sera stockée dans le répertoire output

Le deuxième élément, placé directement au-dessus du premier, est un affichage en direct du flux vidéo lui-même.

III.5 Conclusion

Dans ce chapitre, on a appris comment accéder à notre Raspberry à distance via WiFi pour contrôler les GPIOs et donc le pilotage du robot et de visualiser la vidéo de la webcam par le biais d'une page web ou application Android.

Dans la première partie, nous avons vu comment contrôler le robot à travers une application desktop en utilisant les sockets dans les deux côtés client et serveur.

Dans la deuxième partie, nous avons illustré les différentes interfaces graphiques de notre application Android créée avec le logiciel App Inventor 2 qui permet de commander notre robot en utilisant le framwork Django.

Dans la troisième partie, on a vu aussi comment contrôler le robot via page web PHP. De plus nous avons vu aussi comment se connecter à la base de données et comment administrer les comptes des utilisateurs.

Enfin, nous avons vu comment gérer les traitements d'images de la webcam via une application desktop à travers les bibliothèques Tkinter et openCV sous la langage Python afin d'apprendre à récupérer des images d'une vidéo.

Conclusion générale

Ce projet nous a permis de découvrir et de consolider nos connaissances en électronique et en informatique. Il nous a aussi permis de découvrir la technologie Raspberry Pi qui, une fois maîtrisée, permet à son utilisateur de l'utiliser à des fins multiples, toutes plus intéressantes les unes que les autres.

L'avantage est que nous possédons à présent un bagage de connaissances plus important sur le sujet, nos connaissances en informatique (Linux, Python, développement web et I2C) ainsi qu'en électronique et en mécanique ayant été fortement consolidées. On peut donc dire que le travail réalisé nous a été grandement profitable.

Le projet était vraiment très intéressant mais assez dur à réaliser étant donné que nous partions de zéro. Nos connaissances en programmation (tels que Python, PHP et Django) et en électronique n'étaient pas suffisantes au début du projet, et nous avons dû effectuer beaucoup de recherches avant de pouvoir réaliser notre travail. Et cela a demandé du temps et un investissement personnel assez important dans le but de comprendre correctement le fonctionnement de ce projet.

En perspective, nous pensons à embarquer un système de reconnaissance dans notre robot autonome qui soit capable de prendre des décisions tout seul. Nous pouvons aussi adapter un système de suivi d'objets consiste a réaliser un robot mobile suiveur de ligne de couleur sombre sur un terrain de couleurs claire. Ils constituent déjà une nette possibilité d'amélioration du projet que nous avons effectué.

Enfin, ce projet était une occasion de nous initier au domaine de l'embarqué. Ce dernier qui est devenu un grand domaine de recherche aujourd'hui.

Bibliographie

[1] BOULKROUNE Ramzi, Les systèmes embarqués, Université de Annaba-Ingénieur d'état en électronique option télécommunication 2009.
[2] TAYARI LASSAAD, SUPPORT DE COURS SYSTEMES EMBARQUES, pp 6.
[3] Fondation de l'École polytechnique sur youtube, consulté le début de Février 2017.
[4] TAYARI LASSAAD, SUPPORT DE COURS SYSTEMES EMBARQUES, pp.13.
[5] Francis Cottet, Emmanuel Grolleau, Sébastien Gérard, Jérôme Hugues, Yassine Ouhammou, Sara Tucci-Piergiovanni.SYSTÈMES TEMPS RÉEL EMBARQUÉS, 2 e édition, pp.1,2,3 et 6.
[6] TAYARI LASSAAD, SUPPORT DE COURS SYSTEMES EMBARQUES, pp 9.
[7] Olivier Cardin – IRCCyN, Nantes, Damien Trentesaux – LAMIH, Valenciennes, Les systèmes cyber-physiques de production.
[8] http://connect.ed-diamond.com/ , consulté le mois d'Avril 2017.
[9] https://www.fun-mooc.fr/c4x/MinesTelecom/04013/asset/S4-5_-Objets-communicants.pdf, Objets communicants & Internet des Objets.
[10] https://www-01.ibm.com/software/fr/data/bigdata/enterprise.html
[11] Matt Richardson et Shawn Wallace, À LA DÉCOUVERTE DU Raspberry Pi
[12] Christian Dupaty, Raspberry Pi Installation-Configuration, Interfaces de Communications. BTS systèmes numériques http://genelaix.free.fr
[13] Yann Morère, Atelier Arcade, http://www.graoulab.org/wp/wpcontent/uploads/2015/12/Atelier_arcade.pdf, 2015.
[15] Matt Richardson et Shawn Wallace. À LA DÉCOUVERTE DU Raspberry Pi, pp.12 à 20.
[16] http://profs.cmaisonneuve.qc.ca/hbenameurlaine/linux/linux1/05_866.pdf.
[17] Matt Richardson et Shawn Walla. À LA DÉCOUVERTE DU Raspberry Pi.
[18] http://educ8s.tv/raspberry-pi-gui-tutorial-create-your-own-gui-graphical-user-interface-with-tkinter-and-python/, consulté le 29 Février 2017.
[19] https://fr.wikipedia.org/wiki/Pont_en_H
[20] http://coursdivers.blogspot.com
[21] https://wiki.mchobby.be/index.php?title=AdaFruit_PWM_Driver
[22] http://the-raspberry.com/wifi-config
[23] https://openclassrooms.com, Développez votre site web avec le framework Django
[24] https://raspbian-france.fr/installer-serveur-web-raspberry/

