Table des matières

Chapitre 3 : La programmation et la commande de Mahfoud III

Chapitre 4 : L'application mobile de Mahfoud III

Liste des figures

Liste des tableaux

Glossaire

MCC :	Moteur à courant continu
DSP :	Digital Signal Processor
MCU :	multipoint control unit
ADC :	Analog-to-Digital Converter
CAN :	Controller Area Network
UART :	Univers Asychrenus Receive Transmitter
SPI :	Serial Peripheral Interface
PWM :	Pulse Width Modulation
MIPS :	Million Instructions Per Second
DMA :	Date Memory Acess
CPU :	Central Processing Unit
RAM :	Random Acess Memory
JTAG :	Joint Test Action Group
ICD :	In-Circuit. Debugger
EEPROM :	Electrically-Erasable Programmable Read-Only Memory
IC:	Input Capture
ICI:	Input Capture Interrupt
ICM:	Input Capture Mode
FCY:	Instruction Cycle Rate
CTS:	Clear To Send
RTS:	Request To Send
I²C:	Inter Integrated Circuit
ECAN:	Enhanced Controller Area Network
UART:	Universal Asynchronous Receiver Transmitter
USB:	Universal Serial Bus
ASCII:	American Standard Code for Information Interchange

Introduction générale

La robotique mobile vise à rendre une machine autonome, c'est-à-dire lui donner les capacités de perception, de décision et d'action pour agir de manière autonome sans assistance ni intervention humaine avec son environnement, c'est un axe de recherche à la croisée de plusieurs disciplines scientifiques et techniques tel que la mécanique, l'électronique, l'électrotechnique et l'informatique…etc.

Les études de la commande des robots mobiles à roues remontent aux années 80.Ces robots mobiles sont destinés à réaliser les taches considérés comme pénibles ou les taches s'accomplissant dans des milieux hostiles à l'être humain comme les milieux marin (tel que la recherche de la boite noire des avions), spatiale (tel que l'exploration planétaire)…etc. Le robot peut aussi être utile dans le domaine médical pour compenser les handicapes physiques.

Un véritable robot doit pourvoir se débrouiller dans un environnement partiellement observable, stochastique, dynamique et continu, ce dernier n'a de raison d'être que s'il effectue des taches sans supervision humaine, c'est pour cette raison que l'autonomie des robots est devenue un sujet très étudié dans la robotique moderne.

Le robot MAHFOUD fut l'un des premiers projets de développement d'un robot mobile autonome à l'université Abou Bekr Belkaid, il est doté de deux moteurs à courant continu, trois capteurs numériques de distance, deux codeurs incrémentaux, il comporte aussi 3 cartes une pour la commande, une pour le contrôle et une autre pour l'alimentation notre but de ce travail est d'améliorer la version précédente du robot pour qu'il agisse mieux avec son environnement, il comporte maintenant :

- Une structure mécanique
- Deux moteurs à courant continu
- Deux codeurs incrémentaux
- Un capteur ultrason

- Une seule carte qui regroupe à la fois la commande et le contrôle
- Une application mobile pour récupérer les données internes du mobile ainsi que des informations de positionnement et pour le commander aussi
- Et enfin une batterie d'alimentation

Le nom du robot est évidement MAHFOUD III.

Le mémoire est organisé en 4 chapitres :

Chapitre 1 : donne un bref aperçu sur l'état de l'art de la robotique mobile.

Chapitre 2 : parle de la réalisation de Mahfoud III, nous détaillerons les composants de la carte électronique du robot.

Chapitre 3 : aborde la commande de Mahfoud III.

Chapitre 4 : consacré à la programmation de Mahfoud III, nous parlons de la programmation de la carte électronique et nous présentons aussi l'application mobile du robot.

Chapitre 1 :

Etat de l'art

Introduction :

La robotique est le domaine de l'intelligence artificielle qui donne aux automates la possibilité d'interagir avec le monde physique, le premier robot mobile autonome a été créé par Grey Walter en 1948, c'était une tortue capable de se diriger vers les sources de lumière qu'elle percevait, cependant ce robot n'était pas programmable, autour des années 50 la mise en place des robots était possible grâce à la création des transistors et des circuits intégrés, la première chaine de production complètement robotisé était ouverte par Nissan en 1972.

Dans ce chapitre nous parlons de l'historique des robots, les différentes configurations qui existent pour ces derniers, on présente aussi brièvement des méthodes utilisées pour la localisation des robots mobile

1. Définition d'un robot mobile :

Un robot est un agent physique qui réalise des taches dans l'environnement dans lequel il évolue, c'est un automate doté de capteurs et d'actionneurs qui lui offrent la capacité de déplacement et d'adaptation proche de l'autonomie. [1]

Afin d'être autonome le robot doit mettre en œuvre :

- Un ensemble d'actionneurs : ces actionneurs permettent au robot de se mouvoir en d'autres termes ils servent uniquement à appliquer des forces physiques à l'environnement parmi les actionneurs on trouve : les bras, les roues, les pinces…etc.

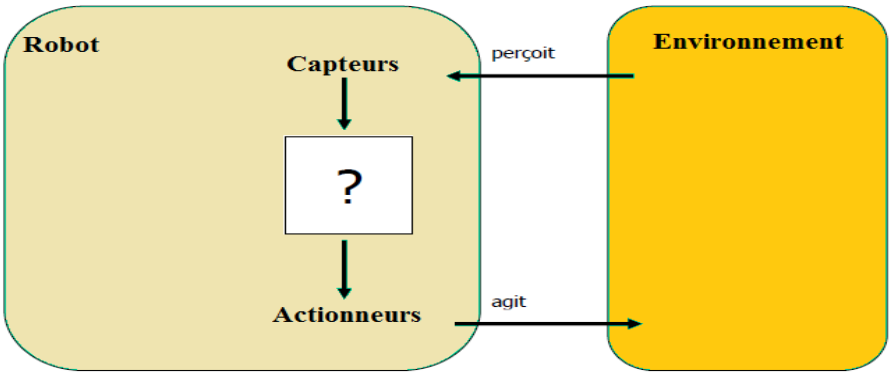


Figure 1.1 : Boucle de décision d'un robot

- Un ensemble de capteurs : ces capteurs permettent au robot de percevoir l'environnement dans lequel il évolue, la robotique utilise plusieurs types de capteurs de deux grandes catégories : [2]
  - ✓ Capteurs proprioceptifs : fournissent des informations sur l'état interne du robot (capteurs de position GPS, capteurs de vitesse…etc.)
  - ✓ Capteur extéroceptifs : fournissent des informations sur l'environnement dont lequel le robot se déplace (boussole, radar, caméra,…etc.)

Un robot mobile doit aussi posséder une certaine intelligence sous entendu par algorithme ou régulateur, qui lui permette de calculer les commandes reçus par les capteurs et envoyés aux actionneurs pour réaliser une tache donnée à partir des données reçus par les capteurs.

2. La disposition des roues d'un robot mobile :[3]

Le choix des roues et de leur disposition donne au robot son mode de locomotion, il existe principalement trois types de roues pour les robots mobiles :

- ✓ Des roues fixes dont l'axe de rotation passe par le centre de la roue.
- ✓ Des roues centrées orientables, dont l'axe d'orientation passe par le centre de la roue.
- ✓ Des roues décentrées orientables, appelés roues folles, pour ce type de roue l'axe d'orientation ne passe pas par le centre de la roue.

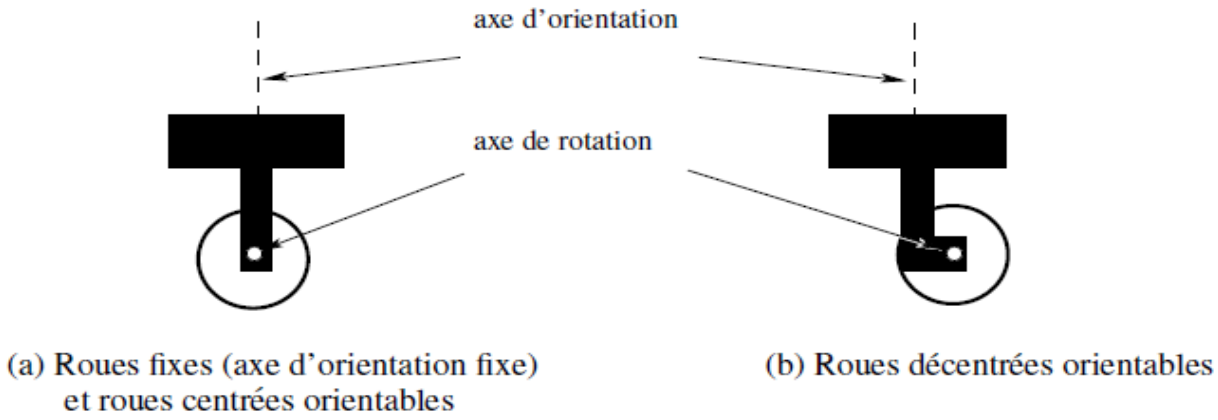


Figure 1.2 : Disposition des roues d'un robot mobile

3. Les différentes configurations des robots mobiles à roues :

Le choix de la disposition des roues d'un robot mobile est primordial, elle peut limiter la mobilité du robot comme elle peut la bloquer, par exemple les robots dotées de deux roues fixes non parallèle ne peut pas se déplacer en ligne droite.

Afin d'assurer une bonne mobilité du robot sans glissement des roues sur le sol, l'ensemble des roues du robot doivent avoir un point seul autour duquel le robot peut tourner instantanément, ce point est déligné par « le centre instantané de rotation soit le CIR ».

Il est nécessaire que le point d'intersection des axes de rotations des différentes roues soit unique car les points de vitesse nulles liés aux roues se trouvent sur les axes de rotation, a cause de cette raison il existe en pratique trois principales catégories de robots mobiles à roues qu'on va présenter dans ce paragraphe.

3.1. Configuration d'un robot mobile à commande différentielle : [4]

Un robot est dit à commande différentielle si le déplacement de ce dernier est assuré par le biais de deux moteurs d'entrainement indépendant, chacun d'eux est lié à une roue motrice
La direction du robot peut être changé en jouant sur les vitesses des deux roues motrices, les roues folles sont des roues libres, elles ont pour rôle d'assurer la stabilité de la plate-forme du robot.

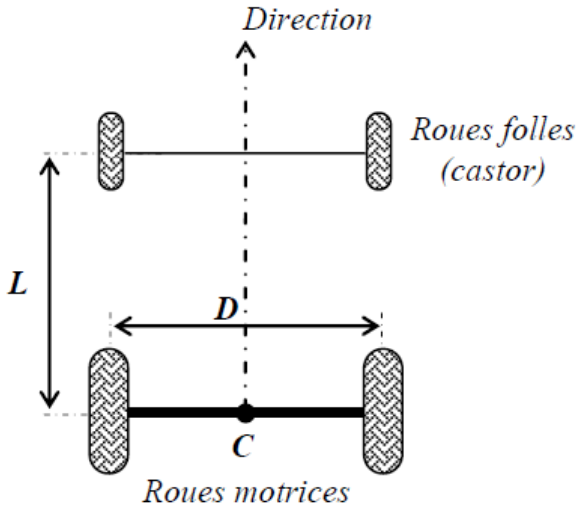


Figure 1.2 : Configuration d'un mobile à commande différentielle

3.2. Configuration unicycle : [4]

Un robot mobile est dit unicycle s'il est actionné par deux roues indépendantes et qu'il possède un certain nombre de roues folles qui assurent la stabilité. La figure (1.3) montre le schéma d'un robot de type unicycle. On y a omis les roues folles, qui n'interviennent pas dans la cinématique, dans la mesure où elles ont été judicieusement placées. Ce type de robot est très répandu en raison de sa simplicité de construction et de propriétés cinématiques intéressantes.

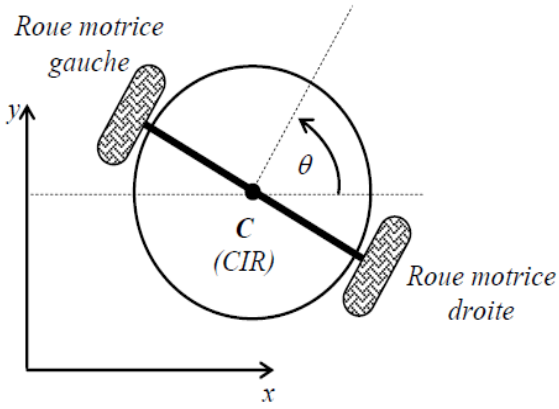


Figure 1.3 : Configuration d'un unicycle

3.3. Configuration tricycle : [4]

Le robot à configuration tricycle illustré dans la figure (1.4), se compose de deux roues passives arrière et une roue motrice avant et vice versa.

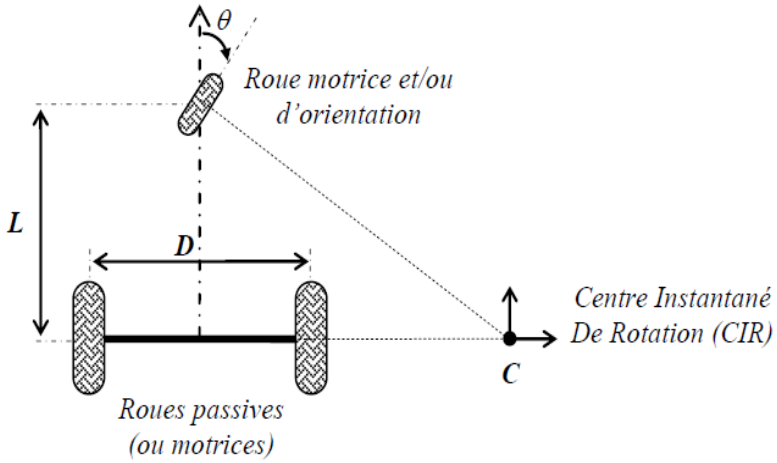


Figure 1.4 : Configuration tricycle

Le mouvement du robot conféré par deux actions : la vitesse longitudinale et l'orientation de la roue orientable. Pour la localisation de ce type de robot, un capteur d'orientation est associé à la roue orientable, et deux codeurs sont associés aux roues motrices.

3.4.Les voitures ou configuration dite d'Ackerman : [5]

Utilisé dans l'industrie automobile, dans cette configuration la roue avant intérieure tourne d'un angle légèrement étroit par rapport aux roues extérieures dans un virage ce qui réduit le glissement des pneus.

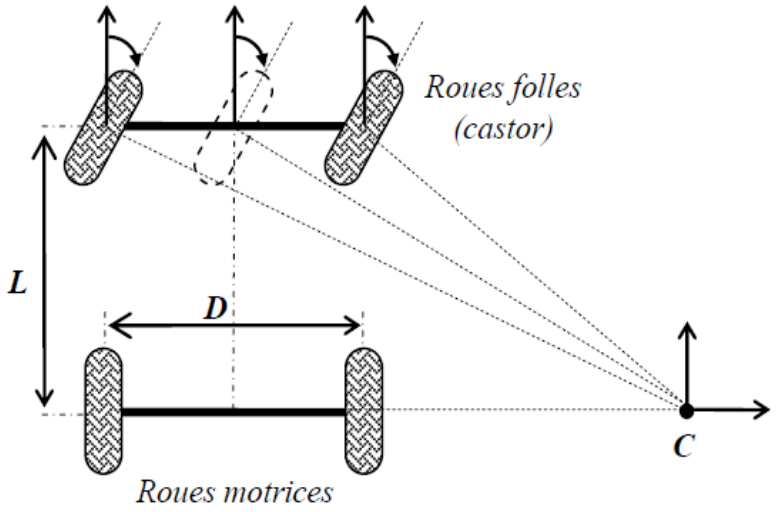


Figure 1.5 : Configuration voiture

3.5.Robot a traction synchrone : [5]

La traction synchrone est une technique utilisée pour minimiser l'effet de glissement et augmenter la force de traction. Cette configuration comporte trois roues ou plus, couplés de façon qu'elles tournent toutes en même direction et à la même vitesse. La synchronisation mécanique peut se faire par différentes méthodes, la plus utilisé est à travers une courroie.

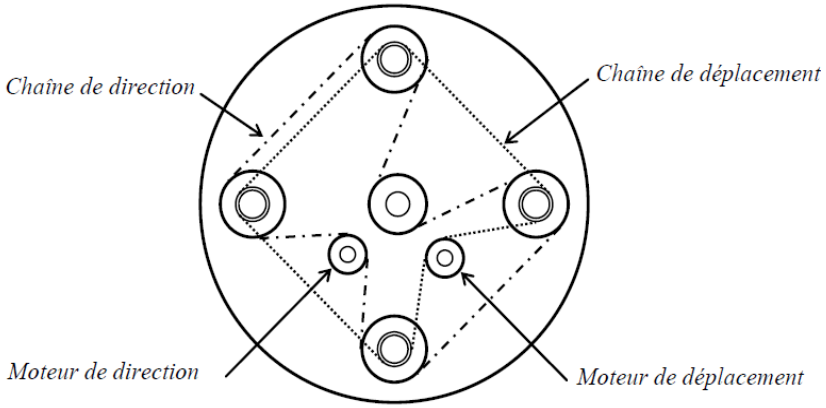


Figure 1.5 : Configuration d'une traction synchrone

4. Localisation d'un robot mobile :

Pour que le robot soit autonome en réalisant certaines taches données il doit pourvoir se localiser dans son environnement, par exemple, un robot doit se déplace d'un point A à un point B, la localisation de ce dernier est l'un des problèmes majeurs de la robotique mobile, pour que le robot soit capable de réaliser se déplacement il doit obligatoirement savoir sa position actuelle pour calculer sa trajectoire afin d'aller à la position cible B, en plus de son déplacement le robot doit mettre a jour sa trajectoire en calculant la position tout au long de son déplacement, et ceci pour savoir s'il est arrivé a sa destination cible ou pas, on va présenter brièvement dans cette partie les moyens de localisation les plus utilisés, le chapitre 2 va aborder en détail les méthodes de localisation utilisées pour le robot Mahfoud III.

4.1. L'odométrie :

La méthode de navigation la plus utilisée pour la localisation d'un robot mobile, cette méthode offre une bonne précision a court terme, le principe de cette méthode est d'intégrer l'information de mouvement incrémental au fil du temps, le point faible de cette méthode est l'accumulation d'erreurs d'orientation, qui va engendrer des erreurs importante sur la position, cette dernière augmente proportionnellement a la distance parcourue, malgré son point faible cette méthode reste présente est préférée par les chercheurs. [5]

4.2. Capteurs de distance :

Les télémètres ou les capteurs de distance sont très utilisé eux aussi dans la robotique mobile, ces capteurs donnent la possibilité d'avoir l'information de distance entre l'obstacle le plus proche et le robot ; pour ces capteurs il existe deux technologies de capteurs de distance :

- Les capteurs utilisant les ultrasons : comme son nom l'indique ces capteurs reposent sur l'utilisation des ondes acoustiques dont la fréquence est trop élevée pour être audible par l'être humain ; c'est les ultrasons.
- Les capteurs utilisant la lumière : ces capteurs détectent l'intensité de la lumière captée.

4.3.Les caméras : [2]

Désigné par la localisation visuelle, cette méthode consiste à identifier des amers et à se localiser en utilisant par exemple une carte de caractéristiques, cette dernière représente l'environnement par les coordonnés globales d'amers (points, lignes, polygones…etc.), ou par une carte topologique, qui caractérise l'environnement par un graphe s'intéressant entre les différentes régions de l'environnement.

5. Conclusion :

Dans ce chapitre nous avons vu un bref aperçu sur la robotique mobile, son utilisation, les différents configurations qui existent pour les robots mobiles, on a aussi parlé de façon peu détaillée sur quelques méthodes utilisées pour la localisation d'un robot mobile, une étude plus détaillée sera abordée dans les chapitres suivants sur ce point.

Chapitre 2 :

Réalisation pratique de Mahfoud III

Introduction :

On détaillera dans ce chapitre la réalisation pratique de Mahfoud III, les différents composants électroniques utilisés dans la réalisation de la carte de commande pour rendre le robot plus autonome que la version précédente (Mahfoud II).

Le schéma et la carte de commande ont été réalisés avec l'aide de notre encadreur.

La carte de commande (Annexe 1) comporte le microcontrôleur dsPIC programmable en langage C et flashable à l'aide d'un programmateur PICkit2, elle comporte aussi les modules de paissance, le module de liaison série USB.

1. Configuration choisie pour Mahfoud III :

On a gardé la même configuration de la version précédente du robot fournit. Il est issu d'un ancien projet de l'EPFL mais nous n'avons gardé que la structure mécanique, les moteurs et les codeurs. Ce robot a une configuration tricycle à commande différentielle, composé de deux roues motrices et d'une roue libre (roue folle).

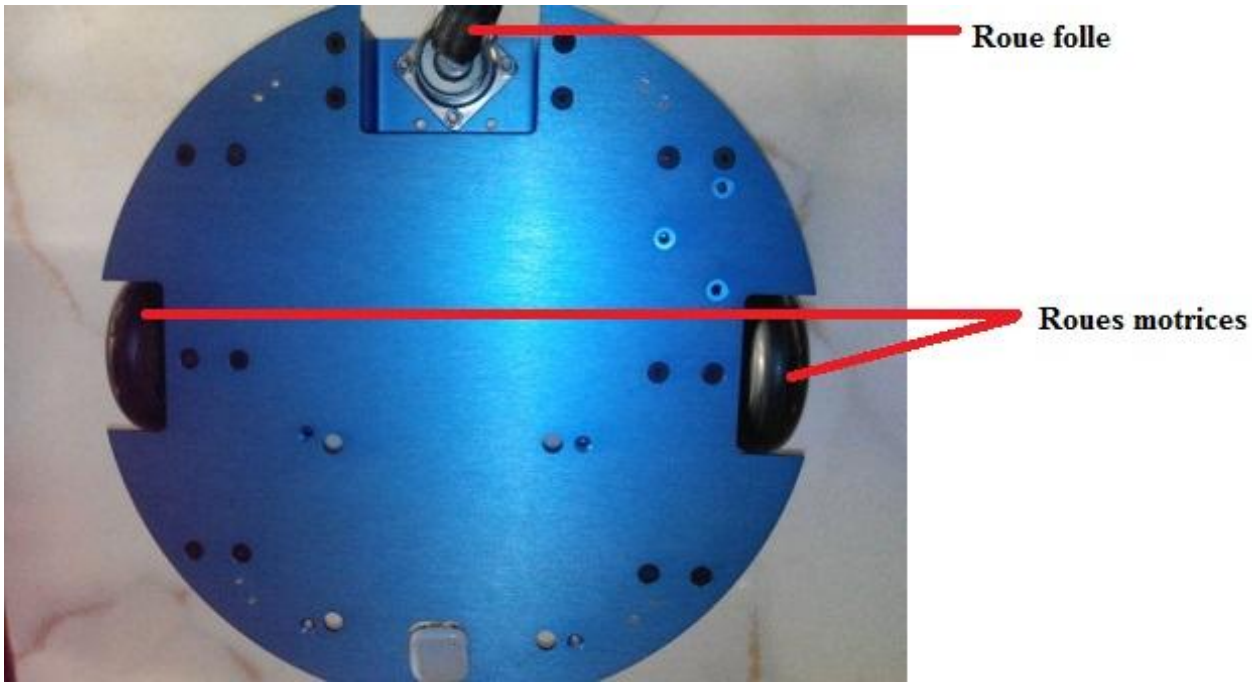


Figure 2.1 : configuration du robot Mahfoud III

2. Moteurs de Mahfoud III :

2.2.Description mécanique de la MCC :

Le couplage entre deux champs magnétique est le principe de base de tous les moteurs électriques, la transformation de l'énergie électrique en énergie mécanique s'opère a travers ce couplage, de ce principe on comprend que tout moteur comporte deux circuits magnétiques dont la partie fixe est appelée stator et la partie mobile appelé rotor.

Dans le cas du moteur étudié qui est un moteur à courant continu, le stator (dit aussi inducteur) crée le champ magnétique tandis que le rotor (dit induit) est alimenté en courant continu, les conducteurs de ce derniers sont soumis aux champs créé par le stator ce qui crée une force, et c'est cette force qui fait tourner le rotor ce qui crée le couple moteur.

