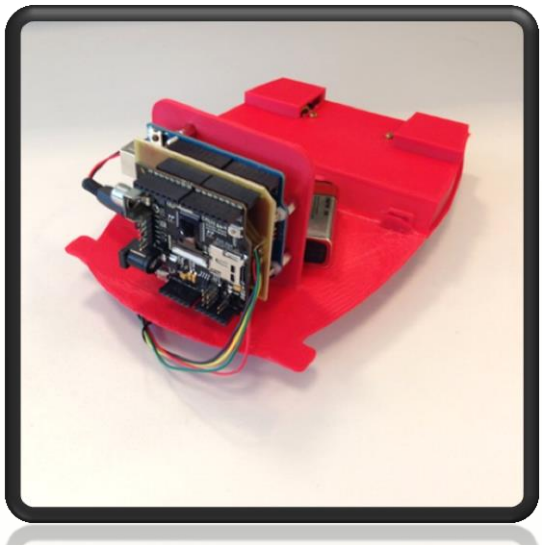
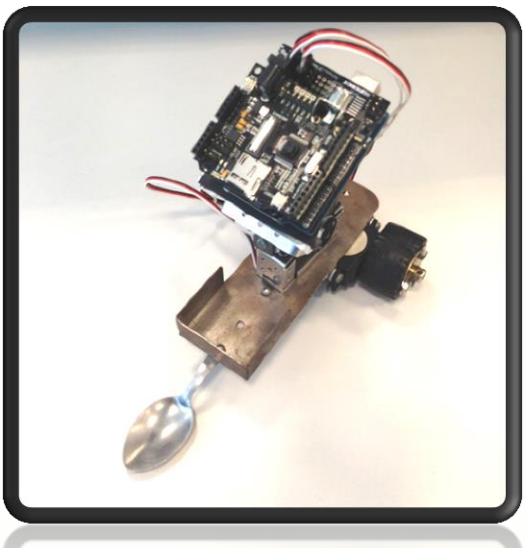
Projet de Physique P6

STPI/P6/2014 – 50

Robot commandé par une CMU CAM 4





Etudiants :

Marine BERTRAND **Sarah DEMERSEMAN**

Sisly FAILLOT **Lucie LEGROS**

Madeleine ZUBER

Enseignant-responsable du projet :

Ludovic HENRIET

Date de remise du rapport : ***16/06/2014***

Référence du projet : ***STPI/P6/2014 – 50***

Intitulé du projet : ***Robot commandé par une CMU CAM 4***

Type de projet : ***Expérimental***

Objectifs du projet :

Notre projet consiste en la fabrication d'un robot équipé d'une CMUcam4 qui exécuterait des commandes prédéfinies. Pour ce faire, le robot devra être construit et programmé. Notre objectif principal est que notre robot fonctionne, c'est-à-dire qu'il n'y ait pas d'erreurs de construction ni de programmation, et qu'il exécute correctement nos commandes.

Mots-clefs du projet : ***CMUCam4 ; Robot ; Arduino***

TABLE DES MATIERES

NOTATIONS, ACRONYMES

- CMUCam4 : Système présentant une caméra et un microprocesseur intégré qui dispose d'un port série permettant de créer une interface avec une machine. C'est la caméra que nous avons utilisée pour détecter les couleurs lors de notre projet.

- Arduino : Circuit imprimé basé sur une interface entrée/sortie simple. C'est aussi le nom du logiciel libre utilisé lors de notre programmation.

- RVB : Abréviation de Rouge Vert Bleu (RGB en anglais). Ce sont les trois couleurs primaires en synthèse additive.

1. INTRODUCTION

1.1. Présentation du projet

Nous avons choisi de nous intéresser au module de reconnaissance vidéo CMUcam4 qui fonctionne à l'aide de la carte Arduino dans le cadre de notre projet de P6 du 4ème semestre. En effet, nous étions curieuses de savoir comment fonctionnent ces éléments qui nous étaient inconnus et ce sujet nous paraissait original, en comparaison avec les autres proposés.

Nous avions pour objectif, et cela a été défini très clairement dès la première séance avec le professeur encadrant, de réaliser un robot qui, en fonction de la couleur qu'il voit, exécute une commande précise. Pour cela, le projet devait se faire en plusieurs étapes, chacune d'elle ayant pour but de nous faire comprendre le fonctionnement et les caractéristiques des différents composants qui allaient servir dans le projet final: d'abord le langage Arduino, puis la carte Arduino qui utilise ce langage, puis la CMUcam4 qui fonctionne lorsqu'elle est connectée à la carte et chargée avec le programme et finalement le robot avec ses 2 moteurs.

Après 14 séances de travaux pratiques, nous avons atteint nos objectifs : notre robot réagit à trois couleurs différentes, les moteurs fonctionnent et nous pouvons donner des ordres au robot afin qu'il suive les couleurs qu'on lui indique.

We have chosen to focus on recognition the CMUcam4 video module which works using the Arduino board as part of our project of P6 during the 4th semester. In fact, we were curious to discover how these elements unknown to us works and this subject seemed original compared to the other proposed.

It was clearly defined at the first meeting with the teacher supervising that we aim to make a robot that, depending on the color he sees, performs a specific order. For this,

the project should be done in several steps, each of which aimed to make us understand the operation and characteristics of the various components that were used in the final project: first the Arduino and the Arduino board using the good language, then CMUcam4 which works when it's connected to the card and loaded with the program and finally the robot with 2 motors.

