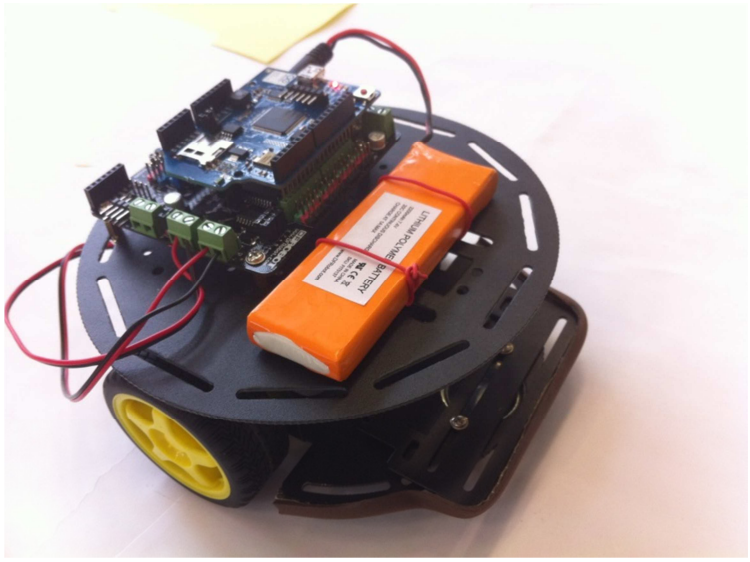
Ecole Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33(0)2 47 36 14 14
Fax. +33(0)2 47 36 14 22
www.polytech.univ-tours.fr

Spécialité Informatique Industrielle
4ème année
2012 – 2013

Rapport de Projet Collectif Système et Réseau

Développement d'une voiture télécommandée à base d'Arduino



Version du 23 Janvier 2013

Sommaire

Introduction

Le projet de Réseau et Communication constitue un complément de formation aux apprentis. La finalité de ce projet est de proposer une voiture télécommandée pilotée à distance par un PC. Les objectifs sont divers mais peuvent être placés dans l'une de ces catégories :

- l'environnement technique : l'objectif est de capitaliser une expérience supplémentaire dans un domaine particulier, ici, l'Arduino et sa communication. Innovation, réflexion, méthodologie sont des qualités importantes que nous avons appliqué afin de mener des travaux d'ingénierie.
- La gestion de projet : c'est grâce à la problématique technique que nous avons pu instaurer un environnement de travail en mode projet. Le but est d'amener chaque collaborateur vers un niveau organisationnel supérieur et notamment permettre de mieux planifier les activités d'un projet, instaurer un travail en équipe et privilégier la communication.

Le rapport technique ci-dessous reprend notre réflexion et la façon dont nous avons procéder afin d'aboutir à ce résultat. Il est possible que ce projet soit reprit par un autre groupe, il nous semblait important de garder à l'esprit le lecteur de ce rapport et de nous poser les questions suivantes : "Que savions-nous sur le sujet à son commencement ?" mais aussi "Que faut-il pour continuer ?"

Ce rapport se décline de la manière suivante, nous aborderons dans un premier temps les spécificités techniques et l'analyse liées à la voiture et ce que nous avons pu en conclure. Nous verrons l'élaboration du programme, des trames, de l'interopérabilité PC/voiture. Nous finirons sur les solutions apportées aux problèmes rencontrés.

1. Cahier des charges fonctionnel

Ce cahier des charges est uniquement un document de conception, c'est à dire qu'il ne comporte que les bases nécessaires à la conception du produit souhaité. Les solutions techniques seront expliquées plus tard.

1.1. Objectifs de la mission

Créer des voitures télécommandées à distance (sans fil), à base d'Arduino, contrôlable par ordinateur. L'utilisateur pourra contrôler la voiture dans les 4 directions, l'arrêter et modifier sa vitesse. Objectifs sur le long terme : être capable de proposer la construction des voitures à des élèves débutants. De plus, nous voulons proposer un système simple capable d'être réalisé en série pour des expositions. Le code ainsi que tous les autres documents devront être clairs, commentés et précis pour permettre la reprise du projet en vue d'améliorations.

1.2. Non-objectifs

Si l'utilisateur démarre plusieurs véhicules, il devra être capable de les contrôler en essaim. Cette gestion de l'essaim ne sera pas implémentée dans ce projet (en revanche cet objectif sera pris en compte lors de la construction des prototypes).

1.3. Contraintes

a. Général

- Le budget global pour la voiture finale devra être limité (< 500€)
- La voiture devra être à base de composants éprouvés et de conception simple.
- Le prototype doit être évolutif en vue d'amélioration (ajout capteur(s) ou fonctionnalité(s))
- Tous les composants de la voiture devront être achetés dans le commerce avec une possibilité d'achat en grande quantité, en vue d'une construction en série.

b. Matériel

- Le châssis et toutes les parties mécaniques seront achetés dans le commerce.
- La voiture aura une longueur de 20cm au maximum.
- L'alimentation devra avoir une autonomie d'au moins une demi-heure, électrique, compacte et suffisante pour alimenter tous les modules embarqués.
- Le déplacement de la voiture télécommandée se fera exclusivement sur une surface plane, lisse et horizontale. La propulsion sera assurée par un ou plusieurs moteurs électriques directement sur les roues. La solution avec des chenilles pourra aussi être étudiée.
- 2 moteurs seront présents sur la voiture, chaque moteur contrôlera une roue.
- La communication devra être par Wifi ou par Bluetooth

- 2 capteurs à pression (1 à l'avant - 1 à l'arrière) pour la détection des collisions frontales.
- la partie électronique de la voiture sera hébergée sur une carte-mère gérant une ou des carte(s) fille(s) (la carte-mère sera une carte Arduino).
- Les voitures n'intégreront aucune remontée visuelle (voyants, écrans, IHM...)

1.4. Programmes

- Les programmes devront être le plus clairs et le plus structurés possible (possibilité d'évolution par une tierce personne)
- Le programme sur l'ordinateur sera capable de contrôler une seule voiture ou tout l'essaim.
- Les voitures devront retourner l'état de leurs capteurs à l'ordinateur.
- Le programme sur ordinateur devra être portable (application légère, et indépendante du matériel) avec une IHM claire et intuitive pour l'utilisateur (contrôle des voitures avec les flèches directionnelles par exemple).

1.5. Livrables

Les documents ainsi que les produits suivant devront être livrés à la fin du projet:

a. Matériel :

- Les voitures télécommandées

b. Programme :

- L'application utilisée pour contrôler la (ou les) voiture(s) télécommandée(s)
- Le programme présent dans la carte Arduino

c. Documents :

- Code source des programmes avec leurs documentations
- Guide d'utilisation du programme sur ordinateur
- Documentation matériel de la voiture
- Analyse fonctionnelle
- Documentations constructeurs de chaque composant utilisé pour le projet
- Schéma électrique