2.3.Stator de la MCC : [6]

Le stator d'une MCC est composé de la carcasse du moteur et d'un circuit magnétique, ce dernier crée un champ magnétique dans le rotor, grâce au stator le flux électromagnétique circule.

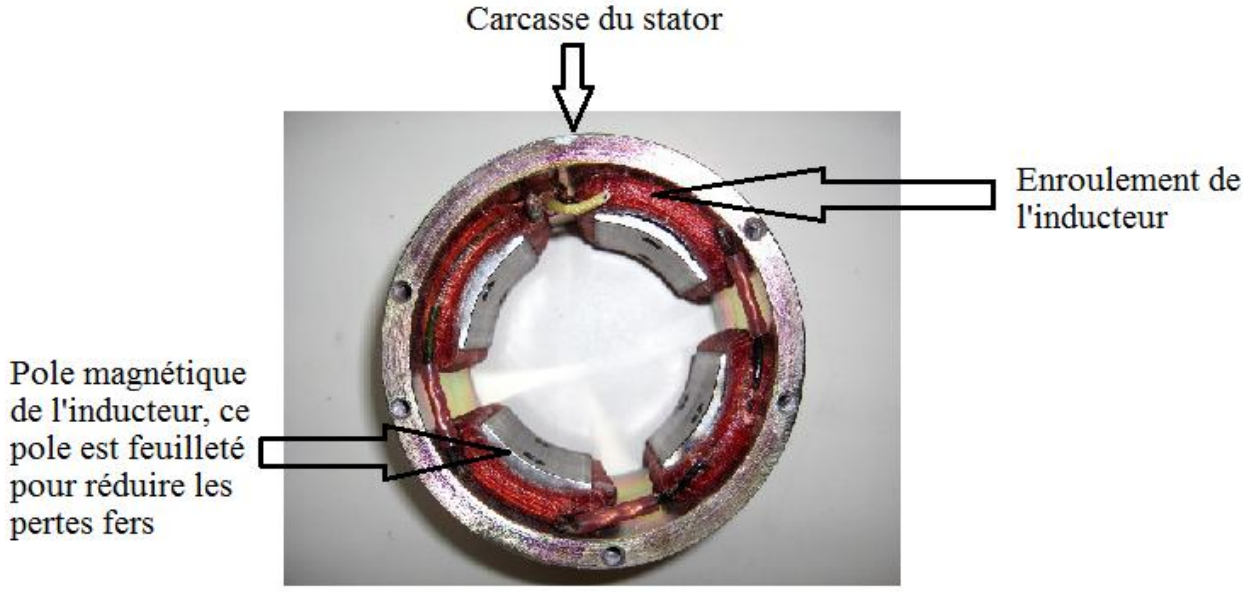


Figure2.2 : Stator d'une MCC

L’ensemble des paires de pole de la MCC est constitué d’un empilement de tôles ferromagnétiques.

Les enroulements de l’inducteur servent à créer le champ magnétique.

2.4.Rotor de la MCC :[6]

Le rotor du moteur à courant continu est composé d'un axe sur lequel est empilé un ensemble de disques ferromagnétiques, il comporte aussi des bobines conductrices (en cuivre isolé verni) bobinées dans les encoches des tôles.

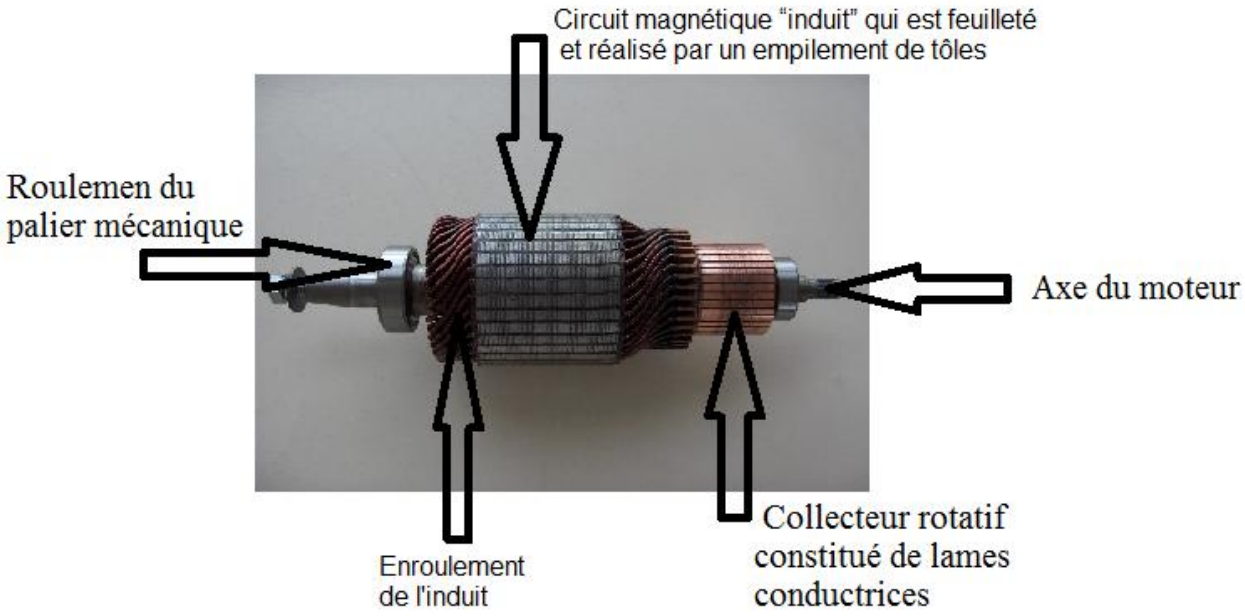
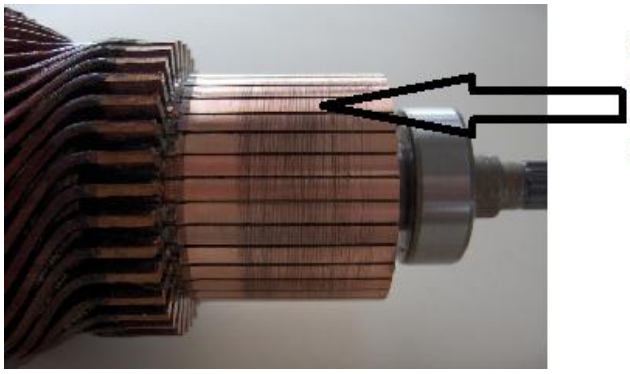


Figure2.3 : Rotor d’une MCC

Le collecteur : constitué de lames conductrices (figure 2.3), permettant d’alimenter les bobines, à chaque fois le collecteur alimente une bobine, le rotor tourne pour que la bobine suivante soit alimentée aussi de sorte que le champ magnétique du rotor reste perpendiculaire au champ magnétique du stator pour que la transmission du couple soit optimale.



Le collecteur est constitué de lames de cuivre isolées entre elles. L'isolant est de la micanite. L'ensemble balais d'alimentation + collecteur assure la liaison entre les conducteurs tournants et la partie fixe.

Figure2.4 : Lame conductrices rotor d'une MCC

2.5.Description électrique de la MCC :

La MCC est un convertisseur d'énergie réversible, elle peut fonctionner soit en moteur ou en génératrice :

- Fonctionnement moteur : dans ce cas l'énergie électrique est convertie en énergie mécanique.

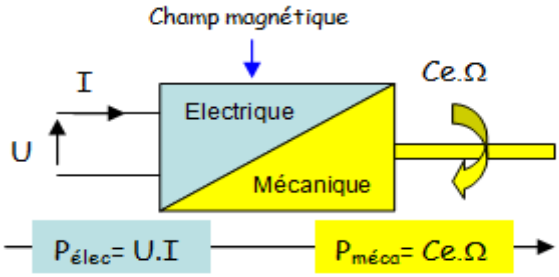


Figure 2.5 : fonctionnement moteur d'une MCC

- Fonctionnement génératrice : dans ce cas l'énergie mécanique est convertie en énergie électrique.

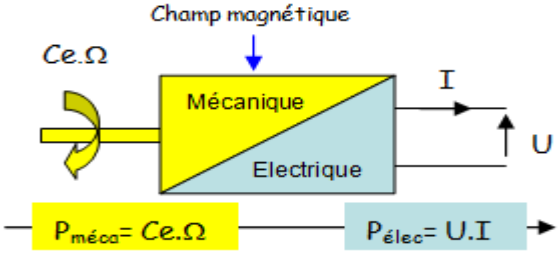


Figure 2.6 : fonctionnement génératrice d'une MCC

2.6.Principe de fonctionnement de la MCC :

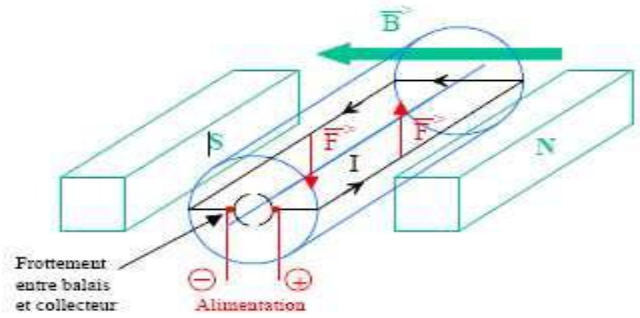


Figure 2.7 : Principe de fonctionnement d'une MCC

Le principe de fonctionnement est donné pour une spire.

L'aiment permanant possède les pôles nord (N) et sud (S), ces pôles créent un champ magnétique B dans le moteur. La spire est alimentée et plongée dans le flux, elle est soumise à la force de Laplace (c'est la force électromagnétique qu'exerce un champ magnétique sur un conducteur parcouru par un courant), le moteur alors se met en rotation, on dit donc qu'il ya création d'un couple moteur.

On peut augmenter la valeur du couple moteur soit:

- ✓ En augmentant le nombre de spires
- ✓ En augmentant le nombre de paires d'aimants

2.7.Les moteurs de Mahfoud III :

Le robot Mahfoud III, est construit avec deux moteurs à courant continu, un pour chaque roue motrice, les moteurs utilisés sont des moteurs A-max de 12V. (Annexe 2)

3. Les roues de Mahfoud III :

Pour que le robot Mahfoud III puisse circuler on a utilisé des roues d'un patinage a roulette de chez Skate Line, les roues sont de diamètre de 72 mm.

Figure 2.8 : les roues de Mahfoud III

Alimentation de Mahfoud III :

Le robot Mahfoud III est alimenté par une batterie au plomb rechargeable de 12V, cette batterie peut tenir une heure sous 2A. Elle a donc une capacité de 2000 mAh.

4. Double pont-H L298 :

Pour notre robot Mahfoud III, on veut commander ses deux moteurs à courant continu, on est donc obligé de faire varier leur vitesse (positive et négative). Pour ce faire, on a choisi le double pont H, le L298. (Voir Annexe 3)

4.1.Description du L298 :

Le L298 de chez STMicroelectronics©, est un module de 2 hacheurs 4 quadrants (figure 2.9) dans un circuit intégré ce circuit intégré permet de piloter des charges inductives tel que les relais, les moteurs pas à pas et aussi des moteur à courant continu ce qui le cas pour nous, il permet de piloter directement sa vitesse dans les deux sens de rotation, et ce qu'il a d'avantage de plus c'est que pour contrôler les deux moteurs à courant continu un seul L298 suffit.

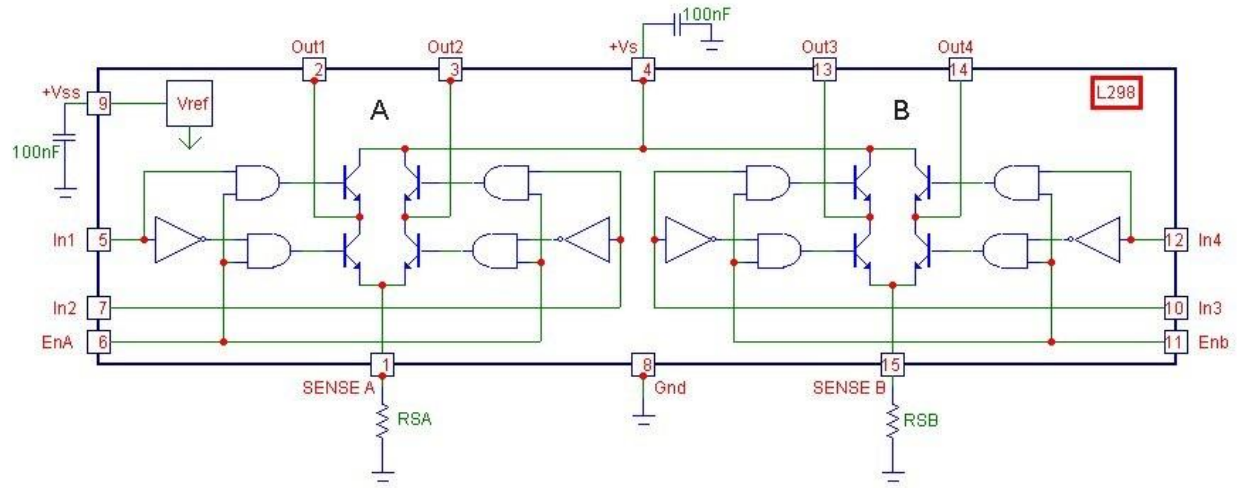


Figure 2.9 : Le schéma interne du L298

Sur le schéma de la figure 2.9 on voit que le L298 se compose de deux étages de puissance configurés en pont, chacun de ces étages est commandé par deux entrées logiques, ainsi que deux entrées de validation In1, In2. [7]

4.2.Fonctionnement du L298 :

Selon le schéma interne du L298 (figure 2.10) ce module possède deux broches d'alimentation :

- Une pour l'alimentation du moteur V_s
- Et une autre pour l'alimentation de la logique interne V_{ss}

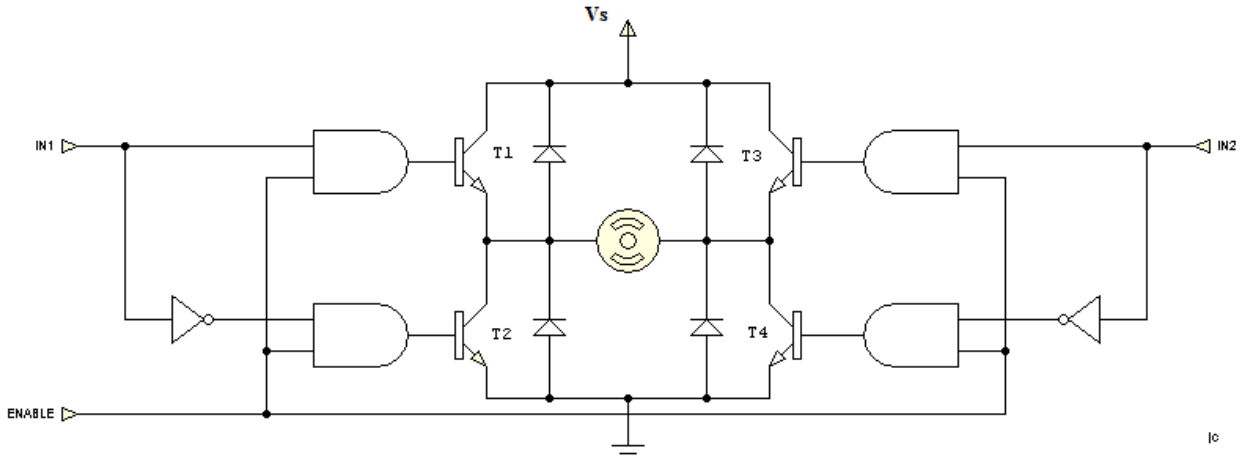


Figure 2.10 : MCC associé à un L298

L'alimentation du moteur se fait par blocage ou saturation des transistors, ce qui lui permet de tourner dans un sens ou l'autre comme le montre le tableau ci-dessous, les diodes servent à protéger les transistors :

Les entrées		La sortie
En=1	In1=1 / In2=0	Sens 1 de rotation
	In1=0 / In2=1	Sens 2 de rotation
	In1= In2	Arrêt rapide du moteur
En=0	In1=1 / In2=0	Moteur libre (arrêt)

Tableau 2.1 : La sortie du L298 en fonction des entrées

5. Convertisseur USB vers série MCP2200 :

Afin de pouvoir connecter le Smartphone a la carte pour réaliser certaines fonctions, il nous a fallu un module de connexion USB, celui qu'on a utilisé c'est le MCP2200 de chez$MicroChip^{©}$. (Voir annexe 4).

Le MCP2200 est un convertisseur série USB vers UART qui permet une connectivité USB dans les dispositifs qui disposent d'une interface UART. Le MCP2200 comporte également 256 octets d'EEPROM intégrée.

6. Capteurs infrarouges GP2Y0A21YK0F:

Pour que le robot Mahfoud III évite les obstacles lors de son parcours de façon autonome, on avait le choix entre lui associer 3 capteurs infrarouges de type GP2Y0A ou bien un capteur ultrason HC-SR04, afin de justifier notre choix pour les capteurs infrarouges on va présenter le fonctionnement de chacun des capteurs : le capteur infrarouge et le capteur ultrason.

6.1.Le capteur infrarouge : [8]

Le capteur infrarouge se compose d'un émetteur et d'un récepteur comme tout autre capteur.

L'émission s'effectue par une diode électroluminescente infrarouge (LED), la réception se fait par une photodiode (ou phototransistor) sensible au flux lumineux rétrodiffusé par l'émetteur.

Les performances d'un tel capteur données par la figure 2.11, ils sont définis par:

- ✓ La distance « l » entre l'émetteur et le récepteur
- ✓ L'inclinaison des axes émetteur/récepteur
- ✓ L'ouverture du faisceau à l'émission et à la réception

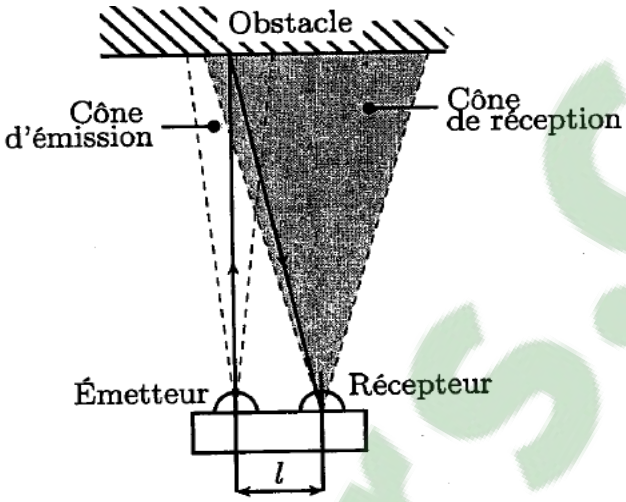


Figure 2.11 : Paramètres d'un capteur infrarouge

Ces paramètres déterminent les performances du capteur, On fait une mesure avec la LED infrarouge éteinte et une avec la LED infrarouge allumée. S'il n'y a aucun obstacle proche, la valeur lue est la même. Sinon, l'obstacle aura réfléchi la lumière infrarouge et la deuxième mesure donnera un résultat plus élevé.

Le capteur infrarouge peut être utilisé en capteur de distance en mesurant l'angle avec lequel le rayon réfléchi arrive sur le récepteur. En fonction de la distance entre l'émetteur et le récepteur, on peut en déduire la distance de l'obstacle.

Par rapport à un capteur ultrasons, un capteur infrarouge permettra une meilleure précision mais au détriment d'une largeur de cône de détection moins importante et d'une portée plus faible. Le capteur infrarouge est sensible aux sources de lumière qui contiennent un fort rayonnement infrarouge et également à la couleur et la nature des obstacles.

6.2.**Le capteur ultrason :**[2]

Comme le capteur infrarouge, le capteur ultrason est composé d'un émetteur et d'un récepteur. L'émetteur envoie des ultrasons réfléchis par l'obstacle et récupérable par le récepteur, la distance de l'obstacle est évaluée en évaluant le temps entre l'émission et la réception c'est à dire le temps de propagation de l'onde.

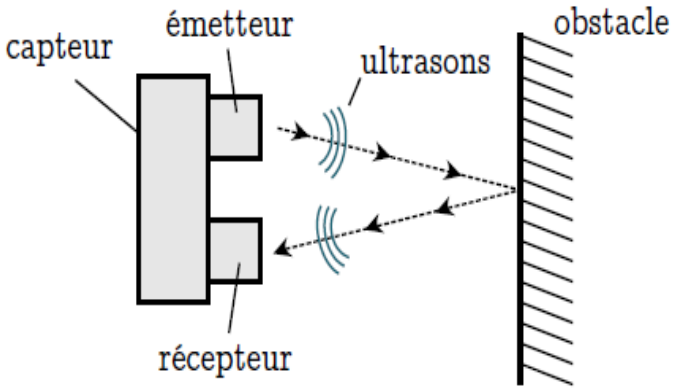


Figure 2.12 : principe de fonctionnement d'un capteur ultrason

L'onde émise par l'émetteur d'un capteur ultrason présente l'une des caractéristiques de ce type de capteur, car cette dernière n'est pas unidirectionnelle, elle prend la forme d'un cône d'environ 30 degrés (figure ci-dessous), ce qui est un point à la fois fort et faible pour ce capteur.

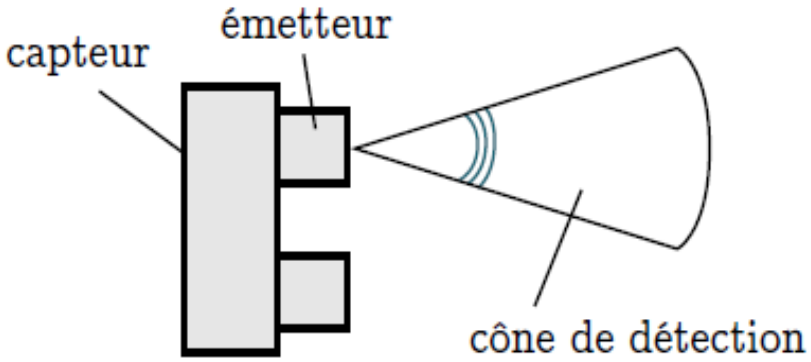


Figure 2.13 : Cône d'un capteur ultrason

L'avantage d'une part d'avoir le cône, se réside dans la taille de la zone détectée. L'obstacle ne doit pas nécessairement être en face du capteur mais dans son cône de détection, cela a pour avantage de détecter les petits obstacles.

D'autre part l'inconvénient, c'est que la détection ne sera pas précise vue que l'obstacle n'est pas détecté en un point ce qui est illustré dans la figure 2.14 dans lequel les arcs de cercle représente les obstacles détectés.