Annexe

Les scripts Python qui permet contrôler les GPIO du Raspberry Pi

Script Python	Code source

forward.py	`#!/usr/bin/env python`
	`import RPi.GPIO as GPIO`
	`import video_dir`
	`import car_dir`
	`import motor`
	`import time`
	`from time import ctime`
	`motor.setup()`
	`spd = 50`
	`motor.setSpeed(spd)`
	`print 'motor forward'`
	`motor.forward()`
	`time.sleep(2)`
	`motor.stop()`

backward.py	`#!/usr/bin/env python`
	`import RPi.GPIO as GPIO`
	`import video_dir`
	`import car_dir`
	`import motor`
	`import time`
	`from time import ctime`
	`motor.setup()`
	`spd =50`
	`motor.setSpeed(spd)`
	`print 'motor backward'`
	`motor.backward()`
	`time.sleep(2)`
	`motor.stop()`

left.py	`#!/usr/bin/env python`
	`import RPi.GPIO as GPIO`
	`import video_dir`
	`import car_dir`
	`import motor`
	`from time import ctime`
	`car_dir.setup()`
	`car_dir.home()`
	`print 'robot left'`
	`car_dir.turn(-200)`

right.py	`#!/usr/bin/env python`
	`import RPi.GPIO as GPIO`
	`import video_dir`
	`import car_dir`
	`import motor`
	`from time import ctime`
	`car_dir.setup()`
	`car_dir.home()`
	`print 'robot right'`
	`car_dir.turn(200)`

webcamright.py

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
import video_dir
import car_dir
import motor
from time import ctime
video_dir.setup()
video_dir.calibrate(-240, -150)
```

webcamback.py

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
import video_dir
import car_dir
import motor
from time import ctime
video_dir.setup()
video_dir.calibrate(200, 0)
```

webcamhome.py

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
import video_dir
import car_dir
import motor
from time import ctime
video_dir.setup()
video_dir.calibrate(-240, 0)
```

webcamleft.py

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
import video_dir
import car_dir
import motor
from time import ctime
video_dir.setup()
video_dir.calibrate(-240, 150)
```

webcamtop.py

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
import video_dir
import car_dir
import motor
from time import ctime
video_dir.setup()
video_dir.calibrate(-150, 0)
```

Fonction system() en php qui va lancer les scripts python

```php
<?php
if (!empty($_POST)) {
    if (isset($_POST['forward'])) {
        system('sudo python ./ordres/forward.py');

    }   elseif (isset($_POST['stop'])) {
        system('sudo python ./ordres/stop.py'):

    }   elseif (isset($_POST['left'])) {
        system('sudo python ./ordres/left.py');
    }
        elseif (isset($_POST['right'])) {
          system('sudo python ./ordres/right.py');
    }
        elseif (isset($_POST['backward'])) {
        system('sudo python ./ordres/backward.py');
    }
         elseif (isset($_POST['webcamtop'])) {
        system('sudo python ./ordres/webcamtop.py');
    }
        elseif (isset($_POST['webcamhome'])) {
        system('sudo python ./ordres/webcamhome.py');
    }
        elseif (isset($_POST['webcamleft'])) {
          system('sudo python ./ordres/webcamleft.py'
    }
        elseif (isset($_POST['webcamright'])) {
        system('sudo python ./ordres/webcamright.py')
    }
        elseif (isset($_POST['webcamback'])) {
        system('sudo python ./ordres/webcamback.py');
    }
        elseif (isset($_POST['home'])) {
        system('sudo python ./ordres/home.py');
  }
}?>
```

Titre Commande à distance d'un robot mobile via un Smartphone Android.

Résumé Le but majeur des systémes embarqués vise à fusionner le monde virtuel de l'informatique avec le monde physique de l'électronique afin d'assurer la fiabilité, rapidité, et la flexibilité des ressources du monde réel en temps réel avec moins des contraintes.

Dans ce contexte, ce projet de fin d'étude consiste à commander un robot mobile à distance via WiFi depuis une application Android et aussi depuis un site web. Le robot a été agrémenté d'un système embarqué disposant d'une webcam mobile pour la diffusion vidéo en temps réel.

Mots-clés systémes embarqués, Raspberry Pi, I2C, Android, Django, Python, PHP, MySQL, WiFi.

Title Remote control of a mobile robot via an Android smartphone.

Abstract The major goal of embedded systems is to merge the virtual computing world with the physical electronics world to ensure the reliability, speed, and flexibility of real world resources in real time with less constraints.

In this context, this end-of-study project consists of ordering a remote mobile robot via WiFi from an Android application and also from a website. The robot has been enhanced with an embedded system with a mobile webcam for real-time video broadcasting.

Keywords embedded systems, Raspberry Pi, I2C, Android, Django, Python, PHP, MySQL, WiFi.

الملخص

الهدف الرئيسي من الانظمة المدمجة هو دمج العالم البرمجي مع العالم المادي من الالكترونيات لضمان موثوقية, سرعة ومرونة موارد العالم الحقيقي في الزمن الحقيقي مع أقل العوائق.
وفي ظل هذا السياق, مشروع نهاية الدراسة الخاص بنا يختصر في السيطرة على روبوت عن بعد عبر تطبيق اندرويد وكذلك من طرف عميل عبر شبكة الواي فاي. الروبوت معزز بكاميرا ويب لبث فيديو في الوقت الحقيقي

الكلمات المفتاحية

الانظمة المدمجة, WiFi, Raspberry Pi, Android, PHP, MySQL, Django.