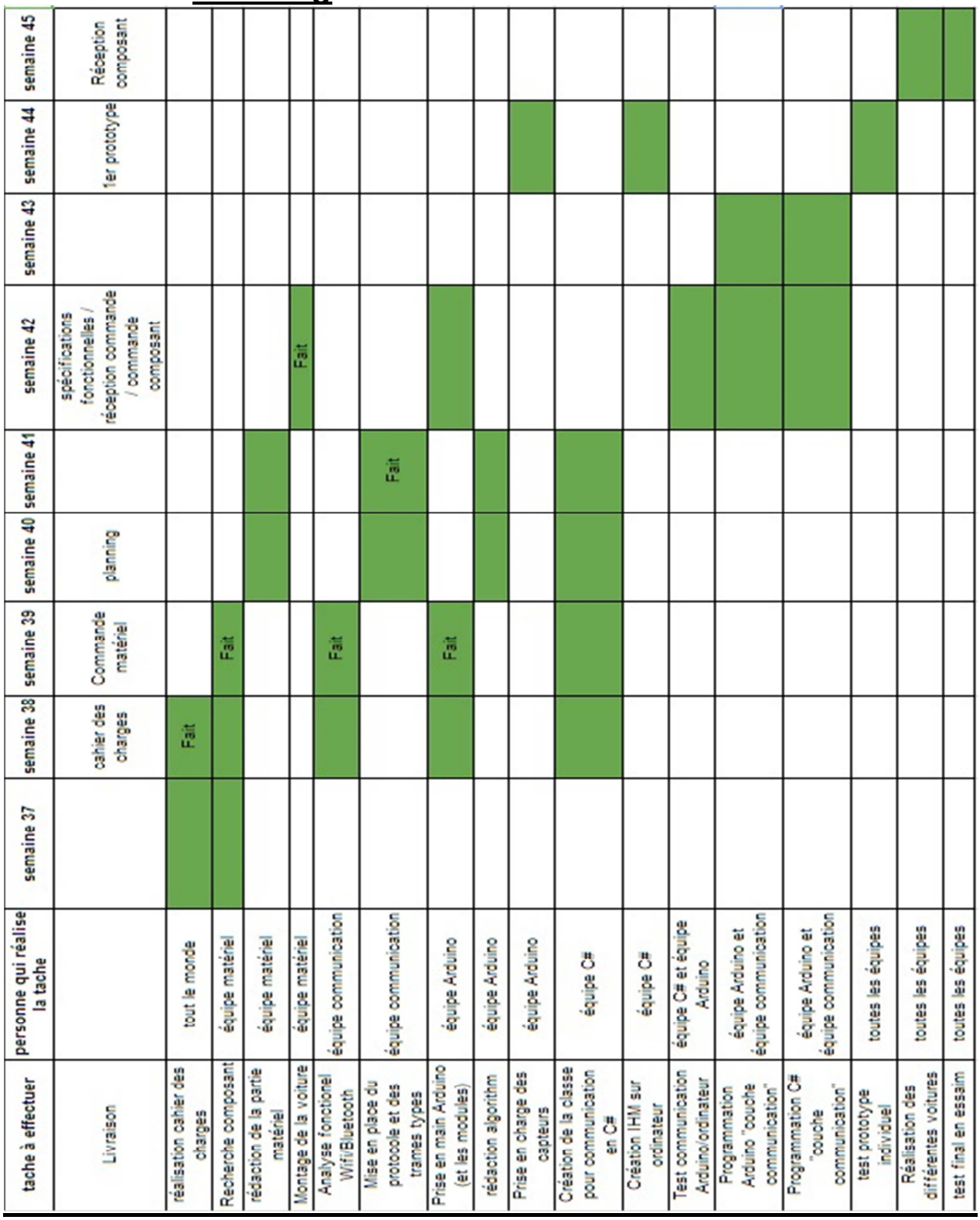
1.6. Compétences requises suite à la demande du

produit

L'équipe devra réunir des compétences dans les domaines suivants :

- Électronique : pour la partie matériel et le choix des composants.
- Programmation : dans divers langages, notamment C/C# pour le développement des applications Arduino et de la partie commande depuis un PC.
- Réseau : des références en réseau et télécommunication sont importantes afin d'étudier rapidement le transfert de données entre le PC et les voitures.

2. Planning



tache à effectuer	personne qui réalise la tache	semaine 37	semaine 38	semaine 39	semaine 40	semaine 41	semaine 42	semaine 43	semaine 44	semaine 45
Livraison			cahier des charges	Commande matériel	planning		spécifications fonctionnelles / réception commande / commande composant		1er prototype	Réception composant
réalisation cahier des charges	tout le monde		Fait							
Recherche composant	équipe matériel			Fait						
rédaction de la partie matériel	équipe matériel									
Montage de la voiture	équipe matériel						Fait			
Analyse fonctionel Wifi/Bluetooth	équipe communication			Fait						
Mise en place du protocole et des trames types	équipe communication					Fait				
Prise en main Arduino (et les modules)	équipe Arduino			Fait						
rédaction algorithm	équipe Arduino									
Prise en charge des capteurs	équipe Arduino									
Création de la classe pour communication en C#	équipe C#									
Création IHM sur ordinateur	équipe C#									
Test communication Arduino/ordinateur	équipe C# et équipe Arduino									
Programmation Arduino "couche commmunication"	équipe Arduino et équipe communication									
Programmation C# "couche communication"	équipe Arduino et équipe communication									
test prototype individuel	toutes les équipes									
Réalisation des différentes voitures	toutes les équipes									
test final en essaim	toutes les équipes									

2.1. Gestion des équipes

Le groupe est séparé en 4 équipes :

- L'équipe dite "hardware", s'occupe du matériel (commande du matériel, compatibilité entre les composants etc.), elle est composée de M. Vien Thomas, et de M. Robineau Baptiste.

- L'équipe dite "communication", gère le protocole de communication entre l'Arduino et l'ordinateur (stratégie à mettre en place lors d'erreur de trame, type de trame etc.), cette dernière est composée de M. Guibert Thomas, et de M. Peyrat Paul-Alexandre.

- L'équipe dite "Arduino", code le programme côté voiture télécommandée, composée de Mlle. Dlaia Khouloud et de M. Tallet-Pinet Paul.

- L'équipe dite C#, crée le programme côté ordinateur (IHM, communication avec les voitures télécommandées), composée de M. Brohard Mathieu.
  En cours de projet, certaines équipes ont évolué. Par exemple, une fois le choix Bluetooth/Wifi effectué, et les types de trames choisies, l'équipe "communication" a été réorganisée, pour aider l'équipe C# et Arduino.
  De plus, les équipes sont autonomes, elles connaissent à l'avance leurs tâches à effectuer pour une durée déterminée. Par moment, certaines équipes ont travaillé en collaboration, par exemple pour tester la communication PC/Arduino.