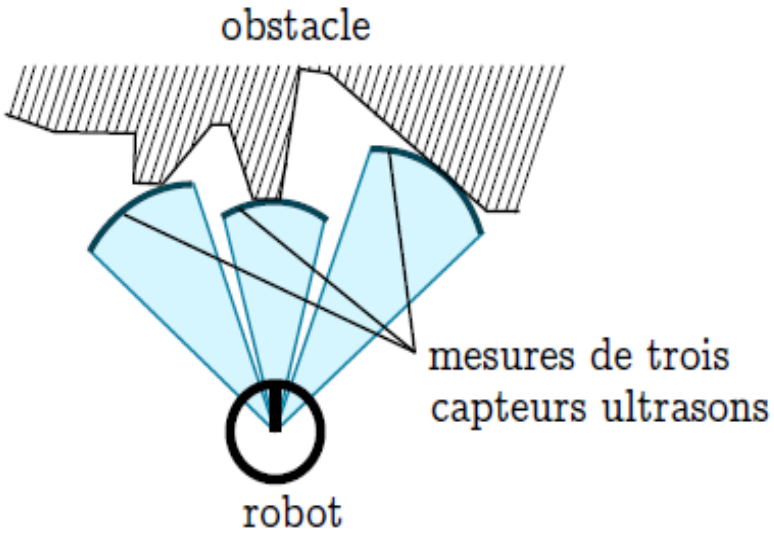


Figure 2.14 : Association de 3 capteurs ultrasonique,

6.3.Comparaison entre le capteur infrarouge et le capteur ultrason :

Le capteur infrarouge	Le capteur ultrason
La portée	
La portée et de 5 cm à 80 cm	Quelques mètres en général
La précision	
Dépend de la distance, excellente à 10 cm, elle régresse de plus en plus jusqu'à 80cm	Dépend de la mesure précise du temps de propagation de l'onde sonore
La sensibilité	
Ces capteurs IR ont une modulation qui les affranchit normalement de l'éclairage ambiant	les capteurs ultrasons sont sensibles à la température et à la pression. Mais il y a plus grave : Ils sont aussi sensibles aux autres appareils utilisant les mêmes

	fréquences, comme les téléobjectifs à ultrasons, ou tout simplement les autres robots !
Le coût	
Moins cher que les capteurs ultrason	Leur cout est légèrement plus important par rapport aux capteurs infrarouges

Tableau 2.2 : Comparaison entre l'US et l'IR

On a fait des tests avec un capteur ultrason et le graphe suivant illustre la distance vu par le capteur en fonction du temps d'un mur qui se trouve a une distance de 4m, on remarque que pour la même distance donnée de (4m) le capteur donne de faux résultats avec une marge d'erreur importante :

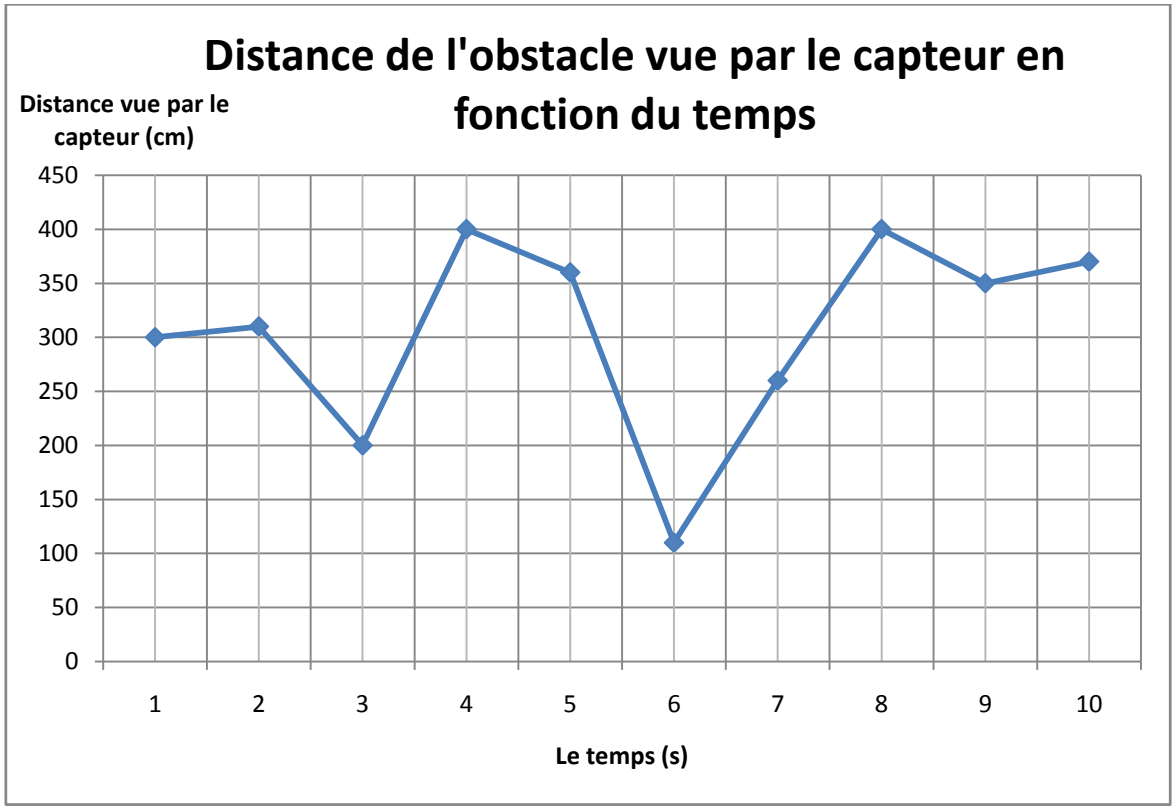


Figure 2.15 : distance vue par le capteur ultrason d'un mur à proximité de 4m

Après avoir fait ce test, on a testé les capteurs infrarouges qui ont donné de meilleurs résultats, et pour cela on a choisi de les intégrer dans notre robot.

L’erreur est certainement due à une erreur dans la programmation que nous ne sommes pas arrivés à trouver, car les capteurs us sont bien plus précis que ça.

6.4.Présentation du capteur infrarouge GP2Y0A21YK0F:

Le module utilisé (Voir annexe 5) donne une mesure de 10 à 80 cm. il embarque toute l’électronique nécessaire à son fonctionnement. De ce faite une simple interface 3 fils suffit pour le faire marcher (GND, VCC, OUT)

7. **Les codeurs incrémentaux MEnc 13**: [10]

Les interrupteurs et les détecteurs de position, ces systèmes de détection conventionnels fournissent des informations de type TOR (tout ou rien). A des endroits prédéterminés, Ainsi ils ne répondent que partiellement aux besoins de précision et de flexibilité.

Les codeurs permettent une maitrise entière du positionnement du mobile, ces codeurs permettent donc de donner des informations sur la position, la vitesse et le positionnement du robot.

Les codeurs incrémentaux (ou les capteurs de déplacement), sont d’un usage très général, notamment dans les problèmes d’asservissement, de mesure et de contrôle de nombreux types de machines.

Le codeur optique est un dispositif électromécanique délivrant sous forme numérique, un signal de mesure déplacement linéaire ou angulaire. Il existe deux types de codeurs rotatifs :

- ✓ Le codeur absolu :qui intègre son propre compteur. Ce genre de capteur est généralement calibré et initialisé une seule fois, et il conserve normalement sa valeur lors de l'arrêt de l'appareil. C'est le cas des compteurs kilométriques des automobiles à la différence du "compteur journalier" qui peut être remis à zéro par l'utilisateur.
- ✓ Le codeur incrémental (ou générateurs d’impulsions) : qui ajoute ou soustrait (selon le sens de rotation) une unité à un compteur à chaque rotation supérieure à la résolution du capteur. Le compteur est généralement remis à zéro lorsque l'appareil est allumé. C'est le cas de la souris d'ordinateur à boule.

Pour notre robot on a utilisé deux codeurs incrémentaux rotatif MEnc 13.(Voir annexe 6)

7.1.Description du codeur MEnc 13: [11]

Le codeur MEnc13 de chez *Maxon* ©est un dispositif qui convertit le mouvement dans une séquence d'impulsions numériques. En comptant un seul bit ou en décodant un ensemble de bits, les impulsions peuvent être converties en des mesures de position.

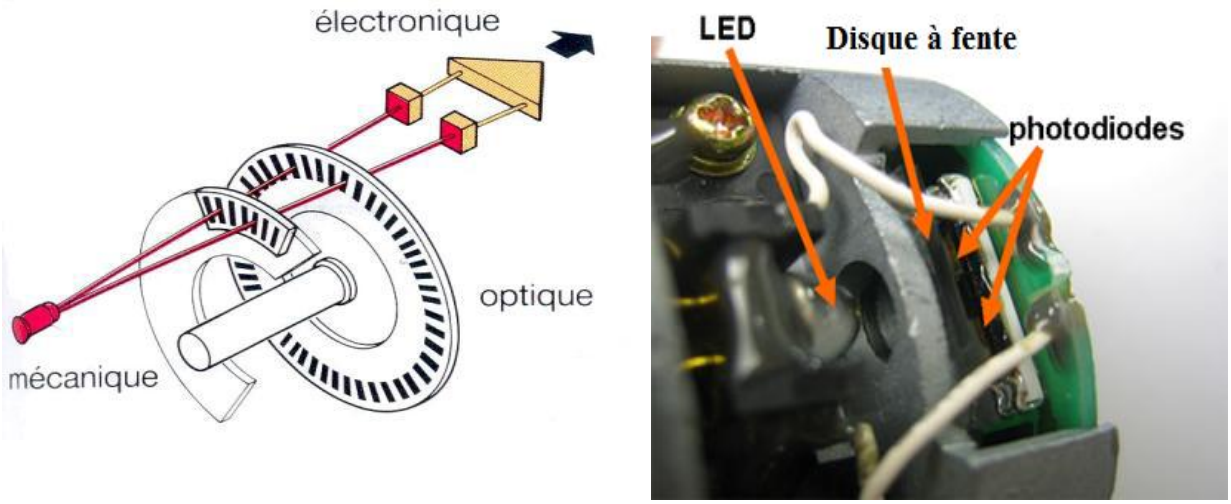


Figure 2.16 : Schéma de principe d'un MEnc13

Le schéma principal du codeur montre que le capteur est composé d'un émetteur qui est la diode électroluminescente (LED), et d'un récepteur photosensible, il comporte aussi un disque qui sera lié mécaniquement a la machine, ce disque comporte des zones opaques et transparentes successivement.

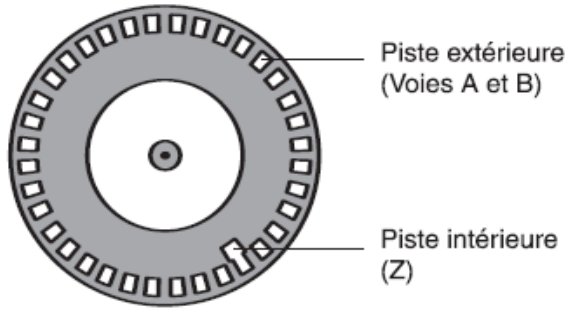


Figure 2.17 : Vue d'un disque gradué d'un codeur incrémental rotatif

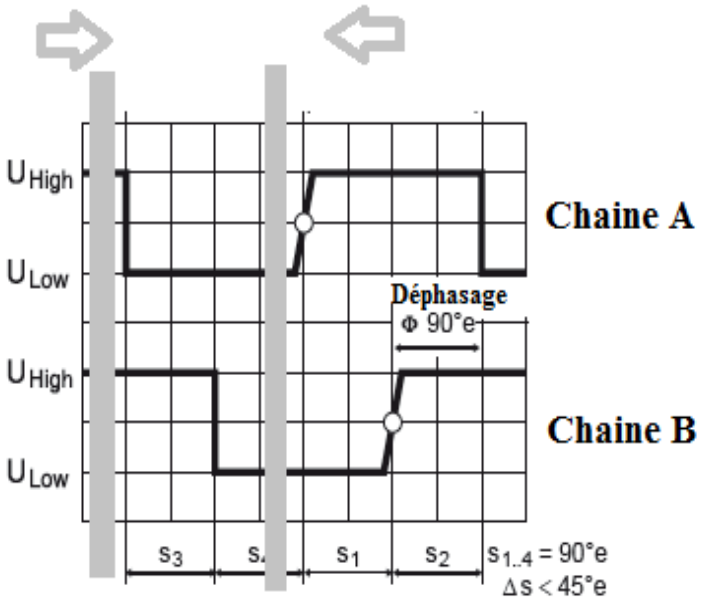
Le disque de ce type de codeur rotatif comporte lui-même deux pistes :

- Une piste extérieure (Voies A et B) : cette piste est divisée en « n » intervalles d'angles alternativement opaques et transparents.
- Une piste intérieur (Z) : le signal de cette piste est appelé top zéro, elle comporte une seule fenêtre transparente, ce signal est le signal de référence et il permet la réinitialisation a chaque tour.

7.2.Principe de fonctionnement du capteur MEnc13: [12]

Ce codeur fonctionne selon le principe du balayage photoélectrique. La rotation du disque gradué en fonction du déplacement de l'objet à contrôler (dans notre cas les roues du robot Mahfoud III) génère un flux lumineux émis par une diode électroluminescente est modulé et les cellules photoélectriques qui captent la lumière à la sortie du disque délivrent un signal carré.

La résolution qui est définie par le nombre d'impulsions par tour correspond au nombre de graduation sur le disque, plus le nombre de ces points est important plus le nombre de mesures par tour donnera une division plus fine du déplacement ou de la vitesse du mobile.



Il existe des photodiodes décalées installées derrière la piste extérieure délivrant des signaux carrée A et B comme la montre la figure ci-contre, chaque fois que le faisceau lumineux traverse une zone transparente.
Les signaux des canaux A et B sont en quadrature, et leur déphasage définit le sens de rotation : le signal B est à 1 pendant le front montant de A, alors que dans l'autre sens, il est à 0.

Figure 2.18 : Formes d'ondes de sortie du MEnc13

8. DSPIC 33FJ :

L'organe de l'intelligence du robot est le microcontrôleur, c'est grâce à lui qu'on peut commander le robot.

Le microcontrôleur utilisé est un *dsPIC33FJ128MC802* fabriqué par Microchip. Il dispose de 28 pins dont 21 pins pour les E/S, toutes ses 28 pins sont utilisés (voir Annexe 1), on l'utilise a 29.4912 MHz ce qui correspond 29.4912 MIPS (Million

d'Instructions Par Seconde).La fréquence d'horloge maximum autorisée est de 40 MHz.

8.1.Présentation du dsPIC33F :

Le *dsPIC33FJ128MC802* (Voir annexe 7) est un microcontrôleur à noyau DSP, donc un DSC (Digital Signal Controller), 16 bits, possédant un jeu de 83 instructions, chacune stockée dans un seul mot (16 bits) de programme et est exécutée en 1 cycle.

Comme tout DSC, chaque broche du dsPIC33F à une ou plusieurs fonctions comme le montre la figure 2.16, le dsPIC33F possède 28 pins montés sur un boitier SPDIP (de largeur 300 MILS).

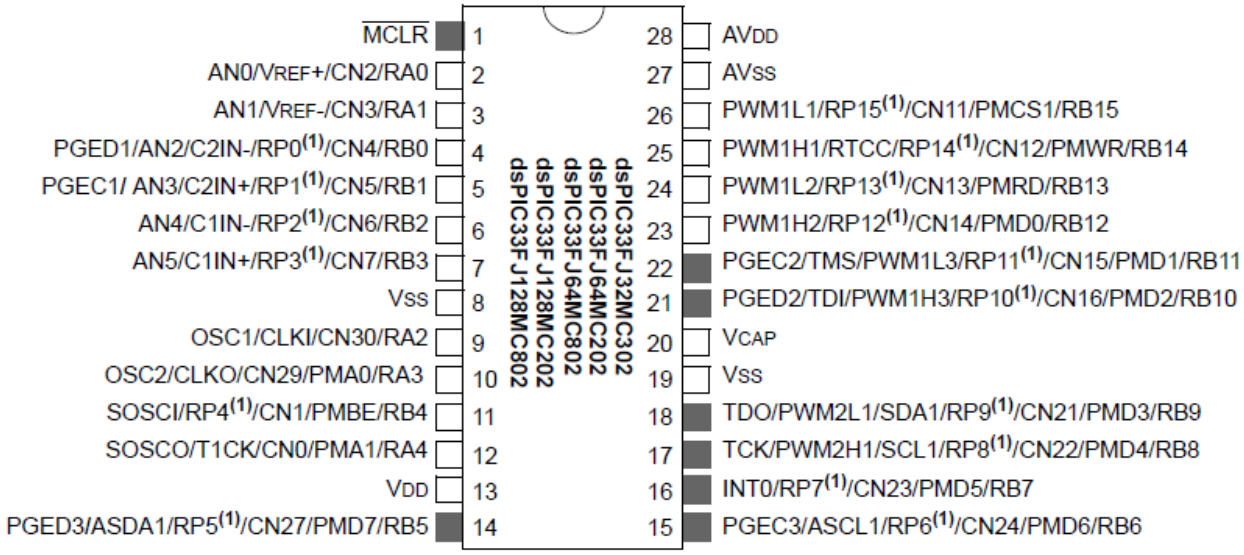


Figure2.19 : brochage du dsPIC33FJ128MC802

Le dsPIC33FJ128MC802 possède aussi :

- ✓ Un bus de données 16 bits
- ✓ Un bus d'instructions 24 bits

8.2.Caractéristiques du dsPIC33F :

Le dsPIC qu'on a utilisé possède les caractéristiques suivantes:

- ✓ Mémoire de type flash
- ✓ Contrôle de moteur par PWM (avec 8 canaux)

- ✓ Conversion analogique numérique 10 bits (ADC),
- ✓ Acquisition de signaux numériques (Input Capture avec 4 canaux)
- ✓ Communications CAN, i2c, UART, SPI.

8.3. Organisation de la mémoire :

La mémoire du dsPIC33F est répartie comme suit :

- ✓ 128kB de mémoire flash (programme)
- ✓ 2048 octets de mémoire données (RAM)

Le dsPIC33F dispose de deux espaces de données, qui peuvent être accédé soit:

➔Séparément, le cas de certaines instructions de traitement numérique de signal (DSP)

➔Simultanément comme une seule plage d'adressage linéaire de 64Kb le cas des instructions MCU du microcontrôleur

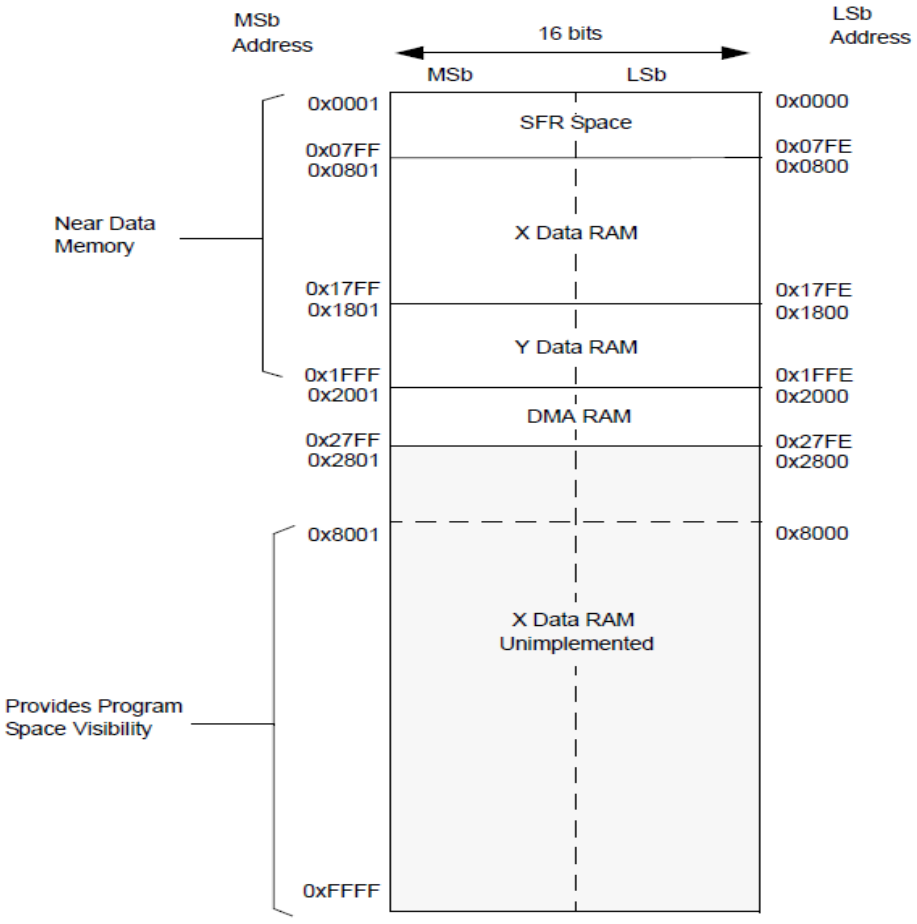


Figure 2.20 : l'organisation de la carte de mémoire de données

Comme le montre la figure 2.20, la RAM commence en 0x0801 et fini en 0x2000 suivant deux espaces mémoire de données X et Y.

- Pour l'écriture de données, les espaces de données X et Y sont accessible toujours comme un espace unique de données linéaires.
- Pour la lecture de données, les espaces mémoire X et Y peuvent être accédées indépendamment, ou comme un seul espace unique et linéaire

8.3.1. L'accès direct à la mémoire (DMA) :

Le contrôleur de l'espace mémoire de donnée du dsPIC33F contient aussi une DMA, c'est un sous système important dans la famille des DSC et DSP, ce sous-système facilite le transfert de données entre le processeur et son périphérique sans l'assistance du CPU.

Le contrôleur DMA dispose de huit canaux indépendants. Chaque canal peut être configuré pour les transferts vers ou depuis les périphériques sélectionnés. Les périphériques pris en charge par le contrôleur DMA comprennent:

- ✓ L'ECAN
- ✓ Interface de convertisseur de données (DCI)
- ✓ Convertisseurs 10 bits/12 bits analogique/numérique
- ✓ Interface périphérique série
- ✓ L'UART
- ✓ Input Capture
- ✓ Output Capture

De plus, les transferts DMA peuvent être déclenchés par des timers (compteurs matériels) ainsi que des interruptions externes. Chaque canal DMA est unidirectionnel. Deux canaux DMA doivent être alloués pour lire et écrire à un périphérique interne (ADC, bus CAN,…).

8.4.L'horloge du dsPIC : [14]

Pour le dsPIC33F il existe 3 types d'horloges :

- ✓ Quartz
- ✓ Oscillateur

- ✓ Horloge interne

On ne va détailler que le quartz puisqu'il va être utilisé dans la programmation du robot (voir chapitre 3)

8.4.1. Utilisation du quartz : [14]

Le quartz est un cristal passif, utilisé par le microcontrôleur pour générer un signal horloge, il est relié aux pins $OSC1$ et $OSC2$ du dsPIC, en fonction de leur fréquence d'oscillation, on retrouve deux types de cristaux:

- ❖ Type XT : quartz lent, entre 3MHz à10MHz maximum
- ❖ Type HS : quartz rapide, entre 10MHz à 40MHz maximum

Les pins $OSC1$ et $OSC2$ nécessitent l'utilisation de 2 condensateurs de 16 à 22 pF pour permettre au quartz d'osciller convenablement.

8.5.Communication : [14]

Plusieurs moyens de communication dans le dsPIC33F sont gérés par le DSC, après une configuration adéquate :

- ✓ I²C: c'est un module de communication série synchrone bidirectionnel, qui offre un support matériel complet des modes multi-maitre et esclave avec une interface 16 bits, sa vitesse est lente (400 kHz maximum)
- ✓ SPI : Interface de communication série synchrone (avec signal Clock) rapide (jusqu'à 10 MHz) mais ne peut être utilisée que sur de faibles distance (une dizaine de cm).
- ✓ UART : définit dans le chapitre suivant
- ✓ CAN ou ECAN (pour Enhanced) : Interface série. Cette interface / protocole a été conçu pour permettre les communications au sein des environnements bruyants, notamment en industrie automobile.

9. La carte de commande :

La carte de commande (Annexe 8) été réalisée sous Proteus le schéma été réalisé avec ISIS, et le circuit imprimé à l'aide de ARES.

9.1.Critères de choix des composants :

Nous avons choisi le dsPIC33F à cause de sa puissance, sa disponibilité et des périphériques internes dont il dispose pour un contrôle adéquat du robot.

Le hacheur L298 est choisi parce que c'est un double pont qui peut aller jusqu'à 4A avec une tension maximale de 46V

Un régulateur de tension pour le dsPIC33F est utilisé pour fournir du 3.3 V au dsPIC à partir de la tension de la batterie qui fournit une tension de 12V.

Le MCP2200 qui sera utilisé pour communiquer par liaison série UART via le port USB au PC et au smartphone Android.

Le dsPIC se met en mode reset lors de démarrage des moteurs, l'utilisation des capacités de filtrage sert à régler ce problème.

10.Conclusion :

Dans ce chapitre nous avons présenté la partie matérielle de notre robot. Cette partie matérielle présentée en deux étapes :

La première consiste à construire une structure mécanique qui est composée d'une plateforme circulaire, des roues et de deux moteurs à courant continu.