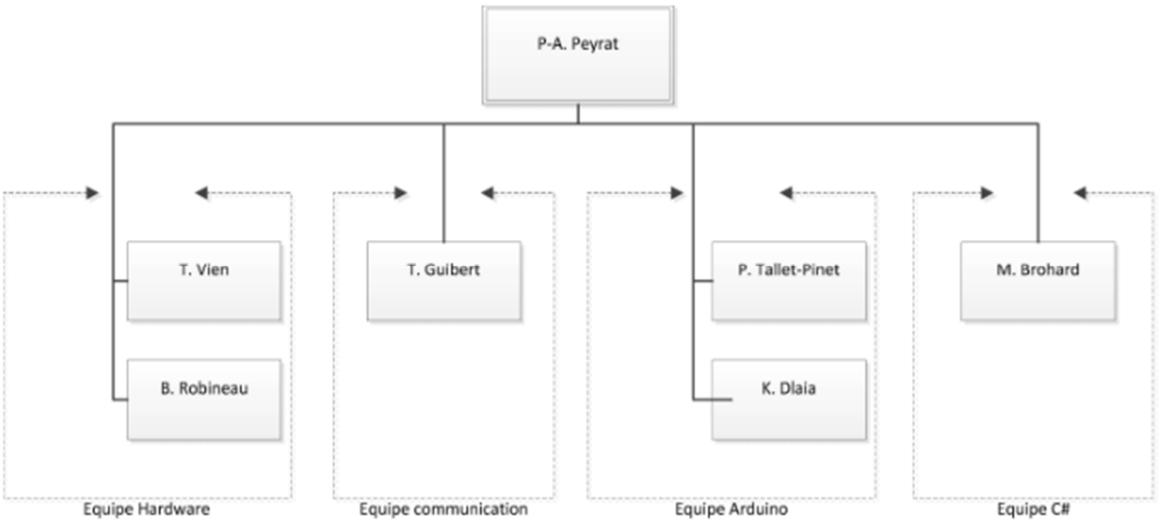


Figure 1 : Organigramme du groupe

3. Choix des composants

Afin de comprendre le déroulement du rapport, nous vous rappelons qu'il faut avoir pris connaissance de l'analyse fonctionnelle avant de commencer cette partie. Celle-ci sert de support au reste des équipes présentes sur le projet. Nous nous sommes concertés afin d'être sûrs que tous les groupes puissent travailler correctement avec le matériel répondant le mieux à leur cahier des charges. Pour répondre aux attentes des coéquipiers ainsi que du client, nous avons découpé notre analyse de façon méthodique en nous posant toujours les mêmes questions :

1. Est-ce que ce composant répond correctement aux règles du cahier des charges ?
    a. Dimension
    b. Prix
    c. Evolutivité

1. Avons-nous une alternative à ce produit ?
    a. Définir les coûts
        i. Temps
        ii. Ressources

3.1. Batterie

Ref: Lipo 2200mAh Battery (Arduino Jack)

Figure 2 : Une batterie LIPO

Li-Po est un élément dans lequel l'électrolyte est un polymère sous forme gélifiée. Elle se rapproche par sa forme et ses caractéristiques des batteries Li-on.

Un élément Li-Po délivre 3,7 V. Livrés en 1, 2, 3, 4, 6 éléments voir plus, ils délivrent donc 3.7V, 7.4V, 11.1V, 14.8V, 22.2V. Souvent équipés d'une prise de charge spéciale, ils peuvent alors être chargés élément par élément à l'aide de chargeurs appropriés. [Source Aerowiki]

Ce type d'accumulateur est très pratique par rapport aux autres accumulateurs existants, puisqu'il n'est pas sensible à l'effet mémoire (attendre la fin de la batterie pour la recharger) et son nombre de recharge est bien plus important que ses concurrents (environ 1000 charges contre 500 pour d'autres).

Spécifications :

- 7.4V (pack de 2 cellules)
- Capacité 2200 mAh
- 1000 cycles de charge
- Jack DC2.1 compatible Arduino
- Taille: 103 x 34 x 15 mm
- Poids: 130 g

On retrouve deux packs de 3.7 V et une capacité 2200mAh. Très apprécié dans notre projet qui nécessite une petite batterie avec beaucoup de "punch". La tension d'alimentation est suffisamment petite pour ne pas risquer de détériorer les circuits de régulation de la carte Arduino et le taux de décharge est suffisamment élevé pour supporter une bonne charge et des moteurs à alimenter.

Répond-elle aux exigences du projet ?

Ses dimensions rentrent complètement dans la recherche d'optimisation de place sur le module ainsi que son poids. En termes de prix, l'accumulateur reste compétitif avec d'autre batterie de même type (environ 23 euros). Nous pouvons donc affirmer que ce choix est judicieux mais un point est à régler puisque nous n'avons aucun moyen de recharger cet accumulateur.

Une alternative possible ?

L'idée d'une alternative semble un risque important puisque ce produit semble presque "fait pour ce projet". Il n'est pas dans notre intérêt de faire nous-mêmes ce genre de composant. Néanmoins nous allons fabriquer un chargeur d'accumulateur adapté à ce composant.

3.2. Montage du module

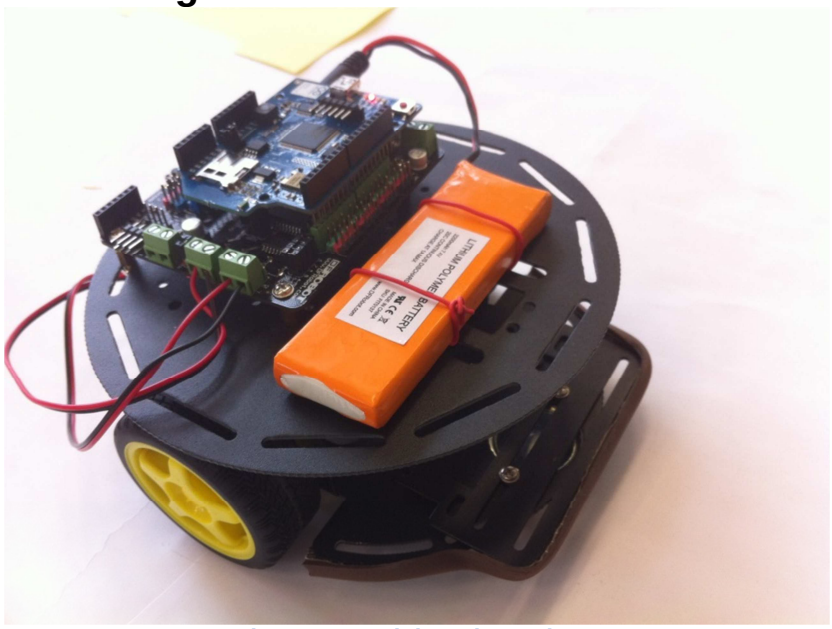


Figure 3 : Module Robot Voiture

Spécifications générales du module :

- 2 moteurs pilotables par Arduino
- Roulette à bille Caster
- Montage complet du châssis sans la partie hardware
- Dimension : 17 cm
- Poids : 400g

Répondent-ils aux exigences du cahier des charges ?

Dans le cahier des charges, un certain nombre de points concernent le module. Dans un premier temps, il était question de la taille, une vingtaine de centimètres environ. Ici, le module fait 17cm de diamètre, ce point est donc respecté. On souhaitait également contrôler les roues de manière indépendante, cela nécessite donc l'utilisation d'un moteur pour chaque roue, cette contrainte est respectée puisque le robot est équipé de deux micros motoréducteurs. Concernant le prix, le coût total du robot rentre dans la limite des 500€ fixé pour l'ensemble.

D'autres modules envisagés ?

Il existe évidemment beaucoup d'autres modules ou robots répondant dans les grandes lignes à nos besoins. Cependant, beaucoup n'avaient pas un prix aussi abordable, ou alors ils étaient plus difficiles à assembler et à piloter. Dans notre cas, la suite de ce projet veut une production en masse du robot, et notre choix s'est avéré être le meilleur pour répondre à cette demande.

3.3. Moteurs

Ref : Mini motoréducteur 6V (FIT0016)

Figure 4 : Motoréducteur

Ce moteur + engrenages, par définition, motoréducteur possède une tension de commande de 6V et une vitesse de rotation de 180 tours/min.

Spécifications :

- Rapport de réduction : 1:120
- Vitesse hors-charge (3V) : 100 tours/min
- Vitesse hors-charge (6V) : 200 tours/min
- Courant hors-charge (3V) : 60mA
- Courant hors-charge (6V) : 71mA
- Courant d'arrêt (3V) : 260mA
- Courant d'arrêt (6V) : 470mA
- Dimension : 55mm x 48.3mm x 23mm
- Poids : 45g

Répondent-ils aux exigences du cahier des charges ?