Pour la deuxième partie nous avons détaillé les différents composants électroniques utilisés dans la conception de la carte de commande de notre robot.

Chapitre 3 :

La programmation et la commande de Mahfoud III

Introduction :

On s'intéresse dans ce chapitre à la programmation du robot Mahfoud III, on détaillera dans cette partie le programme utilisé pour commander le robot, les différents modules du dsPIC33F qui interviennent pour réaliser cette tâche et la liaison choisie entre le robot et le smartphone Android.

1. Présentation de l'environnement MPLAB :

Le programme de commande de la carte du robot est écrit en langage C et réalisé sous l'environnement de développement MPLAB comme le montre la figure suivante :

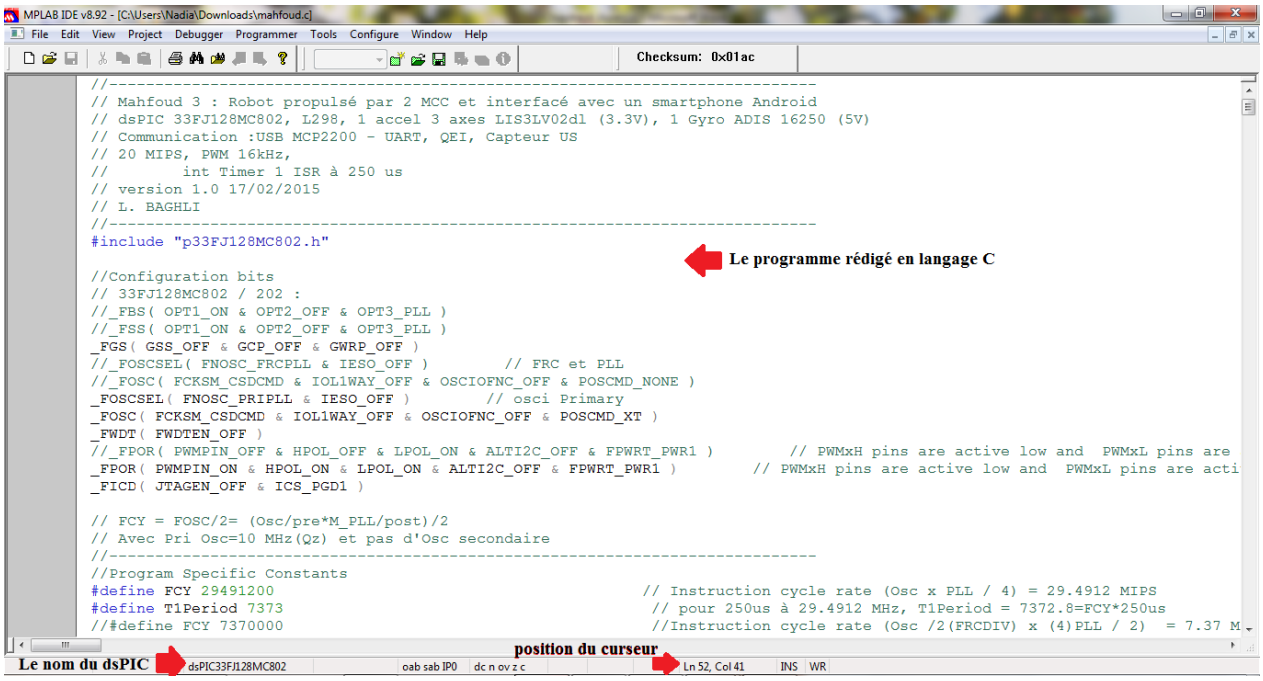


Figure 3.1 : Mahfoud II sous MPLAB

Le lien entre nous et le microcontrôleur est MPLAB, l'application qui permet de taper le code, le déboguer et le charger dans le microcontrôleur, MPLAB offre aussi un outil de simulation qui permet de déboguer le code comme s'il était dans le microcontrôleur, parmi les fonctionnalités de MPLAB on peut citer:

- Une interface graphique unique à tous les outils de débogage
  - ➔ Simulateur
  - ➔ Programmeur (vendu séparément)
  - ➔ In-Circuit Emulateur (vendu séparément)

➔ Débogueur en circuit (vendu séparément)

- Un éditeur complet avec le contexte d'un code couleur
- Un gestionnaire de projet multiple
- Fenêtres de données personnalisables avec modification directe des contenus
- Haut niveau code source débogage
- Passez la souris sur contrôle variable
- Glissez et déposez les variables à partir de la source de regarder les fenêtres
- Un aide en ligne

Le MPLAB IDE permet de:

- Modifier les fichiers source (C ou assemblage)
- Une touche compiler ou assembler, et télécharger des outils d'émulation et des émulateurs (met à jour automatiquement toutes les informations du projet)
- débogage en utilisant:
  ➔ Les fichiers source (C ou d'assemblage)
  ➔ C mixte et l'assemblage
  ➔ Code de la machine

2. Les programmeurs de périphériques :

Le programmateur PICkit 2 (figure 3.2) est l'outil essentiel qui nous permettra de mettre notre code exécutable dans le microcontrôleur. Il est relié à l'ordinateur par le port USB, et au microcontrôleur par un support ICSP (In-Circuit Serial Programming). La connexion au circuit est effectuée à l'aide de six broches. La position de la broche numéro 1 est identifiée par le triangle blanc sur le PICkit. Cette broche doit être reliée à la broche numéro 1 du dsPIC33F.

Legende:

1 – LED de status	3 – Dragonne de connection	5 – Indicateur de la Pin 1
2 – Bouton poussoir	4 – Port de connexion USB	6 – Connecteur de programme

Figure 3.2 : Interface de programmation PICkit2

Le PICkit2 ne sert pas uniquement à transférer du code exécutable. Il sert également à déboguer le code présent dans le microcontrôleur. Le débogage, tout comme la programmation du microcontrôleur, s'effectue très aisément avec l'environnement de développement MPLAB. De plus, le PICkit2 permet d'alimenter le dsPIC33F dans le cas où ce dernier ne serait pas encore relié à une alimentation extérieure 3.3V.

Attention cependant, le PICkit 2 ne permet pas de fournir beaucoup de courant à la carte. C'est d'ailleurs ce qui nous a causé des problèmes dans le débogage de la connexion USB avec le smartphone.

3. La programmation de la carte :

3.1. Configurations des bits :

LedsPIC33Finclut plusieurs fonctionnalités destinées à maximiser la flexibilité et la fiabilité de l'application, et de réduire les coûts grâce à l'élimination de composants externes. Ceux-ci sont:

- ✓ Configuration flexible
- ✓ Le Watchdog Timer(WDT)
- ✓ Protection du code
- ✓ Interface JTAG
- ✓ Programmation série en circuit
- ✓ Emulation en circuit

Avant de commencer la programmation de la carte, on a configuré ces bits en haut du programme et avant toute initialisation.

```
_FGS( GSS_OFF & GCP_OFF & GWRP_OFF )
```

Par cette ligne on veut configurer les segments généraux comme la haute protection, verrouiller la mémoire d'utilisateur à écriture seulement,

```
_FOSCSEL( FNOSC_PRIPLL & IESO_OFF ) // osci Primary
_FOSC( FCKSM_CSDCMD & IOL1WAY_OFF & OSCIOFNC_OFF & POSCMD_XT )
```

La première phrase c'est pour la sélection PRIPLL interne lors de la réinitialisation, et la deuxième c'est pour désactiver la commutation de l'horloge, et pour mettre l'oscillateur primaire en mode XT (c'est-à-dire oscillateur à quartz).

```
_FWDT( FWDTEN_OFF )
```

Le Watchdog Timer est activée / désactivée par programmation

```
_FPOR( PWMPIN_ON & HPOL_ON & LPOL_ON & ALTI2C_OFF & FPWRT_PWR1 )
_FICD( JTAGEN_OFF & ICS_PGD1 )
```

Les pins de PWMxH sont actifs à l'état bas et les pins PWMxL sont actifs à l'état haut, et ils sont contrôlés à la réinitialisation par IOPort,

D'autre part l'interface JTAG (méthode de test des cartes électroniques modernes après assemblage) est désactivée et la programmation in-situ est utilisée via les pins PGD1/PGC1

3.2.Définition des constantes utilisées dans le programme :

Les constantes dans le programme doivent être déclarées avant leur utilisation, leur valeurs sont inchangeable lors de l'exécution du programme. Ce ne sont pas des constantes du langage C mais des redéfinitions que le compilateur C va utiliser. Ceci se fait par la directive `#define`, ce qui permet de remplacer toutes les occurrences du mot qui le suit par la valeur donnée.

```
//Program Specific Constants
#define FCY 29491200                    // Instruction cycle rate (Osc x PLL / 4) = 29.4912 MIPS
#define T1Period 7373                   // pour 250us à 29.4912 MHz, T1Period = 7372.8=FCY*250us
//#define FCY 7370000                   //Instruction cycle rate (Osc /2(FRCDIV) x (4)PLL / 2)  = 7.37
//#define T1Period 1842                 // pour 250us à 7.37 MHz, T1Period = 1842.5=FCY*250us

#define FPWM 16000                      // PWM freq
#define HalfDUTY    921 // set the pwm half duty = PTPER = 921UL=FCY/FPWM/2-1;
#define FullDUTY HalfDUTY*2         //  car PDCx a une double resolution % PTPER
#define MILLISEC FCY/29491          // 1 mSec delay constant
#define Delay12us 62            // 15 us in NOP waiting loop NCOUNT
```

Dans un premier temps on a défini les grandeurs :

- ✓ Taux de cycle d'instruction est à 29 MPIS
- ✓ La période du timer T1 :

Le timer s'exécute chaque 250 μs donc pour calculer la période du timer

Pour calculer la période du Timer 1 on applique la formule suivante tirée de schéma de la figure 3.3

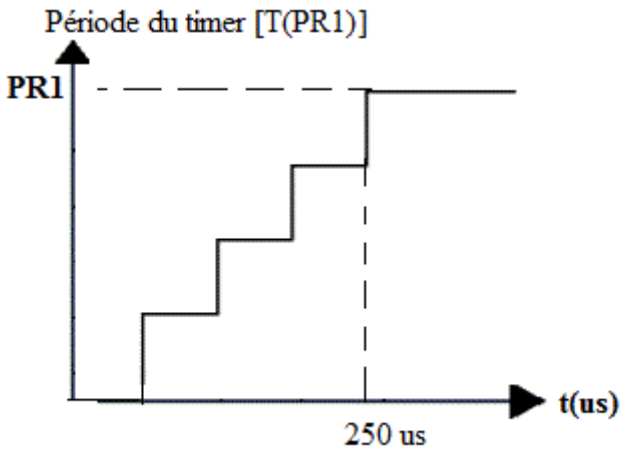


Figure 3.3 PR1 en fonction du temps

Ainsi la formule tirée pour calculer la période du timer est la suivante :

$$T(PR_1) = \frac{250\ \mu s}{1/FCY} = 7372.8$$

- ✓ La fréquence de la PWM est à 16kHz
- ✓ Le demi-cycle de la PWM est 921
- ✓ Puisque PDCx a une double résolution, dans ce cas le cycle complet de la PWM=$HalfDuty * 2$

4. Le timer :

La fonction la plus utile du microcontrôleur est le timer, c'est grâce à ce dernier qu'on peut réaliser une tache avec une fréquence particulière.

Son principe c'est d'incrémenter un registre régulièrement et proportionnellement a la fréquence Fcy, jusqu'à atteindre une certaine valeur, une fois atteinte l'interruption du timer se déclenche permettant de réaliser l'action voulue.

Le dsPIC qu'on a utilisé dispose de 5 timers, mais on n'a utilisé que 3 timers dans la programmation de notre robot,

- ✓ Le timer 2 est utilisé pour la génération de l'input capture,
- ✓ Le timer3 est utilisé pour la lecture du capteur ultrasonique réglé à $10\mu s$ comme le montre le code suivant :

```
//  Read_US lis les capteurs UltraSons
//-----------------------------------------------------------------------------
void          Read_US()
{
        TMR3= 0;
        TRIG_US = 1;                                    // start the US
        T3CONbits.TON = 1;          // enable Timer 1 and start the count
}
//-----------------------------------------------------------------------------
// Timer3 ISR compte 10 us
//-----------------------------------------------------------------------------
void __attribute__((interrupt, auto_psv)) _T3Interrupt( void )
{
        _T1IF = 0;
        T3CONbits.TON = 0;          // Stop Timer 1 and start the count
        TRIG_US = 0;                                    // stop the US starting signa
}
```

5. Input Capture :

Les deux fonctions qui réagissent à un changement d'état d'une entrée du microcontrôleur en générant une interruption sont :

- Input capture : via un couplage au timer, cette fonction permet de mesurer précisément le temps écoulé entre deux changements d'état. (voir figure 3.4)
- Interruptions externes : utilisés principalement pour le comptage d'occurrence de changement d'état.

Ce périphérique est utilisé pour le capteur ultrason.

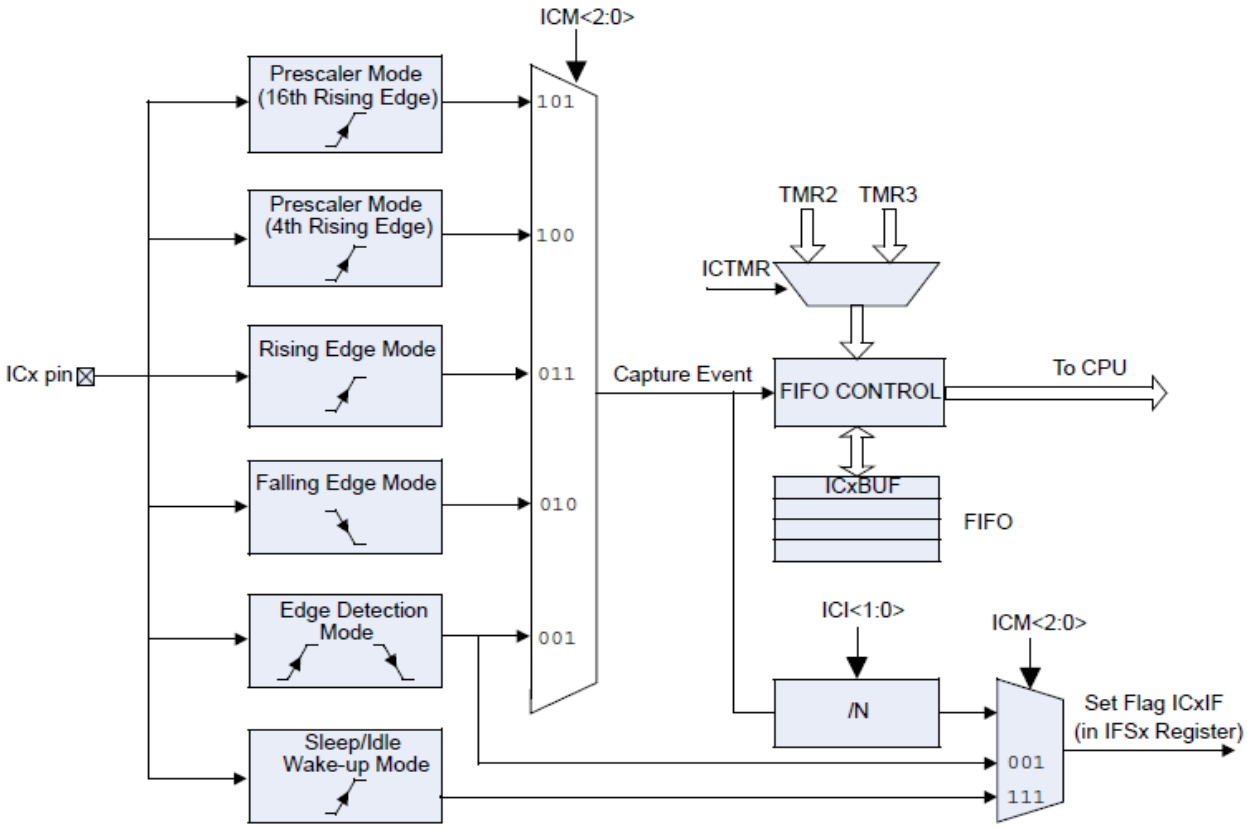


Figure 3.4 : diagramme de l'input capture

On a configuré l'input capture dans notre programme comme suit :

```
void InitInputCapture()
{
// Initialize Capture Module
        IC1CONbits.ICM = 0b000; // Disable Input Capture 1 module
        IC1CONbits.ICTMR = 1;                 // Select Timer2 as the IC1 Time base
        IC1CONbits.ICI = 0b01;          // Interrupt on every second capture event
        IC1CONbits.ICM = 0b001;          // Generate capture event on 001 Edge Capture moc
        // Enable Capture Interrupt And Timer2
        _IC1IP = 1; // Setup IC1 interrupt priority level (higher than Timer1)
        _IC1IF = 0; // Clear IC1 Interrupt Status Flag
        _IC1IE = 1; // Enable IC1 interrupt
}
```

Le registre *ICxCONbits* permet le paramétrage du module d'input capture, dans notre cas on a pris les paramètres suivants :

IC1CONbits.ICM =0b000 ; avant de changer le mode capture on doit en premier le désactiver

IC1CONbits. ICTMR =1 ; cela signifie que lorsque l'interruption survient, le contenue du timer 2 est recopié dans le registre associé à l'input capture.

IC1CONbits. ICI =0b01 ; décale la génération de l'interruption de 1 à 4 survenue de l'évènement scruté. Dans notre cas l'ICI est mis à 2, l'interruption de l'IC sera générée au deuxième événement non à chaque fois, comme le montre la figure 3.5

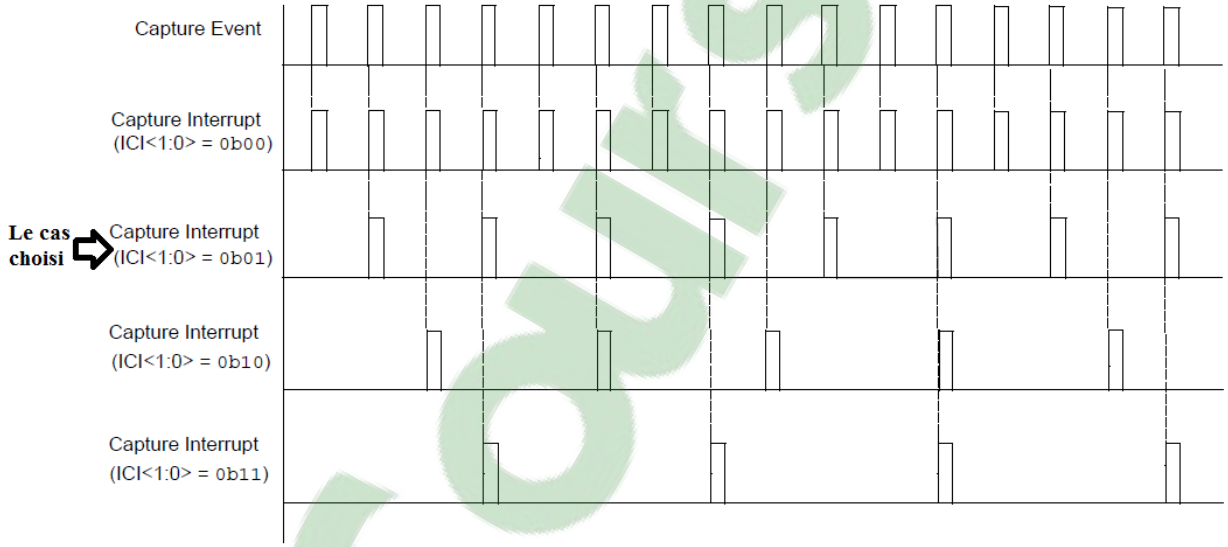


Figure 3.5 ; génération d'interruption de l'input capture

IC1CONbits.ICM =0b001 ; précise l'évènement scruté. On a pris une génération d'interruption à chaque front montant et descendant comme le montre la figure suivante :

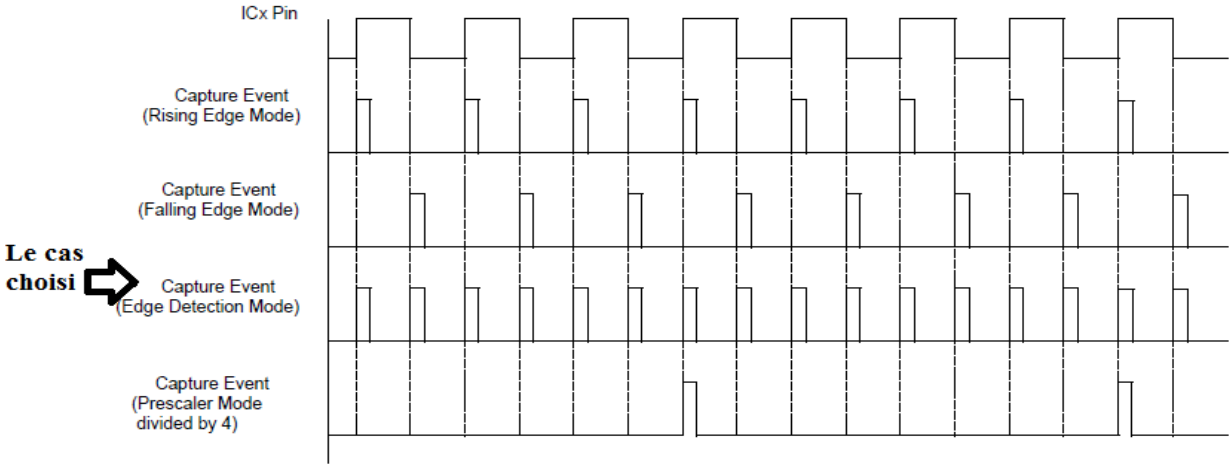


Figure3.6 : génération de l'événement d'input capture

Mais puisque *ICM=001 ;* dans ce cas le décalage de L'<ICI> ne fonctionne pas, et ses bits seront ignoré

Comme pour un timer, après avoir configuré les bits, on active les interruptions de l'IC par :

- _IC1IP = 1; La priorité de l'input capture est supérieure a celle du timer
- _IC1IF = 1; Après s'être assuré au préalable que l'interruption n'est pas en cours, sans quoi le code ne sera jamais appelé
- _IC1IE = 1; On lance l'IC

6. La PWM :

Les signaux qui attaquent les 2 hacheurs du L298 utilisés pour piloter les 2 MCC sont générés par le module PWM du dsPIC33F, le code de configuration pour la génération des signaux PWM est le suivant :

```
void InitMCPWM()
{
        PTPER = HalfDUTY;  // set the pwm period register, ne pas oublier la double précisi
        ENABLE_FIRING ; //Enable PWMs tt le tps sauf RE0, RE1, RE2 qui restent en Input pou
        PWM2CON1=0;                     // RB9, RB8 sont I/O et non pas PWM2L1 et PWM2H1
        PDC1=HalfDUTY; PDC2=HalfDUTY; PDC3=HalfDUTY;// init sans rien, apres une regul ça c
        EnableL298=0;                                                   // disable L6234
        OVDCON = 0xFFFF;        // Cmde MLI, no effect of OVDCON
// pas d'ADC ici, Interruption PWM
//        SEVTCMP = PTPER;               // set ADC to trigeer at ...
        PWMCON2 = 0x0000;               // 1 PWM values
//        _PWM1IF=0;
//        _PWM1IE=1;
        PTCON = 0x8002;                         // start PWM symetrique
}
```

La PWM est à 16Khz, on n'a pas fait une interruption pour la PWM, la figure suivante montre le signal PWM (voie 2)

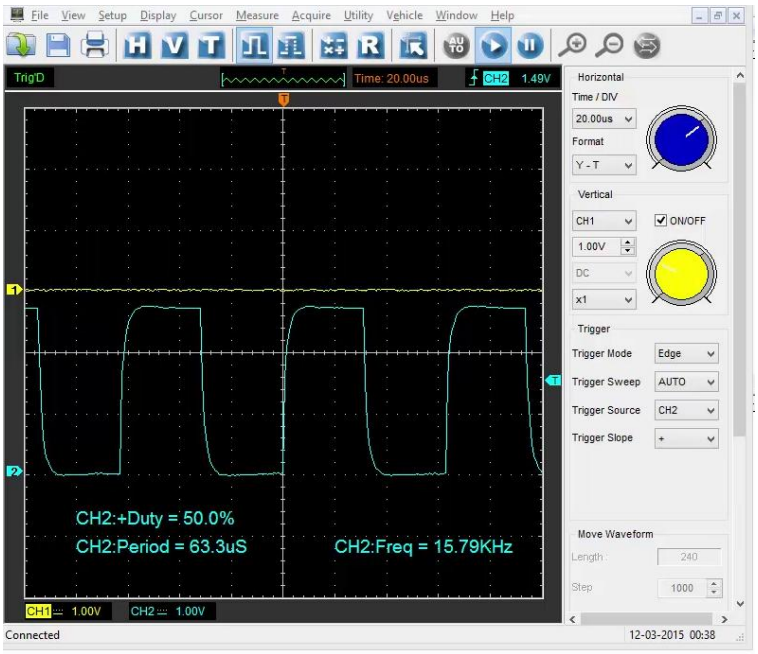


Figure3.7 : variation de la PWM

7. Connexion à l'aide de l'UART :

La communication entre le smartphone doté d'un Android OS et le dsPIC33F, se fait par liaison série,

On peut transmettre les données numériques sous formes de bits par deux manières :

- ✓ Mode série : dans ce mode les bits de données arrivent sur la ligne à la suite les uns des autres comme le montre la figure suivante :

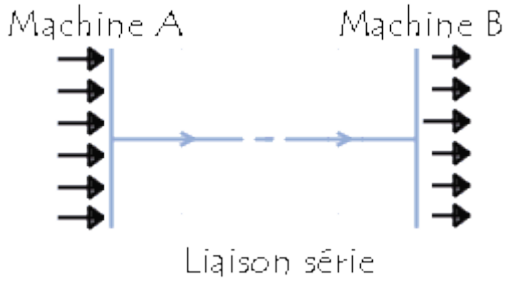


Figure 3.8 : mode liaison série

- ✓ Mode parallèle : dans ce mode la totalité des bits est transmise à la fois comme dans la figure suivante :

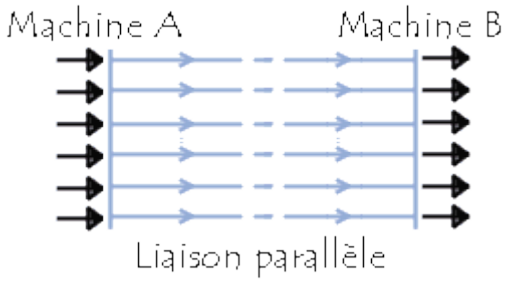


Figure 3.9 : mode liaison parallèle

Pour nous on s'intéresse au mode série, puisqu'elle n'utilise qu'un seul fil, qui lui-même comporte deux catégories :

- Mode asynchrone : une communication entre l'émetteur et le récepteur où les deux se mettent d'accord sur le calendrier du transfert, le récepteur reçoit en continu les données au rythme de l'metteur, ce mode nécessite l'insertion d'éléments de synchronisation, des données de plus sont insérées lors de la transmission afin d'éviter les erreurs.
- Mode asynchrone : chaque caractère est émis de façon irrégulière dans le temps, les transferts asynchrones sont généralement plus rapides que les transferts synchrones. C'est parce que le récepteur et l'émetteur ne prennent pas le temps avant le transfert de coordonner leurs efforts, dans ce mode de transfert chaque caractère est précédé d'une information indiquant le début de la transmission du caractère l'information de début d'émission est appelée bit START et terminé par l'envoi d'une information de fin de transmission appelée bit STOP, il peut éventuellement y avoir plusieurs bits STOP.