Vendu en tant qu'accessoire de la plateforme mobile, ces moteurs sont parfaitement adaptés et prennent place correctement sur la plateforme. Le rapport de réduction nous indique un tour de roue pour 120 tours moteur. Cette caractéristique n'a aucun impact pour la suite du projet puisque rien n'est précisé dans la cahier des charges. Nous n'avons cependant aucune information sur la consommation en charge. Ces données seront calculées pendant les tests de la voiture.

D'autres moteurs envisagés ?

Probablement non. Ces moteurs sont spécialement conçus pour la plateforme mobile, il nous semble peu probable de changer de moteurs sauf pour une référence plus puissante.

3.4. Carte Romeo

Ref : Carte Romeo (ATMega328) rev 1.1

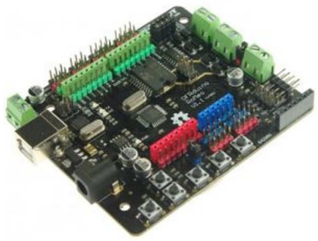


Figure 5 : Carte Romeo pour Arduino

Spécifications :

- Microcontrôleur Atmel ATmega328 à 16 Mhz, 2K RAM, 32K flash.
- 14 Entrées/Sorties numériques dont 6 Sorties MLI (PWM).
- 8 Entrées analogiques utilisables en Entrées/Sorties numériques.
- Interface USB à base d'Atmega8U compatible UNO.
- Alimentation à commutation automatique USB/Externe.
- Connecteur pour programmation directe ICSP.
- Interface série à niveaux TTL.
- Entrée AREF.
- Support des connecteurs à broches mâles et femelles.
- Connecteur pour module radio APC220 ou Bluetooth.
- jeux de connecteurs à broches pour bus I2C.
- Double contrôleur de moteur CC (2A maximum).
- 5 boutons poussoir d'entrées et un bouton poussoir de réinitialisation.
- Alimentation par port USB ou alimentation externe de 7V à 12V CC.
- Sorties d'alimentation en 5V / 3.3V et externe

Répondent-ils aux exigences du cahier des charges ?

Cette carte répond largement aux demandes du cahier des charges. Les entrées et sorties sont en nombre suffisant et l'alimentation suffisamment puissante. Par ailleurs, cette carte est conseillée par le constructeur du module robot pour le développement et le pilotage.

Alternative possible ?

Bien-sûr, mais comme dit précédemment, cette carte répond parfaitement aux différents besoins, et est conseillée par le constructeur.

3.5. Shield Arduino WIFI

Ref : Arduino WiFi Shield

Figure 6 : Shield WIFI Arduino

Cette carte permet de communiquer via une liaison 802.11b/g (WIFI) entre le module Arduino Romeo et un point d'accès. Elle se connecte physiquement sur la carte Romeo. Le principe de fonctionnement ne sera pas explicité dans cette partie, mais dans la partie connexion. Le choix d'une connexion WIFI est une décision commune prise avec les autres équipes et instanciée dans l'analyse fonctionnelle comme fonction contrainte.
Le choix du shield Wifi par rapport à un shield Bluetooth sera expliqué plus tard.

Spécifications :

- Connexion via 802.11b/g
- Signaux TTL 5V
- Encryption personnel de type WPA2 et WEP
- Connexion vers Arduino via port ISP
- Slot SD-Card disponible
- Connexion FTDI pour déboggage du shield WIFI
- Connexion Mini-USB pour mis à jour du firmware

Répond-il correctement aux exigences du cahier des charges ?

En ce qui concerne les dimensions du module, elles sont identiques au module Arduino que nous avons implanté sur la carte Romeo. Le système de carte Arduino se "plug" très simplement les unes sur les autres. Ce qui permet un gain en termes de volume. En revanche, le prix et l'évolutivité du shield laissent à désirer. En effet, l'évolutivité de ce genre de carte est impossible puisque c'est la seule existante pour le moment. Pour le prix, il faut compter 70 euros pour ce shield WIFI et nous avons payé 89 euros. Cette grande différence de prix s'explique par une commande rapide des composants sur un site non officiel pour nous permettre de rapidement commencer le développement du protocole de communication et d'effectuer les tests.

Une alternative possible ?

La question cache en fait deux questions :

- Existe-t-il une alternative technologique à ce produit ?

Oui, il existe une multitude de connexion sans fil, notamment le Bluetooth qui fut l'alternative la plus probable. Ce choix est développé dans la partie connexion.

- Existe-t-il d'autre matériel adapté à ce type d'utilisation ?

Nous avons trouvé un autre composant WIFI qui s'adapte sur la carte Arduino. Il s'agit d'un module wifi pour Arduino disponible sur le site arobose. Son prix est de 90 euros, soit 20 euros de plus que le shield Arduino officiel. De plus, les librairies sont a réaliser puisque ce n'est qu'un produit compatible et non officiel. Cette solution semble pourtant réalisable et nous estimons que la refonte des bibliothèques occuperait une ressource pendant 3 jours à raison de 5 heures par jour.

3.6. Boutons poussoirs

Ref : D6C10LFS

Figure 7 : Bouton poussoir

Ces boutons poussoirs nous servent de capteur de collision. Les BP sont installés de façon à ce que le module appui dessus lorsqu'il rencontre un obstacle. De ce fait, une interruption sera déclenchée au niveau du microcontrôleur.

Spécifications :

- Couleur: Gris
- Courant de contact: 100mA
- Courant de contact c.c. max.: 100mA
- Diamètre: 12mm
- Durée de vie, mécanique: 250000
- Largeur (externe): 11.4mm
- Longueur, course max..: 0.8mm
- Longueur/hauteur: 11.5mm

Répondent-ils aux exigences du cahier des charges ?

Oui. Nous ne souhaitons pas un système complexe mais seulement un système qui déclenche une interruption après actionnement. Le bouton poussoir remplit parfaitement ce rôle aussi bien en matière de coût que d'intégration.

D'autres possibilités envisageables ?

Il existe bien entendu une multitude de possibilités afin d'informer le système qu'il se trouve dans un obstacle. Nous avons choisi cette technique pour des raisons de coût et aussi par expérience. Nous connaissons déjà ce matériel et l'avons déjà utilisé de la même manière.

Remarque : Pour éviter de détériorer les BP, nous avons ajouté des boudins de protection pour porte fenêtre.

Nomenclature :

Désignation	Quantité	Prix €
Carte Romeo	1	31,97
Plateforme Robot mobile	1	34,97
Shield Wifi Arduino	2	179,74
Batterie LIPO	1	22,97
Bouton poussoir	2	4,45
Boudin de protection	1	4,05
TOTAL		278,15

4. Communication voiture – ordinateur

Cette partie est réalisée par l'équipe "communication", dans un premier temps, elle a pour but de choisir le support pour la communication entre l'ordinateur et la voiture télécommandée et les protocoles entre ces deux derniers.

4.1. Choix du support

a. Le Bluetooth

Le Bluetooth utilise un protocole "maitre-esclave". Deux modes sont possibles :

- Le mode "actif" (7 périphériques au maximum): chaque périphérique a une adresse unique sur le réseau
- Le mode "packed" : il n'y a pas d'adresse pour un périphérique, lorsqu'une trame est envoyée sur le réseau, tous les périphériques la reçoivent

Avantages du Bluetooth:

- Consomme moins que le Wifi
- Peu encombrant (cet avantage n'est pas pris en compte avec un shield Arduino)
- Meilleur débit
- Module moins chère que le Wifi
- Le Bluetooth permet le mode "parked", qui signifie qu'une trame peut être envoyée à tout le monde en même temps (ce qui conviendrait pour la conduite en essaim).

b. Le Wifi

Contrairement au Bluetooth, le wifi est une connexion point à point.