Le dsPIC dispose d'un module émetteur-récepteur asynchrone universel qu'on a utilisé pour le transfert de donnée entre, l'UART du dsPIC33F, est un canal de communication asynchrone full duplex (les données circulent de façon bidirectionnelle et simultanément)

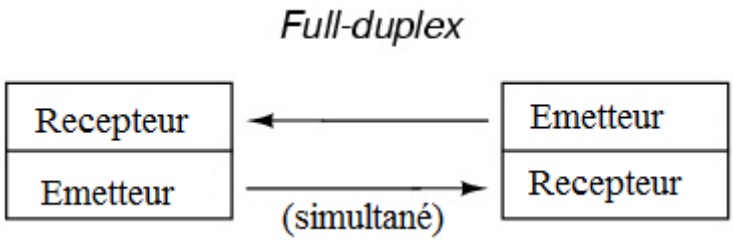


Figure 3.10 : Communication full-duplex

Un schéma de principe simplifié de l'UART est représenté sur la figure 3.10. Le module UART comprend les éléments matériels suivants:

- ✓ générateur de vitesse de transmission (exprimé en Baud)
- ✓ Emetteur asynchrone
- ✓ Récepteur asynchrone

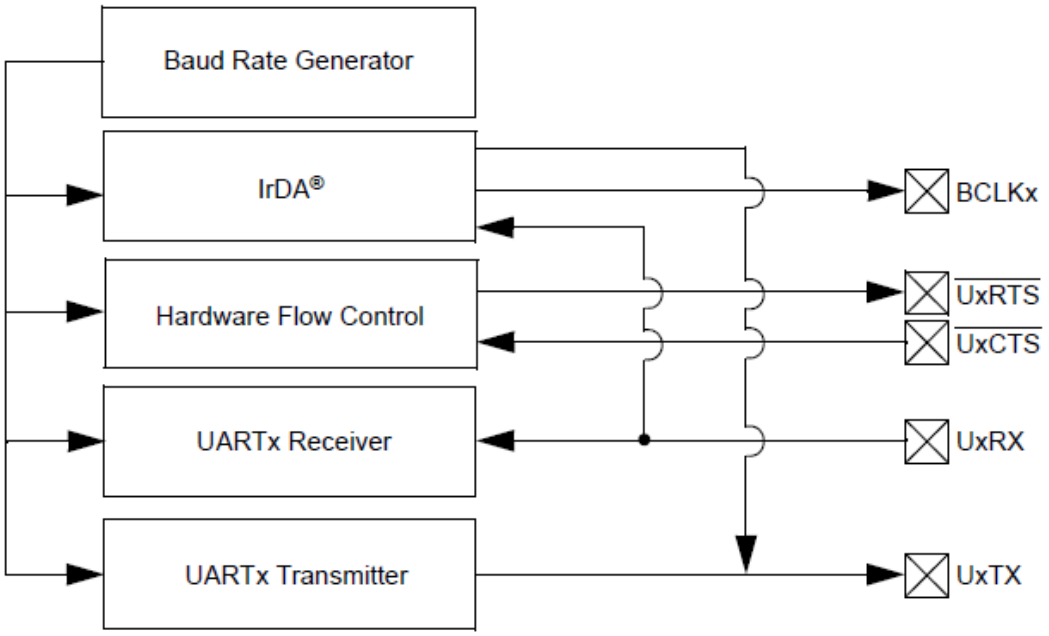


Figure 3.11 : Diagramme simplifié de l'UART

```
void InitUART()
{
//        U1MODE = 0x8000;      // enable UART1, 8 bits, 1 stop bit, no parity, RX normal
        U1MODE = 0x8200;    // enable UART1, 8 bits, 1 stop bit, no parity, RX normal, RTS/CT
        U1STA = 0x0000;
        U1BRG = ((FCY/16)/BAUD) - 1;         // set baud rate
        RXPtr = &InData[0];                 // point to first char in receive buffer
        Flags.CheckRX = 0;                          // clear rx and tx flags
        IFS0bits.U1RXIF = 0;         // clear interrupt flag
        IEC0bits.U1RXIE = 1;                         // enable RX interrupt
        Flags.SendTX = 0;
        Flags.SendData = 0;                 // clear flag
        SeqComm=SeqCommMax;
        U1STAbits.UTXEN = 1;           // Initiate transmission
}
```

L'UART dans notre programme est configuré a transmettre 8 bits avec 1 seul bits de stop, les pins TX, RX, CTS et RTS.

Les pins CTS (Clear To Send) et RTS (Ready To Send) permettent de contrôler le flux de manière matériel et évitent la perte de données.

Pour envoyer les données au smartphone une conversion de l'hexadécimal vers l'ASCII est nécessaire et elle est donnée par la routine SendData du programme :

```
void SendData()
{
// prépare OutData[] pour sortir en ascii sur UART en Hex
    ConvHexa( PWM1, OffsetPWM1, OutData);                // PWM1
    ConvHexa( PWM2, OffsetPWM2, OutData);                // PWM1
    ConvHexa( Vit, OffsetVit, OutData);                  // Vit
    ConvHexa( tcnt, OffsetPosRef, OutData);      // PosRef
//  ConvHexa( PosRef, OffsetPosRef, OutData);        // PosRef
    ConvHexa( Pos, OffsetPos, OutData);                  // Position de l'obstacl
```

Des tests ont été effectués à l'aide de l'interface putty sur PC via un port COM sur USB.

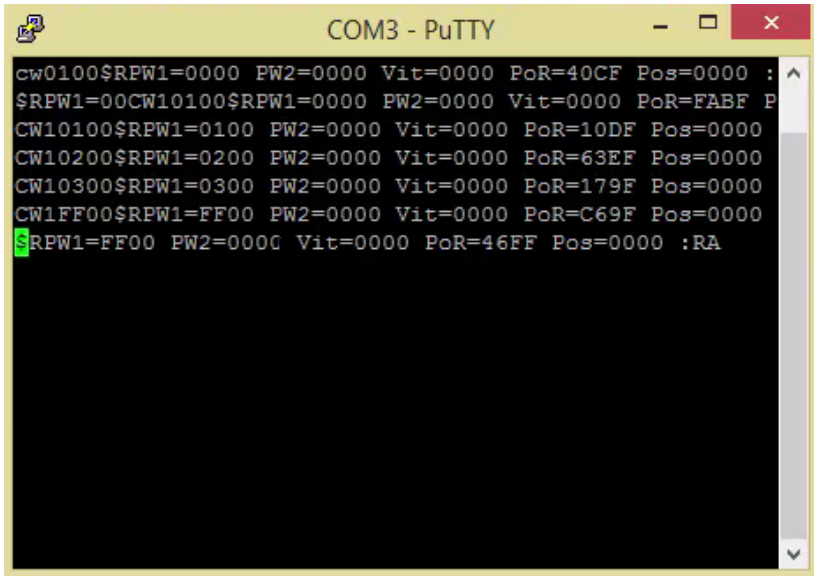


Figure3.12 : exemple d'un test fait sur putty

8. Lecture des codeurs incrémentaux :

L'interface du codeur incrémental du dsPIC33F est utilisée pour la détermination de la vitesse de rotation et la position du rotor des moteurs du robot. Le module QEI est relié aux phases A, B et à l'index du codeur incrémental utilisé dans le robot et décrit précédemment (le MEnc 13), ce module enregistre les impulsions de comptage accumulées dans une base de temps dédié de 16bits.

8.1.Les signaux de sortie du codeur incrémental :

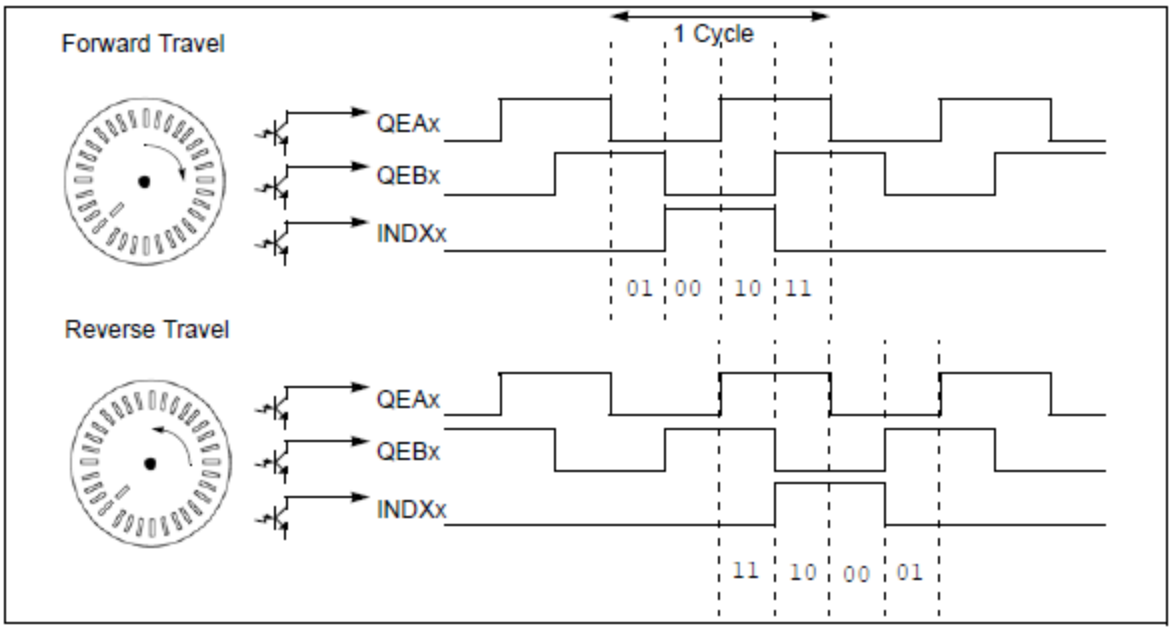


Figure 3.13 : Les signaux de sortie du module QEI

Le sens de rotation du moteur est donné par les signaux de phase A et B :

- ✓ Si la phase A (QEA) est en avance par rapport a la phase B(QEB), le sens de rotation est considéré comme positive ou vers l'avant.
- ✓ Si la phase A est en retard par rapport à la phase B, le sens de rotation st considéré comme négative ou au sens inverse

Les codeurs incrémentaux sont attachés aux pins du registre RP comme le montre le code suivant :

```
_QEA1R = 7;      // RPINR14  RP7 QEA_Left
_QEB1R = 8;      // RPINR14  RP8 QEB_Left
_QEA2R = 10;         // RPINR14  Rp10 QEA_Right
_QEB2R = 9;      // RPINR14  RP3 QEB_Right
```

L'exploitation logicielle de cette partie n'a pas été finalisée dans ce projet, mais la partie matérielle est présente sur la carte.

9. Lecture des capteurs infrarouges :

L'ADC convertit le signal analogique renvoyé par le capteur infrarouge en données numériques, le code utilisé pour cela est le suivant :

```
void InitADC10()
{
AD1CON1 = 0x0044; // ASAM bit = 1 implies sampling ..
// GP Timer3 compare ends sampling and starts conversion
AD1CON2 = 0;
AD1CHS0= 0x0005; // Connect RB7/AN7 as CH0 input ..
// in this example RB7/AN7 is the input
AD1CSSL = 0;
AD1CON3 = 0x0002; // Sample time manual, Tad = internal 2 Tcy

_AD1IF=0;
_AD1IE=1;
T3CONbits.TON = 1; // enable Timer 1 and start the count
}
```

Ensuite, on utilise l'interruption de l'ADC pour lire le capteur infrarouge et piloter les rapports cycliques des 2 MCC. Le code suivant permet un évitement d'obstacle :

```
void __attribute__((interrupt, auto_psv)) _ADC1Interrupt ()
{
_AD1IF = 0;
Pos=ADC1BUF0<<4; // 10 bits * 4 pour avoir 4096, Q12 : 4.12 pu
if(Pos>=0X0260)
     {
     PWM1     =0XF200;
     PWM2     =0X0200;
     }
else
     {
     PWM1     =0X0200;
     PWM2     =0X0200;
     }
}
```

10.Conclusion

Dans ce chapitre, on a présenté la programmation de la carte de commande du robot, contrairement à la version précédente du robot, le robot Mahfoud III comporte maintenant une seule carte de commande, au lieu de 3, cette carte réaliseen même temps le contrôle, la commande du robot et l'interfaçage, ce qui montre la puissance du dsPIC33F. Ceci nous a permis de bien exploiter le robot avec un minimum de composants, on a réussi à le commander en lui associant des capteurs, des codeurs et des interfaces de communication.

Chapitre 4 :

L’application mobile de Mahfoud III

Introduction :

Une façon pour rendre le robot intelligent était de lui associer un smartphone avec lequel il va communiquer par liaison série USB, cette connexion entre le robot et le smartphone doit assurer l'échange d'informations entre les deux, elle va aussi permettre au robot d'interagir avec le monde extérieur par le biais de l'application mobile en envoyant ses données internes à un site web, la réalisation de cette partie est détaillée dans ce chapitre.

1. Notions sur Android :

1.1. Android :

Android est un système d'exploitation mobile qui est basé sur une version modifiée de Linux. Il a été initialement développé par une start-up du même nom, Android, Inc. En 2005, dans le cadre de sa stratégie pour entrer dans l'espace mobile, Google a racheté Android et a repris son travail de développement (ainsi que son équipe de développement).

Ce Système d'exploitation ne permet pas seulement de transmettre des appels et d'envoyer des sms, mais il permet aussi d'interagir avec l'environnement notamment l'emplacement géographique.

1.2. Les avantages de l'Anroid :

Parmi les avantages d'Android on trouve :

- ✓ Open source : c'est un système dont les sources sont disponibles gratuitement sur internet pour les télécharger et les modifier selon les besoins.
- ✓ Publication : il est possible de publier son application sur le Play Store (anciennement Android Mark) pour une somme de 25$, cette somme permet de publier autant d'applications à vie.
- ✓ Facilité de développement : avec les API disponibles et mises à disposition le développement d'une application serait facile et ça réduit le travail aussi.
- ✓ Flexibilité : ce système s'adapte non seulement aux Smartphones, mais aussi aux tablettes, téléviseurs, les consoles de jeux, les appareils photos, il existe même des micro-ondes fonctionnant sous Android !

1.3.Les éléments d'une application Android :

- Activities (les Activités) : c'est la composante principale d'une application Android, elle représente une partie de l'application présentant une vue à l'utilisateur, importé par le paquet (android.app.Activity)
- Services (Des services) : à la différence d'une activité, le service ne possède pas de vue, c'est plutôt une activité tache de fond, importé par le paquet (android.app.Service), elle permet l'exécution d'un algorithme sur un temps indéfini qui ne s'arrête que lorsque la tâche est finie ou son exécution est arrêtée, et il peut être lancé à différents moments :
  - ✓ Au démarrage du smartphone
  - ✓ Au lancement de l'application
  - ✓ Par une action particulière sur le smartphone
  - ✓ Lors un événement (par exemple, l'arrivé d'un appel, un sms, etc.…)
- Intents (les intentions) : ça permet d'envoyer un message pour un composant externe sans le nommer explicitement. Il importé par le paquet (android.app.Intent)
- Broadcast receiver (les récepteurs des intentions) : il permet d'écouter ce qui se passe sur le système ou sur l'application et de déclencher une action prédéfinie, ça permet de déclarer la capacité de répondre à des intentions. Il est importé par le paquet (android.app.BroadcastReceiver)
- Content providers (les fournisseurs de contenus) : permet de partager des informations au sein ou entre applications importé par le paquet (android.app.ContentProvider), ça nous permet d'accéder à :
  - ✓ Les contacts stockés dans le téléphone
  - ✓ L'agenda
  - ✓ Les photos de la galerie

1.4.Le cycle de vie d'une application Android : [15]

A chaque étape du cycle correspond l'appel d'une méthode précise, que l'on peut surcharger pour définir le comportement correspondant à cette étape, il existe plusieurs méthodes relatives à l'interface, aux menus et menus contextuels, à l'environnement et ressources, au cycle de vie, on ne va détailler que la dernière méthode dans cette partie.

Le diagramme de la figure 4.1 représente les méthodes appelées lors de l'exécution d'une application Android, à chaque moment de vie de l'application correspond une méthode (tableau 4.1. [15]).

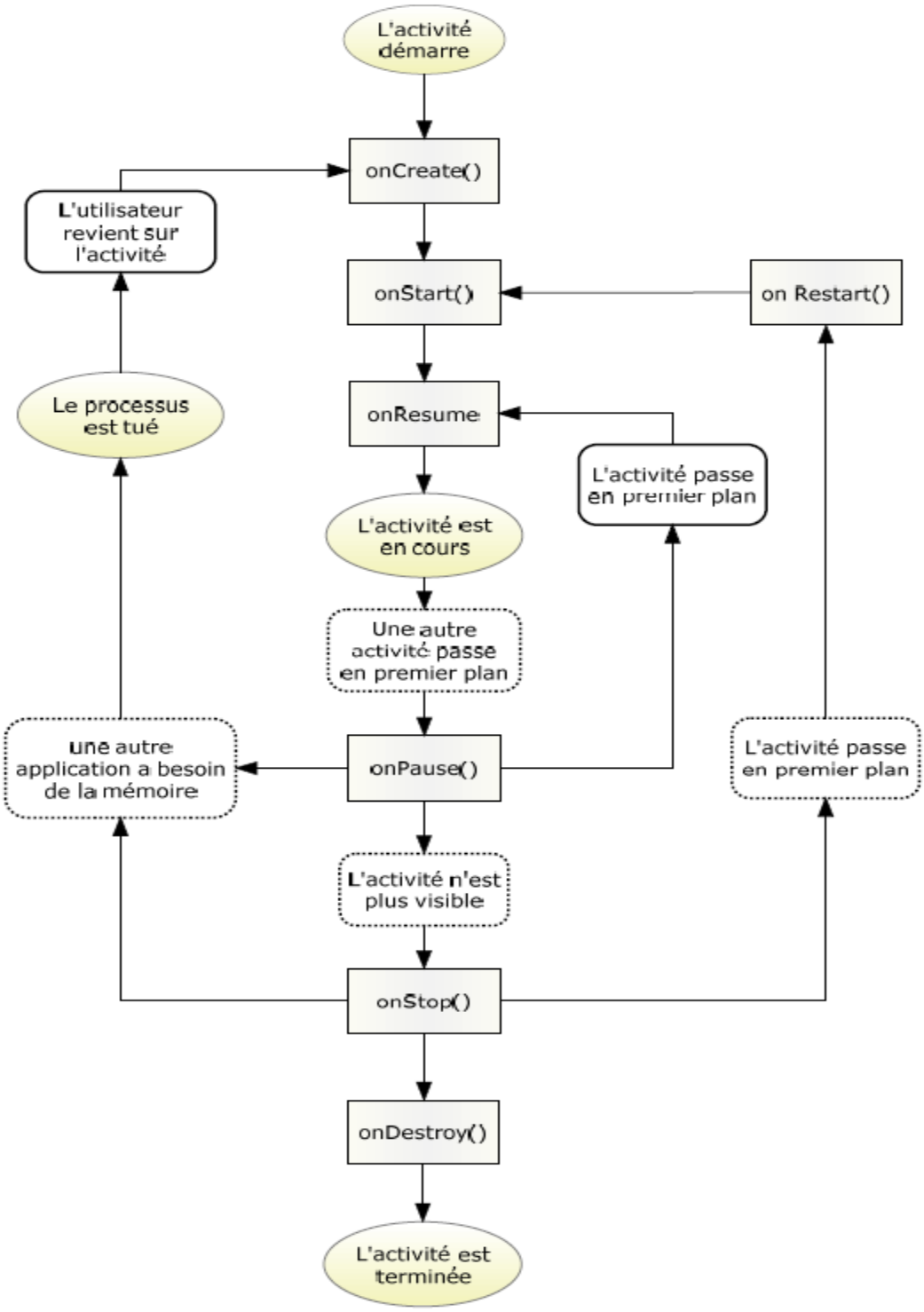


Figure 4.1 : Cycle de vie d'une application Android

- onCreate (Bundle) : appelée à la création de l'activité, sert à initialiser l'activité mais aussi toutes les données nécessaires pour cette dernière. On associe à cette méthode un Bundle en argument, ce dernier contient l'état de sauvegarde enregistré lors du dernier enregistrement de l'activité.
- onStart () : cette méthode est appelée quand l'activité démarre (début du passage au premier plan).Si l'activité ne peut pas aller en premier plan quelque soit la raison, l'activité sera transférée à onStop ( ).
- onResume () : appelée après onStart (), quand cette méthode est appelée l'application se trouve au premier plan et reçoit les interactions d'utilisateur
- onPause () : cette méthode est appelée quand une autre activité passe au premier plan
- onStop () : quand l'activité n'est plus visible cette méthode est appelée
- onRestart () : appelée quand l'activité redémarre
- onFinish () : permet de finir l'activité
- onDestroy () : appelée quand l'application se termine, quand elle est totalement fermée, dans cette méthode, si les données ne sont pas sauvegardées, ils seront perdus.

2. Environnement de développement Android Studio :

L'application mobile de Mahfoud III a été développé sous Android Studio et non pas Eclipse, on va donner un bref aperçu sur cet environnement de développement dans cette section, la figure 4.2 montre le projet de l'application Mahfoud III sous Android Studio.

De plus, Android Studio dispose d'une interface graphique (figure 4.3) qui permet :

- ✓ D'ajouter des widgets dans l'interface de l'application
- ✓ Redimensionner les éléments existant dans l'interface de l'application
- ✓ Changer le layout de l'application
- ✓ Changer le thème de fond de l'application
- ✓ Modifier les propriétés des éléments placés dans l'interface (taille, couleur, alignement…)

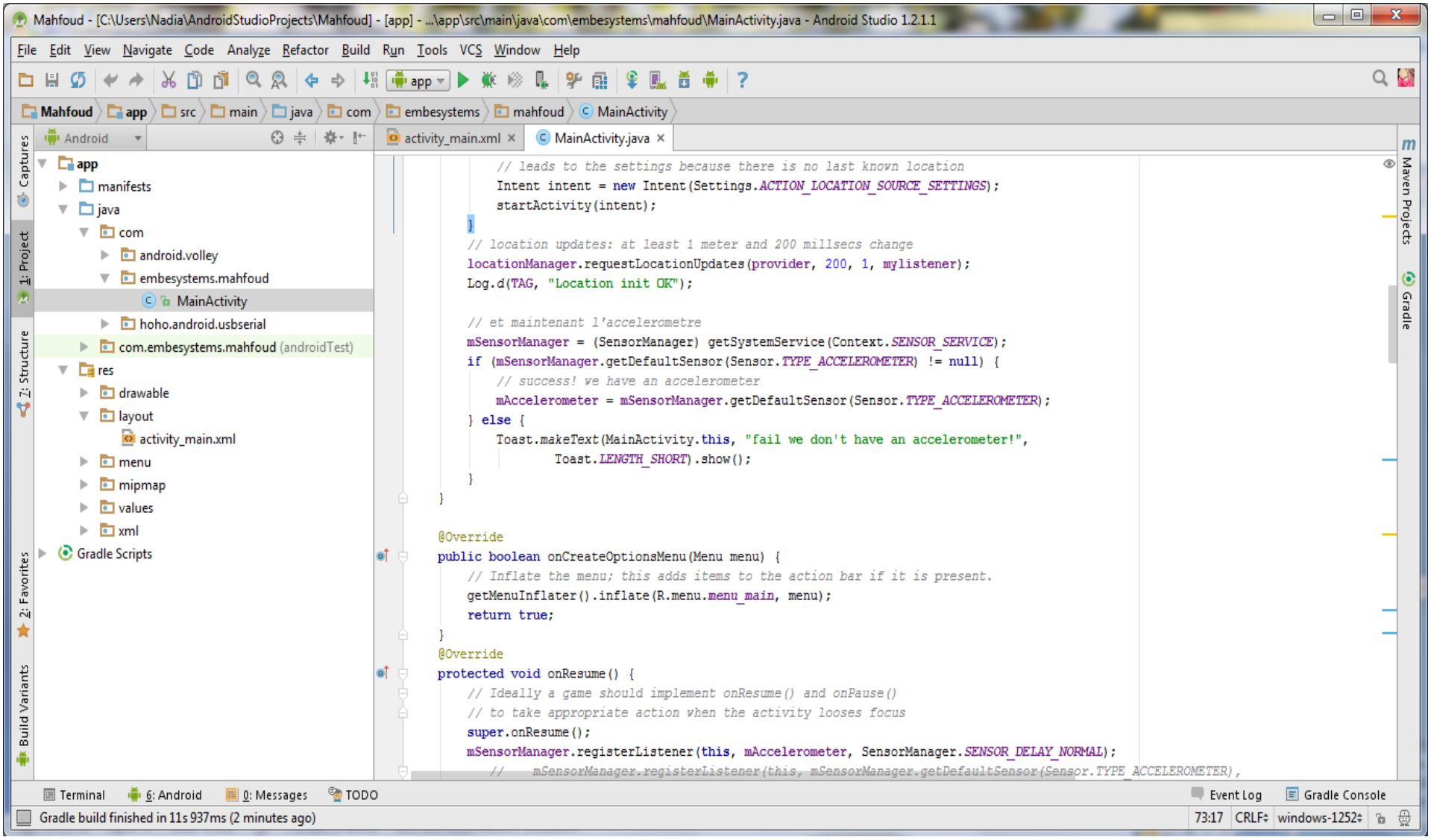


Figure 4.2 : Projet Mahfoud III sous Android Studio

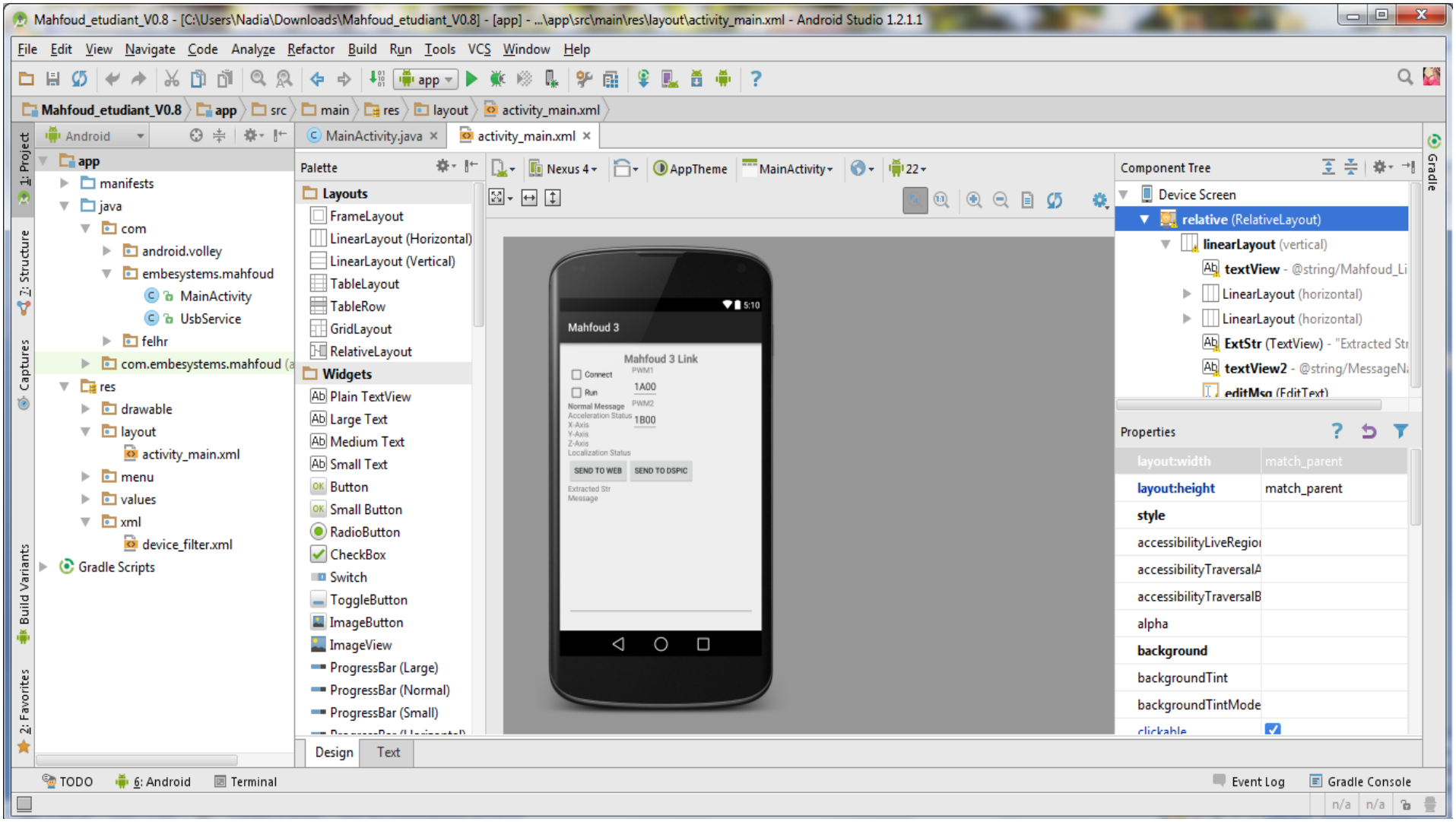


Figure 4.3 : L'interface graphique du projet de Mahfoud III sous Android Studio

2.1.Les panneaux : [16]

L'environnement de développement Android studio est composé d'une vaste gamme de panneaux, des outils et des fonctions afin d'améliorer la productivité le maximum possible en développant une application Android. On détaillera les panneaux les plus utilisés, les fenêtres et les bars d'outils les plus utilisés et avec lesquelles on va interagir ;

1	**Panneau du projet**	**Permet de naviguer dans la hiérarchie du projet et de sélectionner, ouvrir, éditer et faire pleins d'autres actions sur les fichiers**
2	L'éditeur des fichiers	La fenêtre d'édition principale d'Android Studio, c'est là où le code est écrit
3	Panneau Android	Ce panneau présente les émulateurs et les dispositifs physiques connectés au système, ça permet de visualiser la sortie logcat, (pour le débogage) filtrer la sortie & voir les journaux de l'ADB
4	Panneau des messages	C'est au niveau de ce panneau que les messages de l'IDE s'affichent, notamment les erreurs de compilation
5	Le panneau AFaire	Affiche tous les commentaires TODO parsemés à travers le code du projet.
6	Panneau de recherche	Ici vous pouvez examiner les résultats de toute commande « find » que vous exécutez. Les exemples incluent la commande Rechercher Résultats (Edit →Find→Find) et la commande Find Usages (Edit →Find→Find Usages).
7	Panneau Maven	Si le projet est à base Maven, interagir avec ce panneau permet de performer des activités Maven
8	Panneau Gradle	Les outils nécessaires pour interagir avec Gradle se trouvent dans ce panneau
9	Panneau de log d'événement	Ce panneau affiche les erreurs et les événements jugés importants et qui doivent être visibles au développeur lors du développement

Tableau 4.1 : le panneau d'Android Studio

2.2.La barre d'outils: [16]

Android Studio comporte une barre d'outils hautement personnalisable qui est facilement accessible à partir de la partie supérieure de l'écran. La barre d'outils par défaut (figure 4.4) est détaillée dans le tableau 4.2

Figure 4.4: La boite outils standard d'Android Studio

1	**Action sur les fichiers**	**Des actions comme ouvrir, enregistrer et synchroniser**
2	Annuler/Rétablir	Permet d'annuler ou rétablir les actions précédentes
3	Couper/copier/coller	Permet de couper rapidement, copier ou coller de la boite outils
4	Trouver/remplacer	Permet de trouver et remplacer des valeurs dans le fichier du projet
5	Navigation	Permet la navigation en avance ou en arrière des les fichiers accédés ou modifiés récemment
6	Construire/exécuter/ débugger/attacher	Les plus utilisé lors du développement, puisqu'ils permettent de construire, exécuter, débugger et attacher les processeurs fonctionnant pour le débogage
7	Paramètres	Permet d'accéder les préférences de l'IDE et la structure du projet
8	Actions Android	Permet de synchroniser le projet avec les fichiers gradle, ouvrir l'AVD ou bien le SDK manager, & ouvrir le moniteur d'application Android
9	Aide	Permet d'utiliser l'aide d'Android Studio

Tableau 4.2 : la barre d'outils d'Android Studio

3. L'application mobile Mahfoud 3

Cette application mobile va être utilisée pour repérer les données de positionnement GPS du robot Mahfoud 3, et les données d'accélération, elle nous permet aussi de lire les données internes du robot mobile pour ensuite les transmettre à une base de données d'un site web, l'autre fonctionnalisée de l'application mobile et de commander le robot Mahfoud III par liaison série USB.

4. Construction de l'application Mahfoud 3 :

4.1. L'interface de l'application :

Le layout de l'application désigne l'interface de cette dernière. C'est une partie importante, car c'est la partie visible de l'application. Pour obtenir une interface agréable, il est souvent nécessaire de réaliser correctement le positionnement des éléments graphiques. La difficulté est d'arriver à programmer un placement qui n'est pas dépendant de l'orientation ou de la taille de l'écran.

Les layout Android sont crées en utilisant la syntaxe XML, pour définir l'interface de l'utilisateur, Ces fichiers XML contiennent des descriptions de différents widgets de l'interface, qui pourraient être des textes, des boutons, ou des images, etc.… [15]

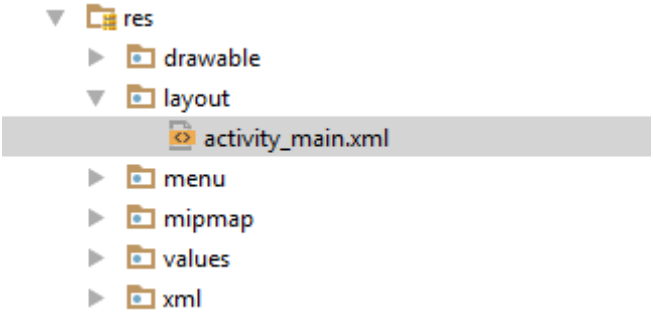


Figure 4.5 :l'emplacement du fichier layout dans le projet Android

Le layout est une ressource (comme le montre la figure 4.5), c'est un morceau de l'application qui n'est pas un code des choses comme des fichiers d'images, des fichiers audio et des fichiers XML.
L'avantage de placer le layout dans le XML c'est que les définitions de l'interface sont séparées à partir du code d'application, de sorte qu'on peut modifier la mise en page sans avoir besoin de changer le code source ou recompiler. on peut créer différentes configurations pour plusieurs orientations de périphériques, tailles d'écran, ou les endroits.

4.1.1. Disposition des éléments placés à l'intérieur de l'application mobile : [17]

L'interface peut avoir plusieurs prédispositions prédéfinies pour les différents objets :

- ✓ LinearLayout: dispose les éléments de gauche à droite ou du haut vers le bas
- ✓ RelativeLayout: les éléments enfants sont placés les uns par rapport aux autres
- ✓ TableLayout: disposition matricielle
- ✓ FrameLayout: disposition en haut à gauche en empilant les éléments

Les éléments de l'interface ont aussi une taille, une largeur et une certaine marge définis comme suit :

- Taille :
  - ❖ android : layout : height= "X " avec$X = fill_parent$ou bien$X = wrap_content$et cela défini l'espace occupé qui peut être espace occupé en hauteur comme il peut être espace occupant ce qui est nécessaire
  - ❖ android : layout : width ="Y" avec$Y = fill_parent$ou bien$y = wrap_content$et ceci pour occuper l'espace en largeur ou seulement ce qui est nécessaire

- Marge externes :
  - ❖ Android : layout_marginBottom="unité" : marge externe en bas
  - ❖ android: layout_marginLeft="unité" marge externe à gauche
  - ❖ android: layout_marginRight="unité" marge externe à droite
  - ❖ android: layout_marginTop="unité" marge externe en haut

Dans le cas de notre application, la disposition de tous les éléments est linéaire, avec un remplissage complet de l'espace de la part des éléments de l'application

```
<LinearLayout
android: orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:layout_alignParentRight="true"
android:layout_alignParentEnd="true">
```

L'application présentée dans la figure 4.6 comporte :

- ✓ Un titre principal de l'application : Mahfoud Link
- ✓ Une check Box
- ✓ Deux boutons Start/Stop pour commencer la commande du robot
- ✓ Deux titres un pour indiquer les données de l'accéléromètre, un autre pour les données GPS
- ✓ Un autre bouton pour envoyer les données récupérées au site web

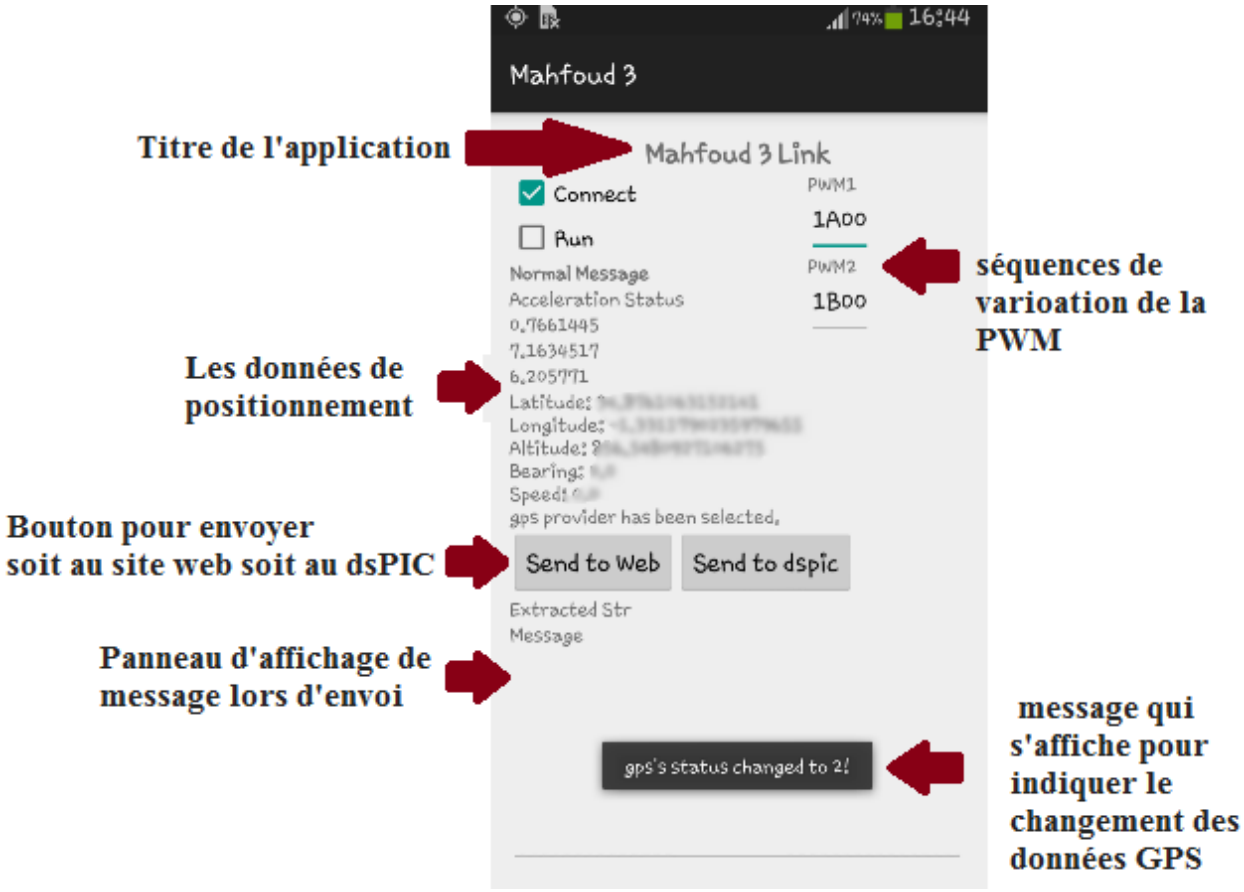


Figure4.6 : aperçu de l'application mobile de Mahfoud III

4.2.Le GPS :

GPS, nom acronyme pour: Système de positionnement à échelle mondiale (Global Positioning System). C'est un réseau de satellites qui émettent en permanence des informations codées. Ces informations permettent d'identifier précisément les positions géographiques sur terre, en mesurant la distance depuis les satellites.

4.2.1. L'utilisation du GPS pour l'application :

Pour que notre application accède aux services de localisation, deux autorisations doivent être demandées, et cela est fait dans le manifeste de l'application comme suit :

```
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
```

On crée l'objet de localisation par :

```
privateLocationManagerlocationManager;
privateMyLocationListenermylistener;
```

L'utilisateur reçoit un gestionnaire de localisation en spécifiant le*LOCATION_SERVICE*en entrée de la méthode*getSystemService*( )de l'activité. Un manager d'emplacement est retourné :

```
// Get the location manager
locationManager= (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
```

Maintenant on initialise le critère de choix du fournisseur de localisation, on a choisi le critère le plus précis : *ACCURACY_FINE*

```
criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_FINE);   // ou ACCURACY_COAST
criteria.setCostAllowed(false);
```

On utilise la méthode *getLastKnownLocation* ( ), qui retourne la dernière position connue par le fournisseur donné.

En outre dans l'application Mahfoud III, on veut gérer les mouvements du robot, et de montrer sa position actuelle, pour cela on utilise un Listener (auditeur) de localisation *LocationListener*, le gestionnaire de localisation *LocationManager* peut

demander $LocationListener$ beaucoup de rappels, afin de gérer l'emplacement ou les changements de status du fournisseur, et avec la méthode $requestLocationUpdates( )$, la localisation obtenue du fournisseur, ou du Listener de localisation va être mise à jour chaque 200 μs et pour $1m$

```
Location location = locationManager.getLastKnownLocation(provider);
mylistener= new MyLocationListener();
if (location != null) {
mylistener.onLocationChanged(location);
} else {
// leads to the settings because there is no last known location
Intent intent = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
startActivity(intent);
}
// location updates: at least 1 meter and 200 millsecs change
locationManager.requestLocationUpdates(provider, 200, 1, mylistener);
Log.d(TAG, "Location init OK");
```

4.3.L'accéléromètre :

4.3.1. Bref aperçu sur l'accéléromètre :

Un accéléromètre est un capteur qui, fixé à un mobile ou tout autre objet, permet de mesurer l'accélération linéaire de ce dernier. On parle encore d'accéléromètre même s'il s'agit en fait de 3 accéléromètres qui calculent les 3 accélérations linéaires selon 3 axes orthogonaux.