Avantages du Wifi:

- Meilleure portée.
- Facilité de mise en place (bibliothèque d'utilisation native en C#).

c. Tableau de comparaison débit/portée

	Wifi (norme 802.11g)	**Bluetooth**
Débit	54Mbits/sec	100Mbit/sec
Portée	100m	10m à 20m (pour 2.5mW: module Arduino)

4.2. Conclusion

La technologie Wifi sera préférable au Bluetooth. Premièrement, il semblerait que le Bluetooth ne soit pas natif en C# (langage choisi pour la programmation). Les exemples trouvés sur Internet semblent soumis au matériel Bluetooth utilisé, il y a donc des risques d'incompatibilité. Ensuite, aucun module Arduino Bluetooth n'a été trouvé parmi les fournisseurs officiels (le module Xbee serait vendu au profit de ce dernier). Par contre, la technologie Wifi serait plus simple à utiliser en C#. De plus nous avons expérimenté un module Wifi dans un autre projet.

Et enfin, le Wifi utilise une connexion point à point, ainsi il sera possible de faire communiquer deux voitures entre elles, utiliser un système de triangulation pour retrouver une voiture ou alors utiliser une voiture comme relais pour augmenter la distance entre l'essaim et l'ordinateur.

5. Fonctionnement et protocole

La communication, entre l'application cotée ordinateur et celui des voitures, se fera par Wifi. Le protocole utilisé sera TCP/IP. Ce protocole a été choisi car il est simple à mettre en place, et il permettra de contrôler individuellement une voiture lorsque plusieurs seront connectées à l'ordinateur, contrairement à l'UDP qui lui, envoie à tout le monde les données.

5.1. Fonctionnement en régime permanent

Toutes les voitures seront connectées sur l'ordinateur, par contre elles seront connectées chacune sur un port différent. Lorsque l'application sur l'ordinateur ou la voiture envoie une donnée, cela se déroule de la même manière :

- L'émetteur envoie une donnée (le premier octet correspondant au type de donnée, le reste étant la donnée en elle-même)
- Le destinataire traite la donnée, et si les données sont correctes, il renvoie le code erreur "0", signifiant qu'aucune erreur n'a été détectée.

5.2. Protocole de connexion

Le but de ce protocole est d'avoir une connexion entre l'ordinateur et la voiture.
Contraintes :

- Les paramètres d'initialisation sont identiques entre toutes les voitures.
- Le protocole TCP oblige à se connecter à un port différent pour chaque voiture.

La solution retenue se déroule en 2 temps :

- Dans le premier temps, la voiture se connecte à un port fixe (commun à toutes les voitures) sur l'ordinateur. L'ordinateur récupère un port disponible. Une fois le port trouvé, il ouvre un serveur TCP qui écoute sur ce port, en même temps il envoie à la voiture le numéro de ce dernier.

- Dans un deuxième temps, la voiture se déconnecte du premier port pour se reconnecter au numéro de port reçu.

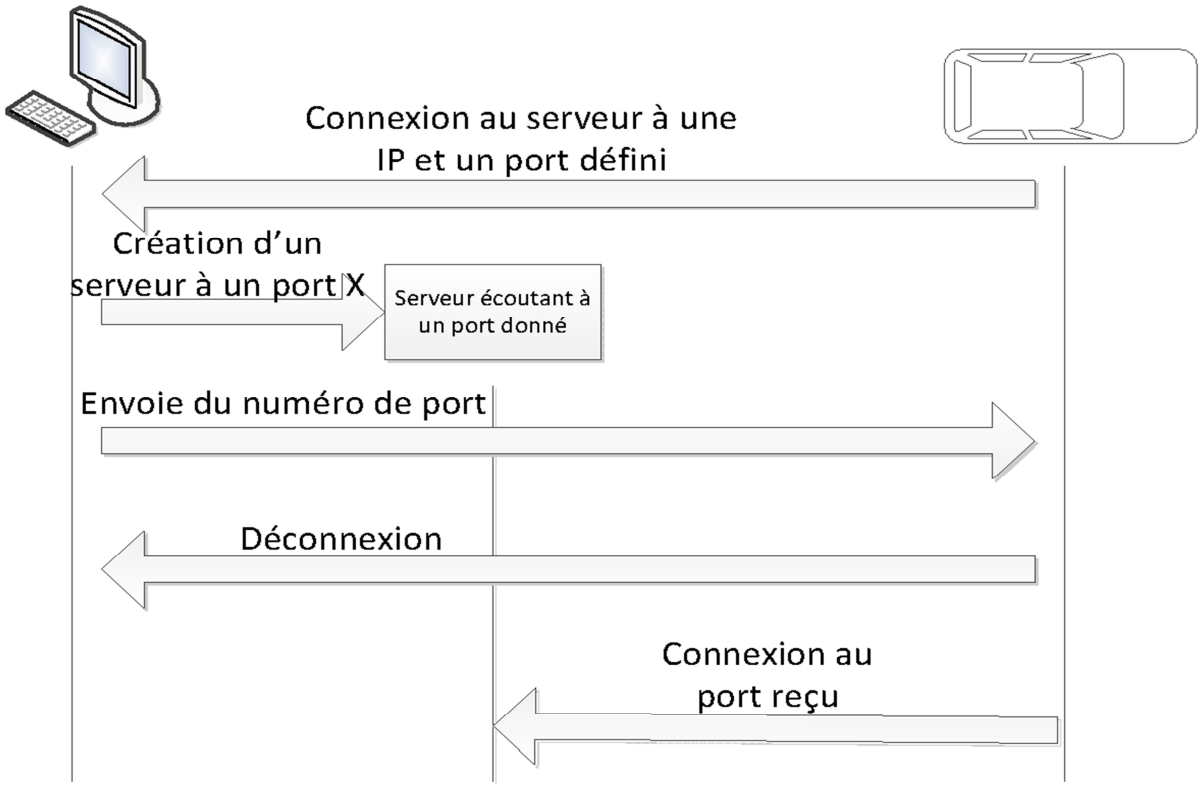


Figure 8 : Protocole de connexion

Grâce à cela, les voitures peuvent avoir les mêmes programmes sans qu'il n'y ait des problèmes de connexion.

Les voitures communiquent avec l'ordinateur grâce aux serveurs créés. Lorsqu'une voiture se déconnecte d'un serveur créé, le serveur est détruit. Le port défini est 8000. L'adresse IP de l'ordinateur est: 192.168.137.201

5.3. Cas particuliers

Pour rappel, les codes d'erreur permettent de générer des acquittements mais également à communiquer des erreurs telles qu'une donnée incompréhensible.
Les cas ci-dessous expliquent la procédure à faire lors de la réception de code d'erreur.

Acquittement non reçu

Si un acquittement n'a pas été reçu dans les 1000 millisecondes, lors de l'envoi d'une donnée, l'émetteur renvoie la donnée, si malgré cela l'acquittement n'a toujours pas été reçu, l'émetteur considère qu'il n'est plus connecté à son destinataire. Dans ce cas, la procédure suivante s'applique:

- Point de vue ordinateur : la voiture a été déconnectée ou est hors de portée. Dans ce cas l'utilisateur est prévenu par un message, le message doit être assez discret pour qu'il ne gène pas l'utilisateur (le message ne doit pas bloquer l'application).