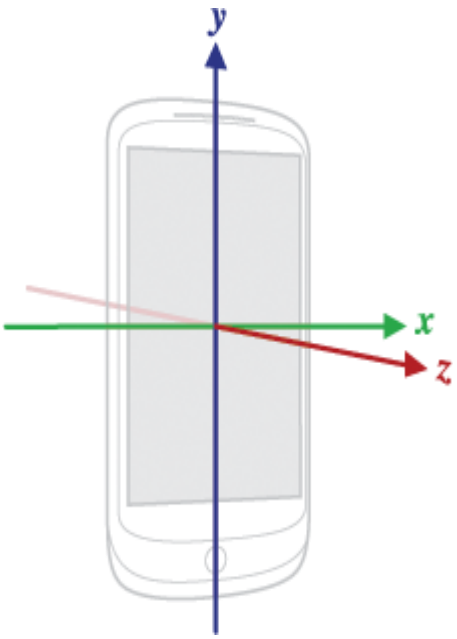


Figure4.7 : les 3 axes d'accélération

4.3.2. Utilisation de l'accéléromètre dans l'application :

Avant d'utiliser l'accéléromètre on demande une permission avec la ligne suivante :

```
<uses-feature
android:name="android.hardware.sensor.accelerometer"android:required="true" />
```

En premier temps on a vérifié si notre appareil supporte le capteur accéléromètre, et qu'il peut exécuter le service spécifique, si l'appareil support l'accéléromètre on initialise le $mSensorManager$ pour que le capteur écoute les événements de l'accéléromètre. Donc on a enregistré un Listener spécifique pour capturer les événements en écrivant dans la méthode $onCreate$ ( ) qui définit le point de départ de l'activité **Main** par le code suivant:

```
mSensorManager= (SensorManager) getSystemService(Context.SENSOR_SERVICE);
if (mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER) != null) {
// success! we have an accelerometer
mAccelerometer=
mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
} else {
Toast.makeText(MainActivity.this, "fail we don't have an accelerometer!",
Toast.LENGTH_SHORT).show();
}
```

Généralement, on ne veut pas que ce genre de Listeners fonctionnent continuellement sur notre appareil, parce que dans la plus part du temps ils consomment de l'énergie et du CPU, pour cela on a désenregistrer Listener quand l'application passe en arrière plan : $onPause$ ( ) dans notre activité, puis l'enregistrer de retour sur $onResume$ ( ) de l'activité, le code pour cela est donc:

```
protected void onResume() {
// Ideally a game should implement onResume() and onPause()
    // to take appropriate action when the activity looses focus
super.onResume();
//sensorManager.registerListener(this, accelerometer,
SensorManager.SENSOR_DELAY_NORMAL);
sensorManager.registerListener(this,
```

```
sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
SensorManager.SENSOR_DELAY_NORMAL);//SENSOR_DELAY_GAME
```

```
protected void onPause() {
// Ideally a game should implement onResume() and onPause()
    // to take appropriate action when the activity looses focus
super.onPause();
sensorManager.unregisterListener(this);
```

Ensuite, il ne reste qu'à remplacer les deux méthodes qui sont associées à $SensorEventListener$ qui sont $onAccurancyChnaged$ et $Sensor\ Chnaged$:

```
public final void onAccuracyChanged(Sensor sensor, int accuracy)
{
// Do something here if sensor accuracy changes.
}
@Override
public final void onSensorChanged (SensorEvent event)
    {
// Many sensors return 3 values, one for each axis.
float x = event.values[0];
float y = event.values[1];
float z = event.values[2];
//display values using TextView
tv.setText("X acceleration:" +"\t\t"+x);
tv1.setText("Y acceleration:" + "\t\t" +y);
tv2.setText("Z acceleration:" +"\t\t" +z);
    }
```

On a quelque chose de similaire pour le GPS

```
private class MyLocationListener implements LocationListener {

        @Override
        public void onLocationChanged(Location location) {
            // Initialize the location fields
            Savedlocation = location;
            String s = "Latitude: " + String.valueOf(location.getLatitude());
            s = s + "\nLongitude: " + String.valueOf(location.getLongitude());
            s = s + "\nAltitude: " + String.valueOf(location.getAltitude());
```

```
            s = s + "\nBearing: " + String.valueOf(location.getBearing());
            s = s + "\nSpeed: " + String.valueOf(location.getSpeed());
            s = s + "\n" + provider + " provider has been selected.";
// affiche todo
            TextView dummyTextView = (TextView) findViewById(R.id.LocalizTxt);
            dummyTextView.setText(s);
            Toast.makeText(MainActivity.this, "Location changed!" + s,
                  Toast.LENGTH_SHORT).show();
      }
```

4.4.Liaison série USB :

Un câble USB OTG est nécessaire pour relier le robot et le smartphone comme le montre la figure suivante :

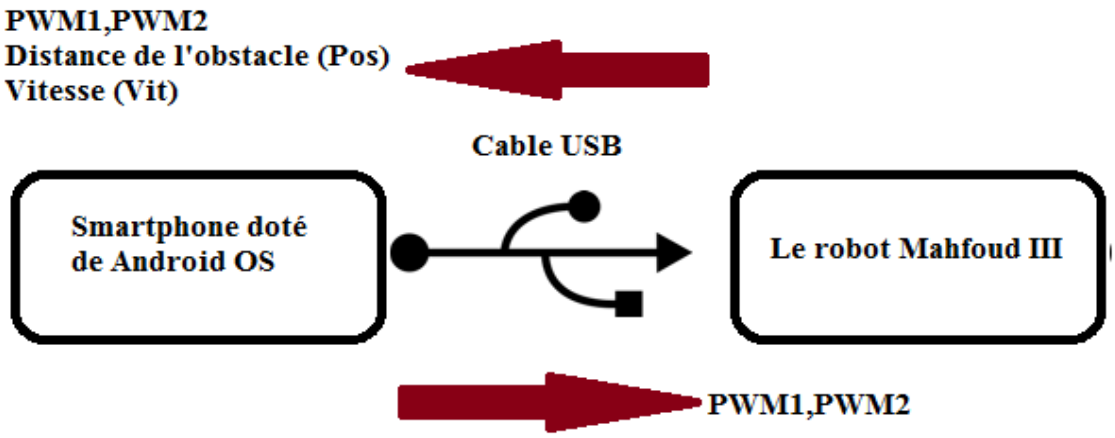


Figure 4.8 : Liaison Robot/smartphone avec les données échangées

Le smartphone dispose de plusieurs fonctionnalités intéressantes (appareil photo, des accéléromètres, des haut-parleurs, un microphone, adaptateur Wifi...) qui font d'excellents ajouts à notre robot, on s'intéresse essentiellement au GPS, à l'accéléromètre et à l'interface http (via Wifi ou 3G).

4.4.1. Le câble USB :

La communication série entre le robot et le smartphone est assuré par le biais d'un câble USB. Le bus USB (Universal Serial Bus, en français Bus série universel) est, comme son nom l'indique, basé sur une architecture de type série. Il s'agit toutefois d'une interface entrée-sortie beaucoup plus rapide que les ports sérient standards.

La connectique est du mini USB sur la carte dsPIC et micro USB OTG sur le smartphone.

4.4.2. Communication UART/USB:

Le MCP2200 est un convertisseur série USB vers UART, il assure la connexion par USB entre le dsPIC et le smartphone.

Le MCP2200 dispose de 8 pins I/O à usage général, et 4 pins indiquant les statuts de la communication USB. Il permet de dialoguer avec un protocole CDC over USB, rendant possible la communication en port COM série classique reconnue par Windows, Linux ou Android.

Une autre alternative est le FTDI 232 mais qui est plus cher et plus difficile à souder (0.65 mm entre les pins contre 1.27 mm entre les pins du MCP2200).

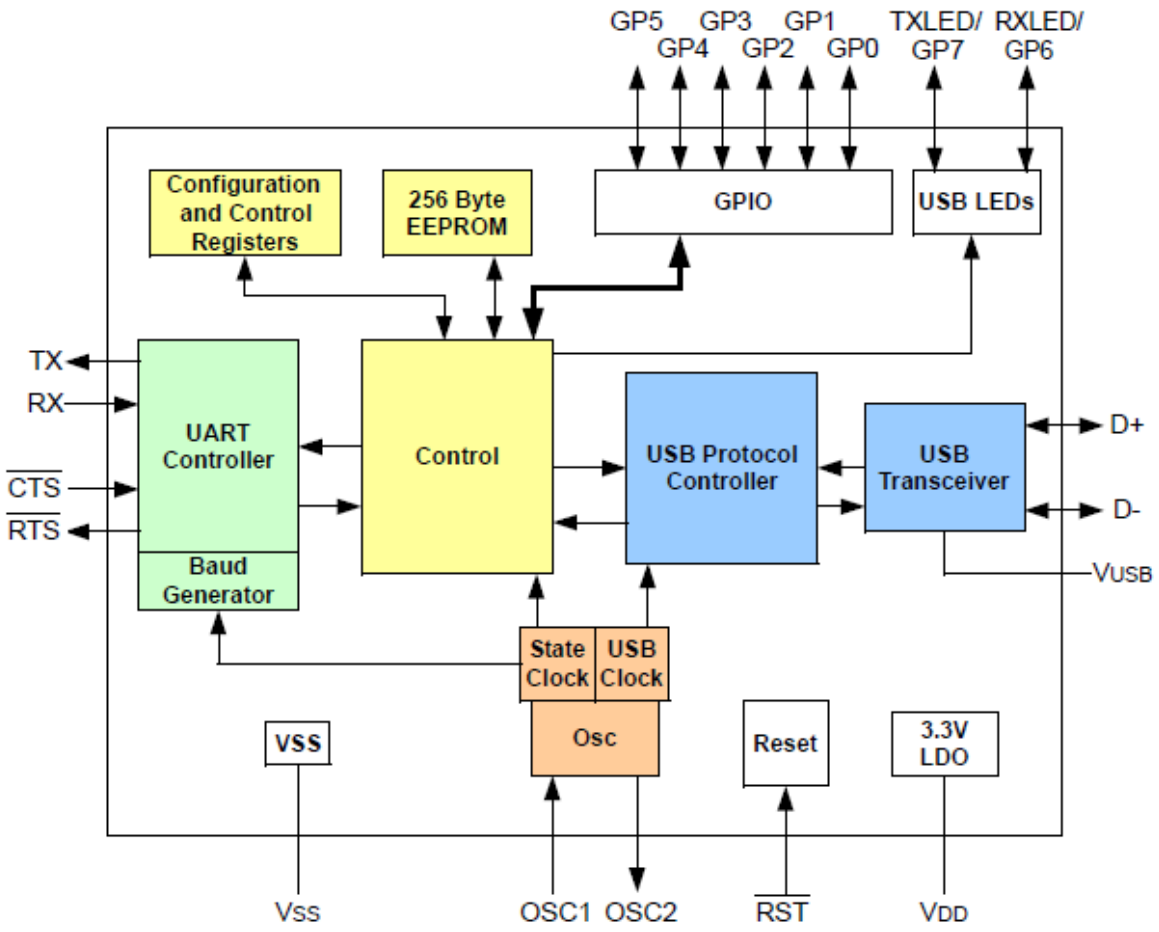


Figure 4.9 : Diagramme du MCP2200

Les pins D+, D-, et VUSB sont les pins du câble USB qu'on a détaillé.

Les pins qui nous intéressent le plus dans ce diagramme sont : CTS, RTS, TX et RX, car c'est à travers ces pins que la communication prend lieu.

4.4.3. L'interface UART:

L'interface UART du MCP2200utilise TX et RX pour le transfert de données, et RTS/CTS pour le contrôle de flux des données, l'UART peut être configurée pour différentes vitesses de transmissions connues par le terme baud et qui sont normalisées : 19200, 9600, 4800, 2400, 1200 bauds.

Le contrôle de flux matériel utilise les broches RTS et CTS comme une poignée de main entre le MCP2200 et le dsPIC (figure 4.10). La broche RTS du MCP2200 est généralement reliée à la CTS du dsPIC

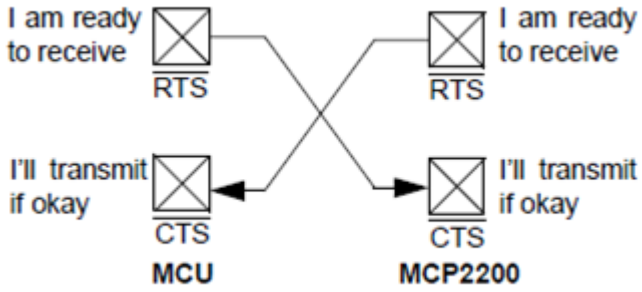


Figure4.10 : Exemple de connexion RTS/CTS

Le pin RTS est une sortie active à l'état bas, et qui informe le dsPIC quand il est prêt à recevoir des données.

Le pin CTS est une entré active a l'état bas aussi, qui informe l'MCP2200 quand il est prêt à envoyer des données.

4.4.4. Transmission de données :

Le MCP2200 et le dsPIC échangent des informations en full duplex : ils sont à la fois émetteurs et récepteurs.

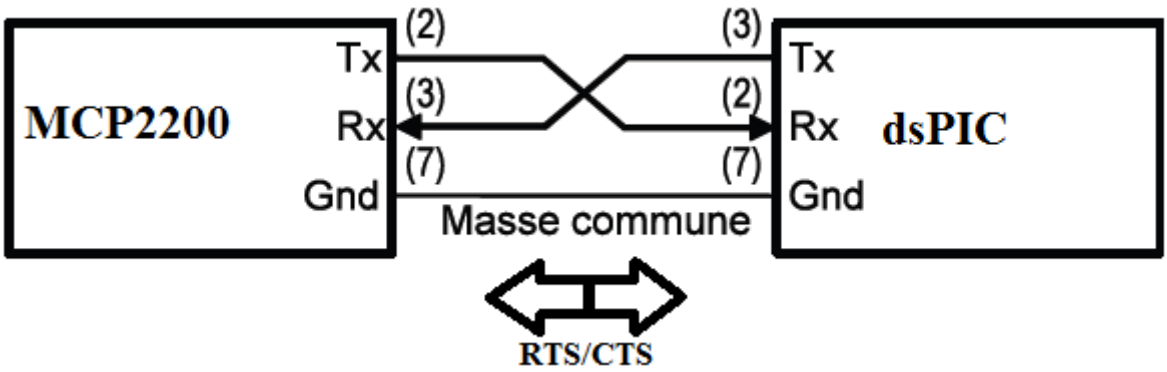
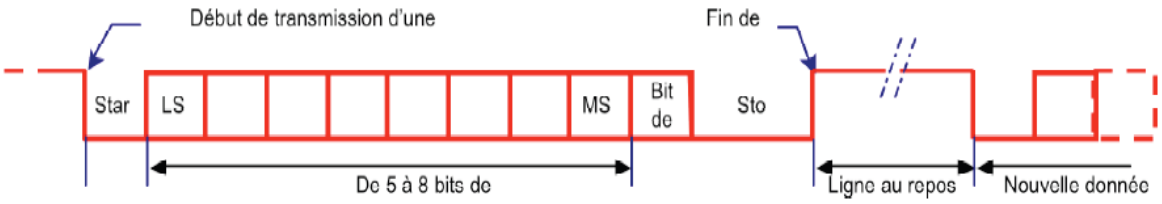


Figure4.11 : Liaison série asynchrone entre le MCP2200 et le dsPIC

Dans ce type de liaison asynchrone, la ligne des données véhicule des informations binaires à une cadence prédéfinie. La séquence est représentée sur la Figure 4.12.



Figue 4.12 : séquence de transmission sur la ligne de données

Au repos la ligne des données supporte un signal au niveau logique haut (1). Dès le début de la transmission, la ligne passe au niveau bas (0) : c'est le START. La donnée est transmise sur la ligne Tx en commençant par le bit de poids faible, reçue sur la ligne Rx du récepteur et la donnée est reconstituée (mémorisée). La longueur du mot peut varier de 5 à 8 bits suivant le code des données utilisé. Après le bit de poids fort de la donnée, un bit supplémentaire peut être ajouté pour effectuer un contrôle de transmission. Après ce dernier bit, la ligne passe au niveau logique 0 pendant une durée de 1, 1.5 ou 2 fois la durée attribuée à un bit : c'est le STOP. La ligne repasse alors au repos en attendant une nouvelle donnée. La durée d'attente est quelconque, imposée par le rythme des informations à transmettre, d'où la dénomination d'asynchrone.

La communication peut avoir lieu dans chaque sens car l'UART est un canal full duplex donc l'émission et la réception sont simultanées.

4.4.5. Codage de l'information :

La longueur des données transmises dépend de la liaison. Elle est fixée au préalable et connues de l'émetteur et du récepteur, les codes se sont multipliés, mais c'est le code ASCII qu'on a utilisé.

Les données échangées entre le robot et le smartphone sont les valeurs des commandes PWM1 et PWM2 pour faire bouger le robot et la validation ou invalidation du hacheur (Enable / Disable) et en retour, le Smartphone reçoit la position des obstacles pour qu'il les évite et la vitesse des roues du robot.

Pour la liaison série entre le robot et le smartphone, nous avons utilisé une bibliothèque fournie par [18]. Elle permet au Smartphone de se connecter au MCP2200 et donc au dsPIC puis d'échanger les informations.

La méthode qu'on a utilisé repose sur l'utilisation d'un Listener :

```
setFilters();  // Start listening notifications from UsbService
startService(UsbService.class, usbConnection, null); // Start UsbService(if it was not started before) and Bind it
```

Comme pour tous les Listeners on l'arrête dans le $onPause()$ de l'activité :

```
unregisterReceiver(mUsbReceiver);
unbindService(usbConnection);
```

L'affichage de données reçus du robot se fait en utilisant un $handler$

```
@Override
public voidhandleMessage(Message msg) {
//Toast.makeText(mActivity.get(), "received", Toast.LENGTH_SHORT).show();
    //Log.d(TAG, "msg RX");
switch (msg.what) {
case UsbService.MESSAGE_FROM_SERIAL_PORT:
            String data = (String) msg.obj;
// Log.d(TAG, "msg:" + data);
            //mActivity.get().msgText.append(data);
RXbuffer+=data;
intdebut = RXbuffer.indexOf("$");
intfin = RXbuffer.indexOf("\r", debut);
if (fin !=-1){
                String ExtractedStr = RXbuffer.substring(debut, fin);
RXbuffer= RXbuffer.substring(fin, RXbuffer.length());
mActivity.get().ExtStr.setText( ExtractedStr);
Log.d(TAG, "msg:" + ExtractedStr);
Log.d(TAG, "buf:" + RXbuffer);
            }
break;
    }
```

5. Le site web et la base de données :

Le smartphone qui est connecté au robot par un câble USB envoi des données de positionnement au site web, la connexion entre les trois est donnée par la figure suivante :

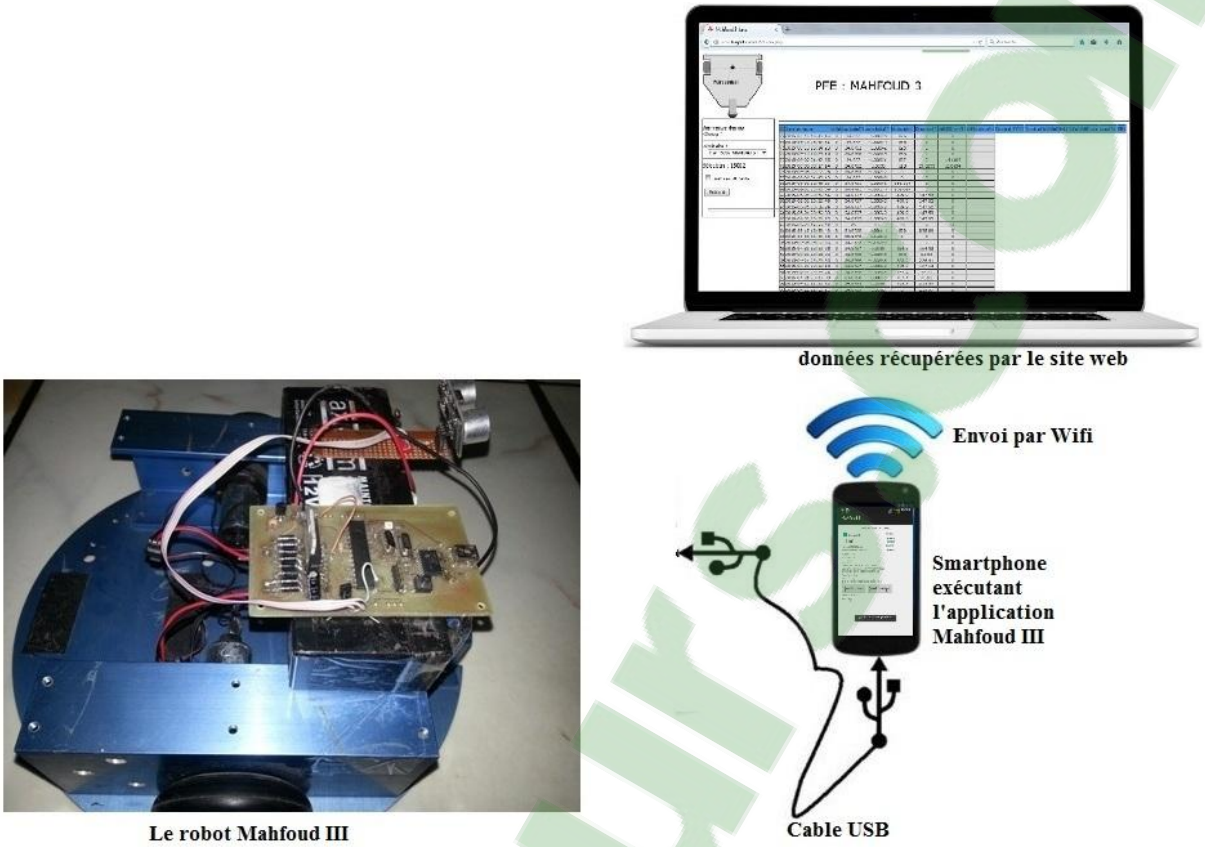
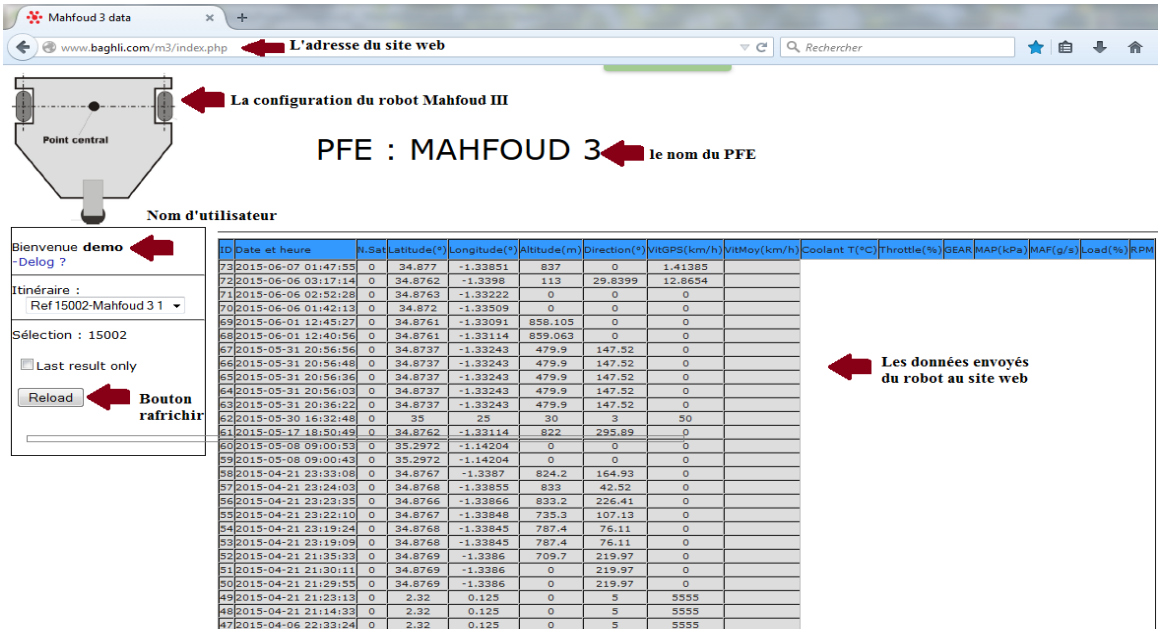


Figure 4.13 : liaison entre le robot et le smartphone et envoi des données

Les données récupérées par le smartphone du robot sont ensuite envoyés à une base de données (figure 4.14) sur un serveur distant.