L'application supprime également le serveur d'écoute au port où la voiture était connectée.

- Point de vue de la voiture: la voiture est hors de portée ou l'application sur l'ordinateur a été arrêtée. Dans ce cas, la voiture relance la procédure de reconnexion au port 8000 (port de connexion par défaut), si elle ne le prévient pas, elle retentera la connexion toutes les 10 secondes.

Code d'erreur: "1" type de donnée inconnue

Si ce code d'erreur a été reçu :
Dans ce cas, le destinataire de ce code tente de renvoyer une seconde fois ses données, si malgré cela, il reçoit toujours le code erreur "1", le protocole passe à une seconde phase :
Si le code d'erreur est reçu coté voiture: la voiture se déconnecte et se reconnecte au port 8000 (port par défaut)
Si le code d'erreur est reçu, coté ordinateur : l'ordinateur signale à l'utilisateur que la voiture a un problème, par un message d'erreur, ne bloquant pas le programme.

Code d'erreur: "2" donnée incorrect

Si ce code d'erreur a été reçu :
L'émetteur renvoie la donnée, en la reconstruisant de A à Z. Si suite à cela, le code d'erreur "2" est toujours retourné, la donnée est tout simplement détruite.

Code d'erreur: "3" commande impossible

Ce code d'erreur n'est reçu que par l'ordinateur, puisqu'il n'est envoyé que par la voiture. Si ce code d'erreur est reçu, l'ordinateur prévient, grâce à un signal, l'utilisateur que sa commande n'a pas pu être exécutée (ce signal ne doit pas bloquer l'application). Par exemple, il y a un obstacle devant la voiture, et l'utilisateur lui demande d'avancer, la voiture retournera le code d'erreur "3".

5.4. Remarques

Les stratégies misent en place pour les codes erreur n'ont pas été mises en place, ainsi que le système du "time out".
Les codes d'erreur n'ont pu être implémentés faute de temps.
Le système des "times out" ne peut être implémenté avec un système d'interruption timer, pour plus d'explication, voir le chapitre "problèmes rencontrés"

5.5. Construction des trames

Pour transmettre et recevoir les informations, un protocole a été mis en place pour traiter, facilement, différents types d'informations. Les données seront séparées en 2 parties:

- Le premier octet représentera le type de données.
- Le reste représentera les données utiles

a. Différents types de trame venant des robots :

Liste des types de trames:

- Code d'erreur
- État capteur

Code d'erreur :

Lorsque la voiture aura une erreur telle qu'une trame reçue incompréhensible, elle utilisera le code d'erreur pour parler avec le serveur.
Premier octet : 255
Donnée: type de l'erreur (1 octet)

Etat capteur:

Il s'agit de l'état des capteurs
Premier octet : 1
Donnée : valeur des capteurs. Puisque les capteurs sont en tout ou rien, chaque bit de l'octet représentera l'état d'un capteur.

Bit	7	6	5	4	3	2	1	0
Etat capteur	X	X	X	X	X	X	capteur avant	capteur arrière

b. Différents types de trame venant de l'ordinateur :

Liste des types de trames:

- Code d'erreur
- Commande moteur
- Initialisation
-

Code d'erreur :

Nombre d'octet : 2
Lorsque l'ordinateur aura une erreur telle qu'une trame reçue incompréhensible, il utilisera le code d'erreur pour parler avec le serveur.
Premier octet : 255
Donnée: type d'erreur (1 octet)

Commande moteur :

Nombre d'octet : 3
Premier octet : 1
Donnée: la data contient 2 octets, le premier octet correspond à la vitesse, les 7 bits de poids faibles représenteront la vitesse (de 0 à 127, qui correspondra à 0% jusqu'à 100% de la vitesse max des moteurs). Le bit de poids fort correspondra lui, au signe de la vitesse. Quand la vitesse est négative (bit 7 à 1), la voiture ira en arrière.

bit	7	6	5	4	3	2	1	0
Vitesse	bit de signe	bit 6 de vitesse	bit 5 de vitesse	bit 4 de vitesse	bit 3 de vitesse	bit 2 de vitesse	bit 1 de vitesse	bit 0 de vitesse
Direction	X	X	X	X	X	X	bit 1 direction	bit 0 direction

Les 7 bits de vitesse : à 0 la vitesse sera à 0% et à 127 la vitesse sera à 100%.
Lorsque le dernier bit de vitesse sera à 1, la vitesse sera négative.

Les 2 bits de direction :
(00)b : Tout droit
(01)b : A droite
(10)b : A gauche

Initialisation :

Nombre d'octet : 2
Premier octet : 0
Donnée : La donnée envoyée correspondra au numéro de la voiture dans l'essaim. Ce numéro sera également utilisé pour que la voiture connaisse le port de connexion du nouveau serveur sur lequel il est associé (voir protocole de connexion).
Le nouveau numéro de port utilisé par la voiture sera: 8000 + le numéro reçu.
Valeur de la data: [1-255]

6. Programme

6.1. Logiciel utilisateur

Le logiciel utilisateur permet au travers d'une interface graphique intuitive, de visualiser et de piloter un ou plusieurs robot(s) connecté(s). Ce logiciel utilisateur est structuré en deux couches logicielles bien distinctes :

- La couche présentation qui constitue l'interface graphique,
- La couche métier, partie fonctionnelle de l'application, elle permet de réaliser des opérations logiques commandées par la couche présentation.