Un site web associé permet de lire la BDD et d'afficher les données à l'aide d'un navigateur web à partir de n'importe quel PC connecté. La page web comporte :

- ✓ Un titre de la page
- ✓ Un bouton rafraichir
- ✓ Un tableau dans lequel les données envoyées sont stockées



ID	Date et heure	N.Sat	Latitude(°)	Longitude(°)	Altitude(m)	Direction(°)	VitGPS(km/h)	VitMoy(km/h)	Coolant T(°C)	Throttle(%)	GEAR	MAP(kPa)	MAF(g/s)	Load(%)	RPM
73	2015-06-07 01:47:55	0	34.877	-1.33851	837	0	1.41385								
72	2015-06-06 03:17:14	0	34.8762	-1.3398	113	29.8399	12.8654								
71	2015-06-06 02:52:28	0	34.8763	-1.33222	0	0	0								
70	2015-06-06 01:42:13	0	34.872	-1.33509	0	0	0								
69	2015-06-01 12:45:27	0	34.8761	-1.33091	858.105	0	0								
68	2015-06-01 12:40:56	0	34.8761	-1.33114	859.063	0	0								
67	2015-05-31 20:56:56	0	34.8737	-1.33243	479.9	147.52	0								
66	2015-05-31 20:56:48	0	34.8737	-1.33243	479.9	147.52	0								
65	2015-05-31 20:56:36	0	34.8737	-1.33243	479.9	147.52	0								
64	2015-05-31 20:56:03	0	34.8737	-1.33243	479.9	147.52	0								
63	2015-05-31 20:36:22	0	34.8737	-1.33243	479.9	147.52	0								
62	2015-05-30 16:32:48	0	35	25	30	3	50								
61	2015-05-17 18:50:49	0	34.8762	-1.33114	822	295.89	0								
60	2015-05-08 09:00:53	0	35.2972	-1.14204	0	0	0								
59	2015-05-08 09:00:43	0	35.2972	-1.14204	0	0	0								
58	2015-04-21 23:33:08	0	34.8767	-1.3387	824.2	164.93	0								
57	2015-04-21 23:24:03	0	34.8768	-1.33855	833	42.52	0								
56	2015-04-21 23:23:35	0	34.8766	-1.33866	833.2	226.41	0								
55	2015-04-21 23:22:10	0	34.8767	-1.33848	735.3	107.13	0								
54	2015-04-21 23:19:24	0	34.8768	-1.33845	787.4	76.11	0								
53	2015-04-21 23:19:09	0	34.8768	-1.33845	787.4	76.11	0								
52	2015-04-21 21:35:33	0	34.8769	-1.3386	709.7	219.97	0								
51	2015-04-21 21:30:11	0	34.8769	-1.3386	0	219.97	0								
50	2015-04-21 21:29:55	0	34.8769	-1.3386	0	219.97	0								
49	2015-04-21 21:23:13	0	2.32	0.125	0	5	5555								
48	2015-04-21 21:14:33	0	2.32	0.125	0	5	5555								
47	2015-04-06 22:33:24	0	2.32	0.125	0	5	5555								

Figure4.14 : aperçu sur le site web

6. Résultat et exploitation:

- ✓ On a réussi à envoyer les données de positionnement au site web à travers l'application
- ✓ On a réussi à établir une connexion USB série entre le robot et le smartphone

Par manque de temps, on n'a pas réussi à envoyer les commandes PWM au robot pour le commander par le smartphone

On n'a pas aussi réussi à terminer l'application pour afficher les données récupérées par le smartphone et à les envoyer sur la BDD.

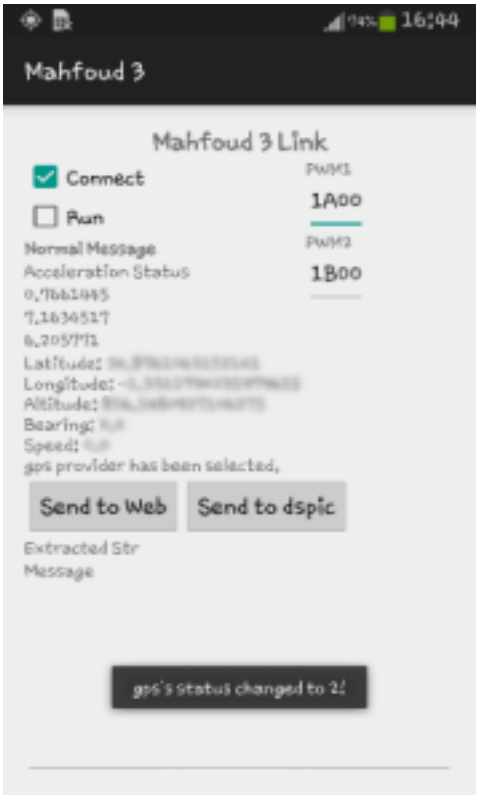


Figure4.14:Aperçu sur l'application Mahfoud 3

7. Conclusion :

Nous avons détaillé dans ce chapitre l'application mobile développée dans le cadre de ce projet et qui sert à récupérer les données de positionnement du robot, communiquer avec le robot par une liaison série via un câble USB et échanger des données avec le robot, la première partie de l'application fonctionne avec succès, la partie communication est réussie aussi, la partie échange des données n'a pas été complètement faite vu le manque de temps, parce que la partie communication était délicate est qui a nécessité l'aide de note encadreur et l'aide du développeur du code source qu'on a utilisé, afin de compléter l'application il reste donc l'extraction des données reçues du dsPIC et les afficher sur l'interface de l'application Android, il reste aussi à faire l'envoi et la lecture simultanées des commandes PWM du robot.

Conclusion générale

Ce projet de fin d'étude est un projet vaste qui regroupe plusieurs disciplines, et qui requiert des connaissances et des capacités multidisciplinaires (programmation Java, Android, programmation C, dsPIC).

Il nous a permis de mettre en œuvre les connaissances requises le long de notre cursus, il nous a donné l'opportunité de construire et contrôler un robot mobile, il nous a permis aussi d'ouvrir l'esprit sur la robotique mobile et sur la programmation mobile,

A travers ce projet on a exploré la puissance du dsPIC33F, ce dernier était le cœur du robot et du projet, sa programmation a permis de commander le robot Mahfoud III qui dispose de deux moteurs à courant continu, on est arrivé à lui donner la capacité de se mouvoir dans son environnement. L'association des capteurs infrarouges a donné au robot plus d'autonomie, en évitant des obstacles sans notre intervention.

Le développement de l'application mobile qui n'était pas une chose facile, nous a offert la chance de découvrir le monde de la programmation mobile, cette application avait pour but d'extraire les données GPS et les données d'accéléromètre, et vu que le smartphone possédant cette application est mis sur le robot, ces données peuvent être considérés comme les données du robot mobile et cela pour ne pas ajouter d'autres capteurs et d'autres composant a la carte de commande, l'application mobile sert aussi à commander le robot par une liaison série USB.

Vu le manque du temps nous n'avions pas pu tout faire comme par exemple la commande du robot à travers l'application mobile, l'échange des données entre le robot et le smartphone.

Comme perspectives à ce travail, on peut mentionner le suivi de ligne grâce à la bibliothèque graphique OpenCV(Open Computer Vision) en utilisant la caméra du smartphone par exemple.

Enfin, ce fut un immense plaisir de travailler sur ce genre de projet très intéressant, de découvrir de nouveaux aspects, on espère que Mahfoud continuera à se développer avec de futurs PFE.

Bibliographie

[1] : Laetitia Matignon, « Introduction à la robotique », Université de Caen France, 2012.Disponible : iris.cnrs.fr/laetitia.matignon/index/coursL1robotique.pdf

[2] : Rémy GUYONNEAU, «Méthodes ensemblistes pour la localisation en robotique mobile », thèse de doctorat, l'ISTIA, école d'ingénieur de 'université d'Angers, 2013

[3] : Bernard BAYLE, « Robotique mobile », université de Strasbourg, 2012

[4] : Benmakhlouf Abdeslam, «Contrôleur flou pour la navigation d'un robot mobile d'intérieur », thèse de magister, université de Batna, 2006

[5] : J.Borenstein, H.R.Everett&L.Feng, "Where am I? Sensor and methods for mobile robot positioning", University of Michigan,1996

[6] : Louis COUFFIGNAL, « Moteur courant continu », Strasbourg, 2006

[7] : Ralph AVANZINI,"La commande des moteurs pas à pas : Les circuits intégrés spécialisés», disponible sur:http://tcremel.free.fr/circuits_de_commande.html

[8] : Michel KREUTNER,"Perception multi sensorielle pour le positionnement d'un véhicule autonome dédié aux personnes handicapés moteurs», Université de Metz, Septembre 2004

[9] : Les codeurs incrémentaux MEnc 13, disponible sur :www.micropik.com

[10] : Le double pont H L298, disponible sur :http://ww2.ac-poitiers.fr/

[11] : D.GRIDAINE, « Codeurs rotatifs industriels », lycée les Lombards, France

[12] : Schneider Electric, « guide des solutions d'automatisme », France, 2008

[13] : Marc Cabinet Ballot SchmitRiou, « Mémoire tampon à adressage modulo », 11 octobre 1995

[14] : dsPIC33FJ128MC802, disponible sur :http://wiki.electrolab.fr

[15] : M. Dalmau, « Développement d'applicationspour Android »,

[16] : Mike WOLFSON, « Android Développer Tools Essentials », USA, 2013

[17] : Jean-François Lalande, « Développement sous Android », ENSI de Bourges, 2013

[18] : felhr85, USBCOM Library, disponible sur : https://github.com/felHR85/UsbSerial/tree/master/src/com/felhr

Annexes

1. Le schéma de la carte de Mahfoud III :

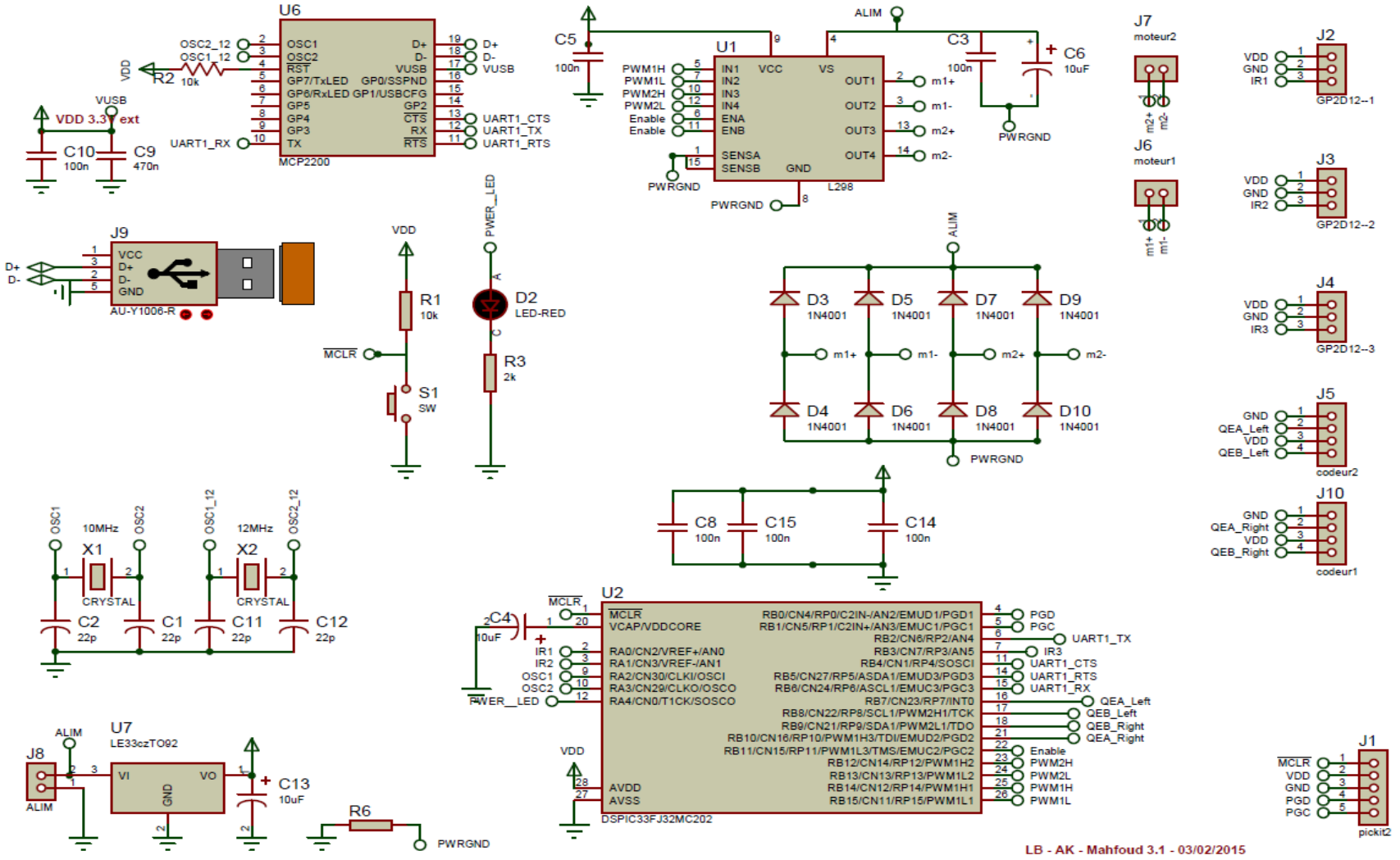


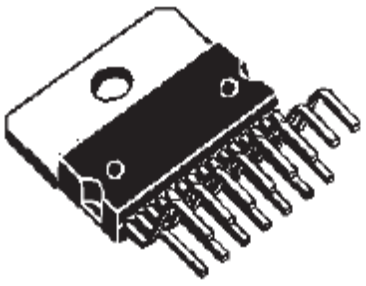
Schéma électronique de la carte de commande de Mahfoud III

2. Moteur A-max :

- Tension nominale : 12V
- Vitesse à vide :12300tr/min
- Courant à vide : 0.954A
- Vitesse nominale : 6660tr/min
- Couple nominal : 0.217 Nm
- Courant nominal : 0.243A

3. Double pont-H L298 :

- Circuit intégré L298N, double pont en H de pilotage.
- Boîtier Multiwatt15
- Tension d'alimentation du circuit logique : 5 V
- Tension d'alimentation du circuit puissance : 0 V - 48 V (La tension utilisée est de 6 à 8V)
- Tension minimale de sensibilité MLI : 2.3 V Courant maximal du circuit puissance : 3A


Multiwatt15

4. Convertisseur USB vers série MCP2200 :

- Dimensions:12.8 x 7.5 x 2.05mm
- Débit de données:12Mbit/s
- Hauteur:2.05mm
- Largeur:7.5mm
- Longueur:12.8mm
- Nombre de broche:20
- Nombre de transceivers:1

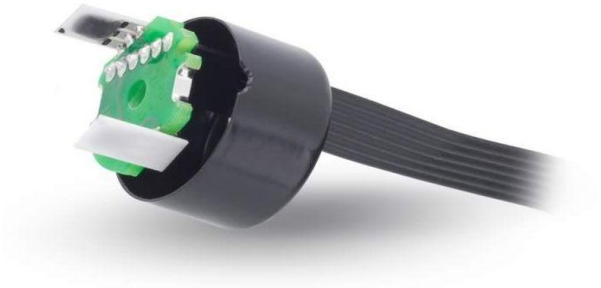
- Protection ESD: Oui
- Protocoles supportés: USB 2.0
- Température d'utilisation maximum:+85 °C
- Température de fonctionnement minimum:-40 °C
- Tension d'alimentation asymétrique typique:5,5 V
- Type d'alimentation: Analogique

5. Capteur Infrarouge GP2Y0A21YK0F :

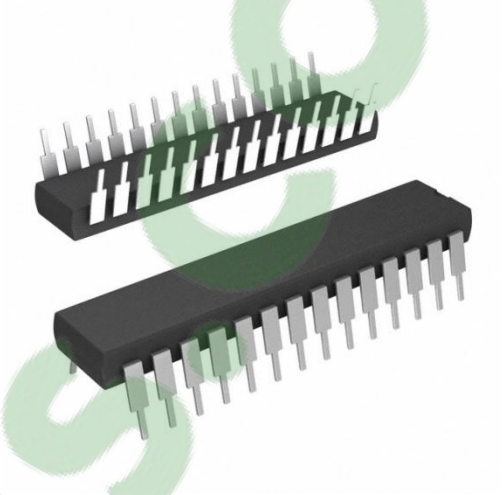
- Alimentation : 4.5V, 5.5V
- Courant de fonctionnement : 35mA
- Signal de sortie: Analogique
- Distance de détection : de 10cm à 80cm

6. Codeur incrémental MEnc 13 :

- Impulsions par tour : 16 impulsion/tour
- Nombre de canaux : 2
- Fréquence maximale : 20kHz
- Vitesse de rotation : maximale : 75000tr/min

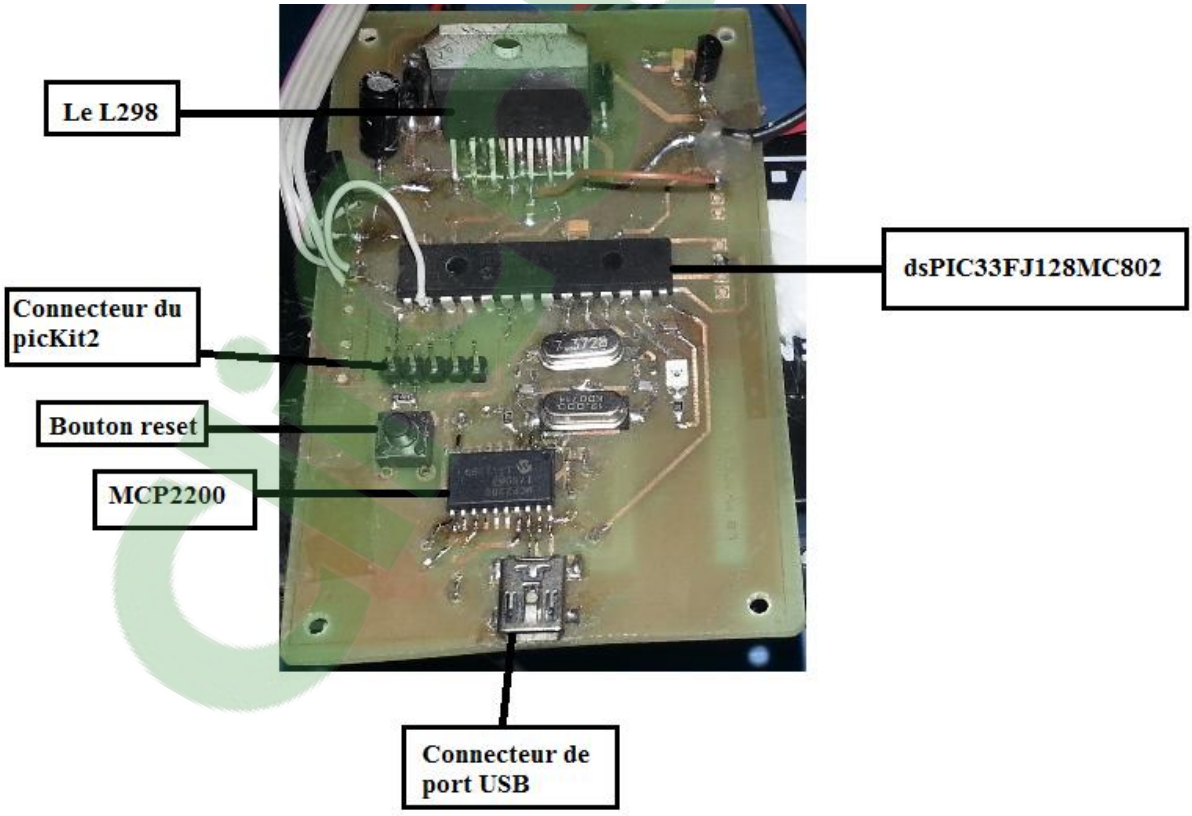


7. dsPIC33FJ128MC802:

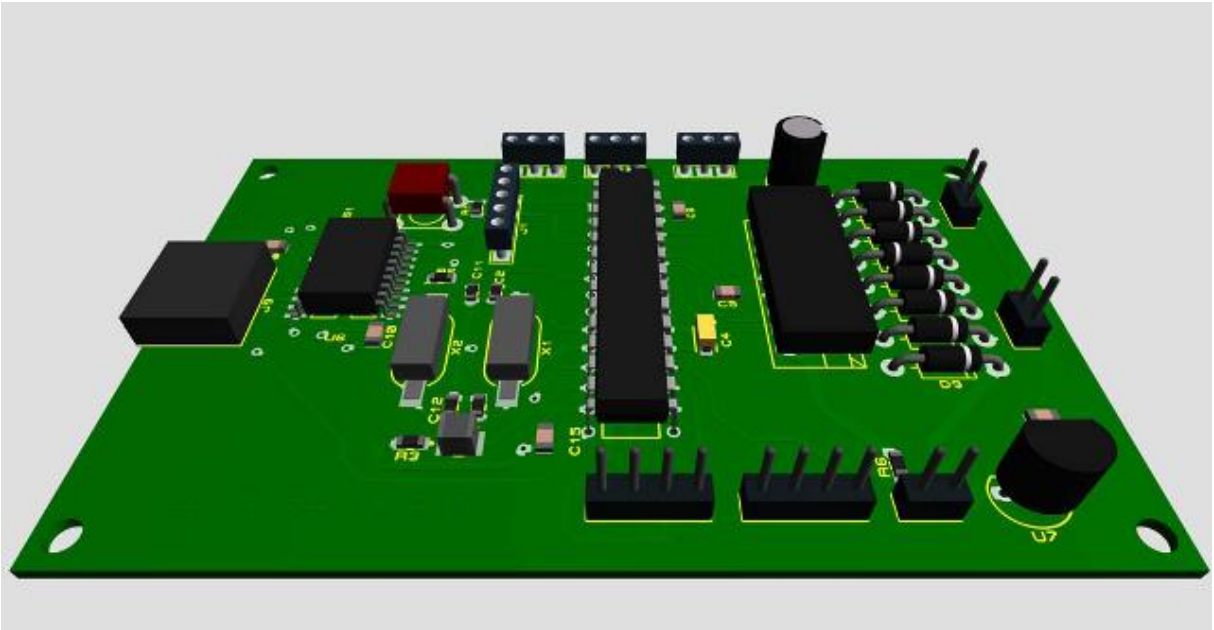


- CPU Haute performance.
- Mémoires SRAM, Flash.
- entrée-sortie (I/O) digitales.
- 6 canaux16-bits pour le contrôle de moteur PWM.
- ADC (analog -to-digital converter).
- DAC (digital -to- analog converter).
- Communication série UART.
- Compteurs(Timers).
- Programmation en C
- Faible énergie consommée

8. Carte de commande du robot Mahfoud III :



- ✓ **Vue de la carte en 3D**



Résumé

Ce projet consiste à créer un robot mobile appelé Mahfoud III, notre objectif était de rendre le robot plus autonome et interactive, pour cela, nous avons inséré un capteur à ultrasons, qui vise à donner au robot la possibilité d'éviter les obstacles qui lui rencontre dans sa trajectoire, une application mobile développée sous Android studio est également construit dans l'objectif de rendre le robot interagit avec un site en envoyant ses information internes à ce dernier, la communication entre le robot et le smartphone est assurée par un câble USB, l'extraction a partir de l'application mobile de la position actuelle et l'accélération du mobile est plus facile que d'associer plus de composants et de capteurs a sa carte de commande, l'application permet également de contrôler le robot.

Mots clés : Robot, capteur, ultrasonique, control, android, application, site web

Abstract

This project is about creating a mobile robot called Mahfoud III, our aim was to make it more autonomous and interactive, for that, we inserted an ultrasonic sensor, that aims at giving the robot the possibility of avoiding the obstacles that face him in his trajectory, a mobile application developed under Android Studio is also built at the aim of making the robot interact with a website by sending its internal information to this last, the communication between the robot and the smartphone is assured by a USB cable, through the application extracting the mobile's current position and acceleration is easier than associating more components and sensors to its control board, the application allows also to control the robot.

Keywords: Robot, sensor, ultrasonic, control, android, application, website

ملخص

هذا المشروع هو حول إنشاء الروبوت المتحرك المسمى محفوظ الثالث، كان هدفنا جعله أكثر استقلالية وتفاعلية، لذلك،أدراجنا جهاز استشعار بالموجات فوق الصوتية، الذي يهدف إلى إعطاء الروبوت إمكانية تجنب العقبات التي يواجهها في مساره ، قمنا بتطوير تطبيق للهاتف الذكي الذي طورناه تحت برنامج اندرو يد اتستوديو هذا التطبيق يهدف إلى جعل الروبوت يتفاعل مع موقع على شبكة الانترنت عن طريق إرسال معلوماته الداخلية لهذا الأخير. التواصل بين الروبوت والهاتف الذكي يتم عن طريق كابل ، التطبيق يسمح كذلك بتوفير معلومات حول الوضع الحالي للروبوت و كذا تسارعه هذا الطريقة أسهل من ربط أكثر من مكونات وأجهزة استشعار للوحة التحكم في الروبوت، وتطبيق الهاتف الذكي يسمح أيضا بالتحكم في الروبوت

الكلمات المفتاحية : روبوت ، جهاز استشعار، فوق صوتي ، تحكم ، اندرو يد ، تطبيق، موقع الكتروني