6.2. Couche Présentation

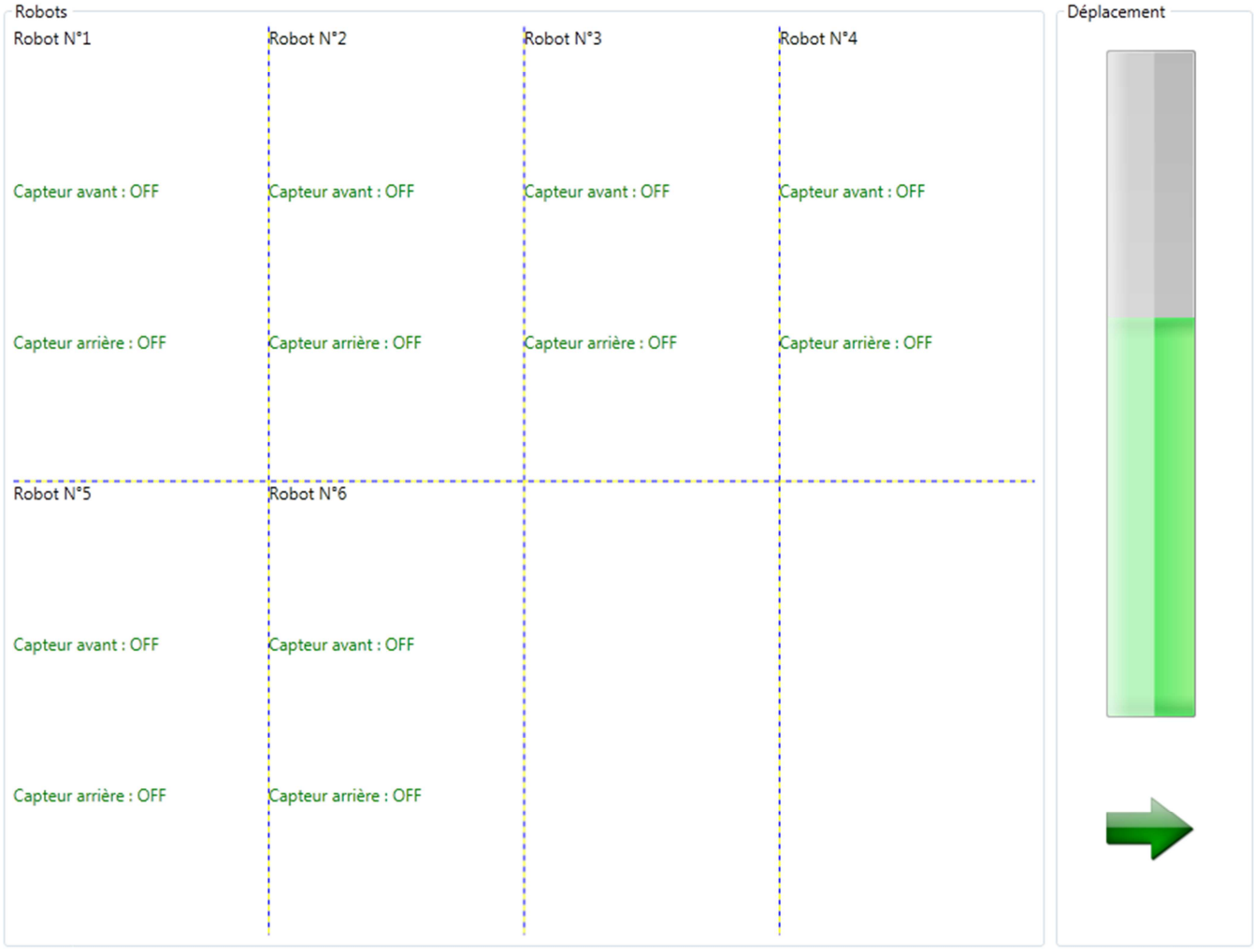


Figure 9 : Interface couche présentation

La couche présentation a pour but d'indiquer à l'utilisateur toutes les informations du ou des robots. Elle est développée en WPF pour que la programmation visuelle soit

plus simple. Nous ne connaissions pas ce langage au début, mais il est une dérivation du C#, il a donc été très simple pour nous de le mettre en pratique. Cette couche se compose en deux parties distinctes.

Dans la première partie, il y a les informations relatives aux robots tels que le nombre de robots connectées, les numéros de ports utilisés et les états des capteurs. Cette partie est indépendante de l'utilisateur, lorsqu'un robot se connecte, il est automatiquement ajouté à la liste des robots et lorsqu'un robot se déconnecte, il est retiré de la liste.

Dans la seconde partie, est indiqué dans quelles directions sont dirigés les robots. En effet, c'est uniquement l'utilisateur qui intervient pour le fonctionnement. Il utilisera les flèches directionnelles du clavier pour orienter les robots. Une barre de progression affichera à quelle puissance sont les moteur, et une flèche indiquera si les robots vont : tout droit, tournent ou sont à l'arrêt. Si la flèche est verte, les robots avancent, par contre si la flèche est rouge, les robots reculent.

6.3. Couche Métier

Au sein de notre application, la couche métier permet de gérer la communication wifi entre le robot et le logiciel utilisateur suivant un mode de communication client/serveur. Afin de structurer au mieux le code de la couche métier, nous avons utilisé les concepts de programmation orientée objets. Cela nous a permis de créer différents modules fonctionnels indépendants et facilement modifiables grâce à l'encapsulation des données et des méthodes au sein de classes instanciables.

Au commencement de notre projet, nous avions fait le choix d'une communication TCP synchrone que nous avons abandonné au profit d'une liaison TCP asynchrone afin de ne pas être contraint par des tâches bloquantes.

Ainsi, nous avons créé une classe "ServeurTCP" utilisant le protocole TCP, qui réalise la connexion/déconnection asynchrone d'un canal de communication entre un robot et le logiciel et également l'envoi et la réception de trame. Une seconde classe "Robot", modélise le comportement logique d'un robot et permet d'obtenir une image de celui-ci au sein de notre application.

Au sein de la classe "Robot" nous avons créé une instance de la classe "ServeurTCP" en tant que propriété, ce qui permet à la classe "Robot" de gérer la communication asynchrone entre le robot et le logiciel utilisateur.

La classe "ServeurTCP" étant exclusivement dédiée à l'envoi et à la réception de buffers de données brutes (tableau de Byte), il est nécessaire que l'interprétation de ces buffers, au travers du protocole de communication choisie, soit réalisé au sein de la classe "Robot".

Nous avons fait le choix de rendre indépendantes les fonctionnalités de gestion, et d'interprétation des buffers car cela permet d'obtenir, du point de vue de la classe "Robot", une abstraction concernant le moyen de communication utilisé.

En effet, si on prend en compte les évolutions éventuelles de notre application, et que l'on envisage la prise en charge d'une communication Bluetooth par exemple, il suffira de créer une classe "Bluetooth" contenant les mêmes méthodes que la classe "ServeurTCP" pour assurer la compatibilité du code.

Idéalement si notre application est voué à évolué dans ce sens, il serait judicieux de créer une interface "Icommunication", assurant un contrat de méthode entre les éventuelles classes de communications (Wifi, Bluetooth, Radio, etc) et la classe "Robot".

Suivant le cahier des charges, notre application doit pouvoir géré plusieurs robots simultanés, or la classe "Robot" ne permet pas cela. Pour ce faire, nous avons crée plusieurs instance de la classe "Robot", ainsi on crée un canal de communication propre à chaque connections robot/logiciel utilisateur. N'ayant à notre disposition qu'un unique robot physique, nous avons crée un simulateur logiciel qui reproduit le comportement d'un robot, afin de validé le fonctionnement multi-connections.

6.4. Programme Arduino

Le programme Arduino est conçu afin que l'utilisation du robot soit la plus simple possible. En effet, l'utilisation de celui-ci ne requiert aucun réglage, il est parfaitement autonome. Il suffit de brancher la batterie pour assurer la connexion du robot au serveur.

Le programme Arduino est décomposé en deux parties :

- Une partie **Setup** : qui sert à initialiser la connexion au réseau et au serveur.
- Une partie **Loop** : qui s'exécute en permanence et qui permet la gestion du robot dès réception de trames et leur envoi en cas de contact d'un capteur.

Dans un premier temps nous voulions utiliser les interruptions pour agir lorsqu'un capteur change d'état mais il s'est avéré qu'il n'était pas possible d'envoyer une trame à l'ordinateur lors d'une interruption, car d'après les stratégies mises en place, lorsque la voiture rencontre un obstacle, elle devait s'arrêter et renvoyer à l'ordinateur l'état de ses capteurs. Nous avons donc choisi de vérifier l'état des capteurs et d'envoyer les trames dans la fonction Loop.

Il a été décidé aussi que l'on fasse appel à une bibliothèque que nous avons créée (Robot.h) afin de simplifier l'écriture du programme principale. Cette bibliothèque regroupe les principales fonctions de gestion du robot :

- Fonction de connexion au serveur.
- Fonctions de déplacements du robot pour chaque direction.

- Fonctions de lecture des trames reçues (en provenance du serveur).
- Fonctions d'envoi de trames au serveur (acquittement états des capteurs).

Robot.h regroupe aussi tous les "defines" qui correspondent aux différents paramètres du programme.

Finalement, le diagramme suivant illustre le comportement du programme Arduino

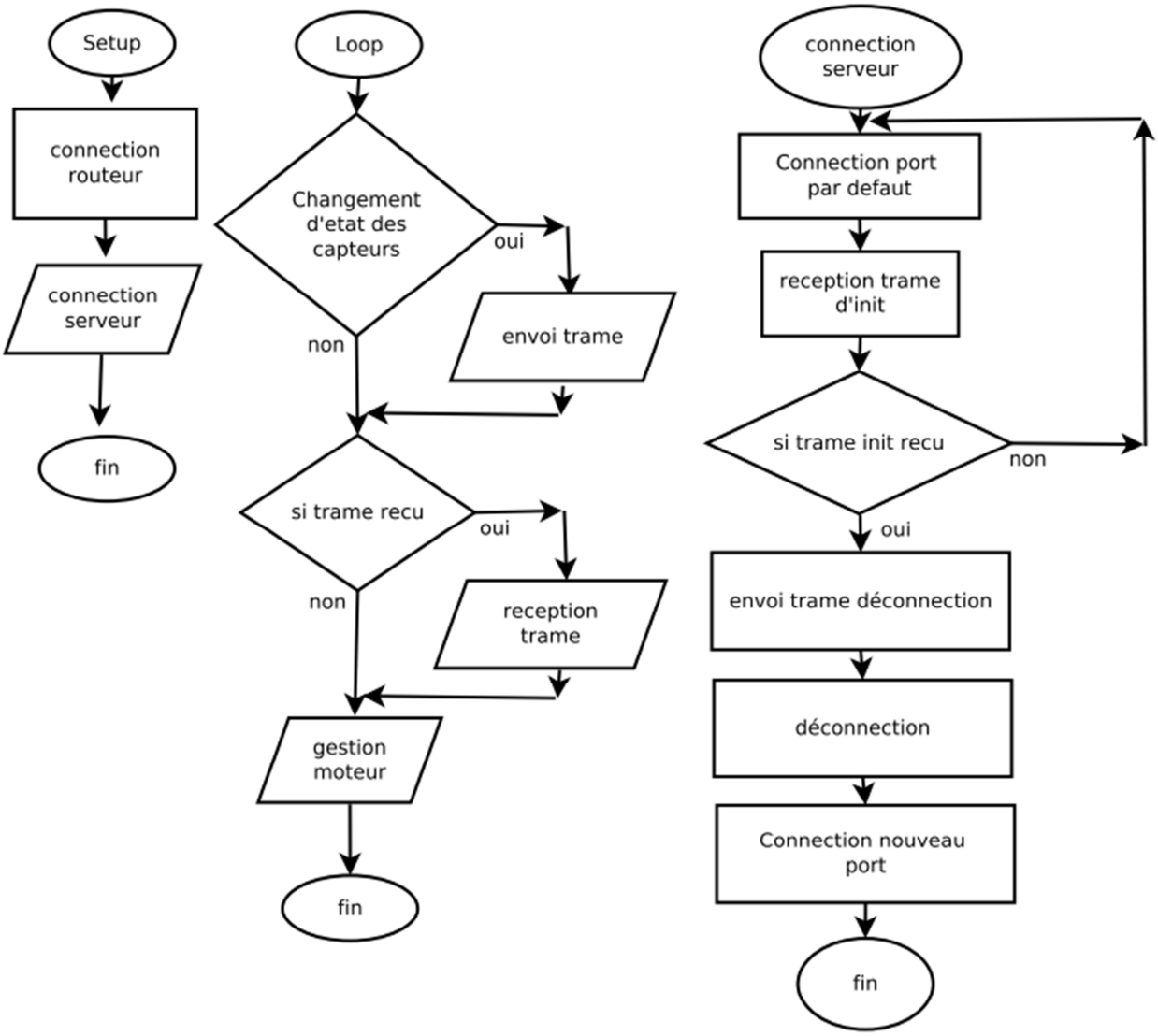


Figure 10 : Diagramme du comportement Arduino

7. Problèmes rencontrés

Un des problèmes majeurs que nous avons rencontré et qui nous a fortement pénalisé dans l'avancement de notre projet est la gestion de reconnexion des robots sur le même point de terminaison local (liaison adresse IP, port). En effet, la gestion de la communication robot/logiciel utilisateur est réalisé de manière asynchrone ce qui provoque des problèmes de synchronisations. De plus, le système d'exploitation ne permet pas de réutilisé immédiatement un canal de communication précédemment libéré, par conséquent nous n'avons pas trouvé le moyen de contourné cette restriction imposé pour reconnecter les robots après déconnexions.

Le seul moyen de reconnecté un robot est de redémarré le logiciel utilisateur. Lors de nos test pour utiliser les interruptions de l'Arduino nous nous somment rendu compte que le shield wifi utilisait beaucoup de pin pour communiquer avec la carte Arduino, Comme par exemple les pins qui correspondes aux interruptions.

Le shield wifi est un shield compatible Arduino. Nous avons été déçu de son manque de fonctionnalité en début de projet, car il ne gère pas les connexions au réseau ad 'hoc. Ce qui nous a obligés à créer notre propre réseau Wifi à l'aide d'un logiciel tiers.
De plus nous avons trouvés que le shield manquait de fiabilité (connexion et déconnexion aléatoire).

Le châssis, la carte Roméo et le shield Wifi ne peuvent être commandés ensemble que sur le site Arobose, malheureusement, il ne s'agissait pas d'un fournisseur officiel. Ceci nous a ralentis lors des commandes des composants.

Conclusion

Pour conclure, nous pouvons dire, que ce projet nous a permis d'améliorer la gestion d'un travail en équipe, comme la confrontation de solutions techniques pour un même problème. De plus, nous avons pu nous améliorer en communication réseau, et en programmation (C# avec une gestion en WPF et Arduino).

Au niveau du projet, le cahier des charges est en partie réalisé. Nous avons perdu beaucoup de temps, notamment sur la partie commande ainsi qu'avec le Shield Arduino peu fiable. Du aux problèmes de commande, nous n'avons pu tester la commande de plusieurs voitures (par contre des tests ont été effectué grâce à un simulateur). En revanche, nous avons réalisé le principal objectif, qui était de construire une voiture télécommandée simple d'utilisation, ainsi que l'IHM pour contrôler celle-ci.

Bibliographie

- Choix du matériel :

Arobose,"Site internet Arobose", http://www.arobose.com/shop/, année NC.

- Analyse fonctionnelle :

LP AULNOYE,"COURS ANALYSE FONCTIONELLE", http://noel.wifeo.com/documents/Cours---Analyse-fonctionnelle.pdf, année NC.

- Batterie :

Auteur incconu, "B ATTERIE LIPO 7.4 V", http://boutique.3sigma.fr/14-batterie-lipo-74-v-2200-mah-connecteur-jack-arduino.html, année NC.

Auteur libre, "BATTERIE LIPO", http://aerowiki.free.fr/wiki/index.php/Batterie_Li-Po, année 2010.

- Communication :

Droit d'auteur libre, "BLUETOOTH",http://fr.wikipedia.org/wiki/Bluetooth, Année 2013.

- Arduino :

Arduino,"SITE INTERNET OFFICIEL ARDUINO", http://www.arduino.cc/, année 2013.

Annexes

Analyse Fonctionnelle

Document technique Arduino

Document technique Application

Guide utilisateur

Code source Arduino

Code source Application

Application

Développement d'une voiture télécommandée à base d'Arduino

Spécialité Informatique Industrielle
4ème année
2012 – 2013

Rapport de Système et Réseau

Résumé : Ce dossier comprend toutes les informations relatives au projet Développement d'une voiture télécommandée à base d'Arduino

Mots clefs : Arduino – Voiture télécommandée – DII – Polytech