After 14 practice sessions, we have achieved our goals: our robot reacts to three different colors, motors operate and we can give orders to the robot so that it follows the color we told it (red).

1.2. Contexte de travail

Pour travailler sur ce projet, nous nous retrouvions chaque mardi pour une séance d'1h30 dans un laboratoire robotique. Là, nous avions accès aux ordinateurs pour compiler nos programmes, compléter nos recherches ou finaliser notre rapport ainsi qu'au matériel nécessaire pour nos trois expériences. Durant les deux premières séances, nous nous sommes familiarisés avec le sujet en faisant des recherches sur internet ou en lisant des documents que le professeur avait mis à notre disposition pour nous aider. Ensuite, nous avons commencé nos expériences en se répartissant les tâches à chaque étape.

To work on this project, we met every Tuesday for a 1:30 session in a robotics laboratory. We had access to computers to compile our programs, complete our researches or finalize our report and the equipment needed for our three experiments. During the first two sessions, we have become familiar with the subject by searching on the internet or reading documents that the teacher had at our disposal to help us. Then we started our experiments by dividing the tasks at each stage.

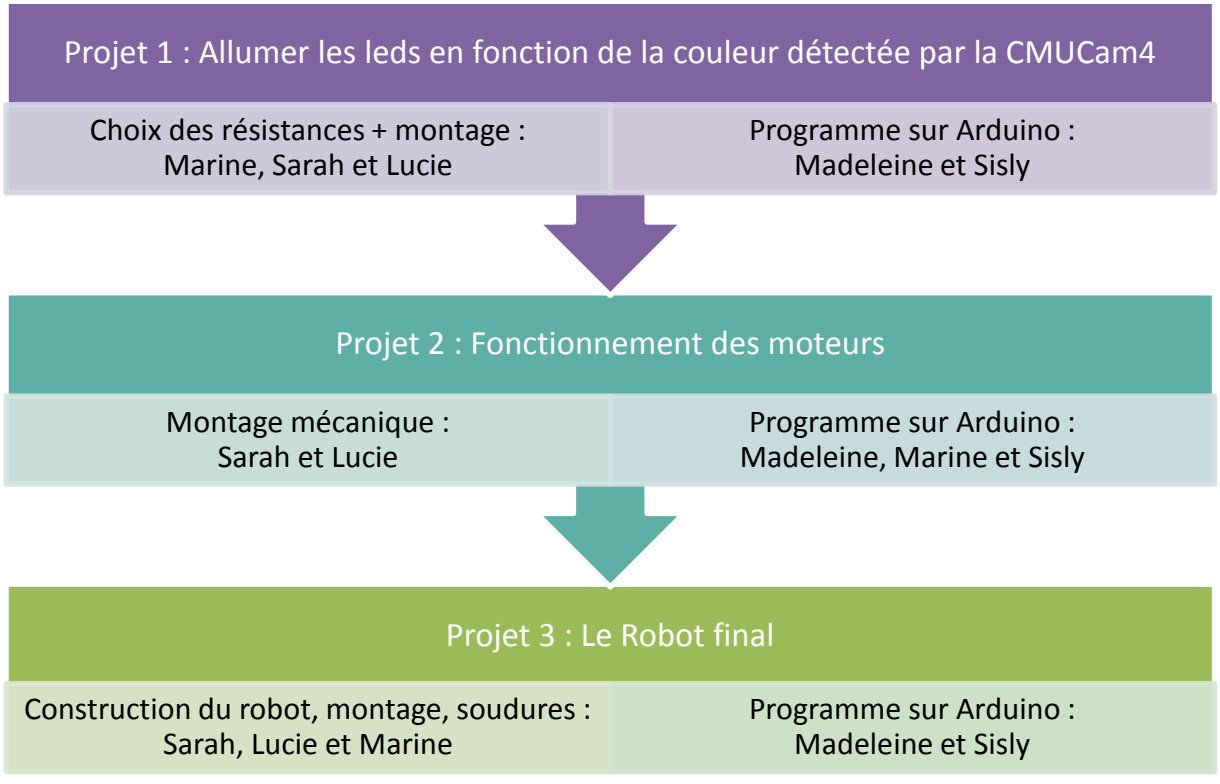
1.3. Cahier des charges

Notre robot sera équipé de deux roues mises en rotation par des moteurs, et fixées sur un support. Il sera constitué d'une carte Arduino et d'une caméra, fixés sur une «tourelle», qui pourra être déplacée dans l'espace.

Notre Robot suiveur devra avoir les fonctionnalités suivantes :

- Se déplacer vers la gauche et la droite grâce aux deux roues.
- Déplacer la tourelle du haut vers le bas, ou de gauche à droite.
- Capturer une image de son environnement plusieurs fois par seconde, avec un intervalle de temps déterminé.
- Reconnaître la couleur ROUGE sur cette image, que l'on pourra définir plus ou moins finement avec le code RVB.
- Identifier la zone dans laquelle il détecte du rouge.
- Se déplacer de façon à suivre la couleur rouge et à s'en approcher : on pourra soit simplement suivre la couleur rouge en déplaçant la tourelle, soit tourner toute la structure du robot grâce aux deux roues.

2. METHODOLOGIE / ORGANISATION DU TRAVAIL

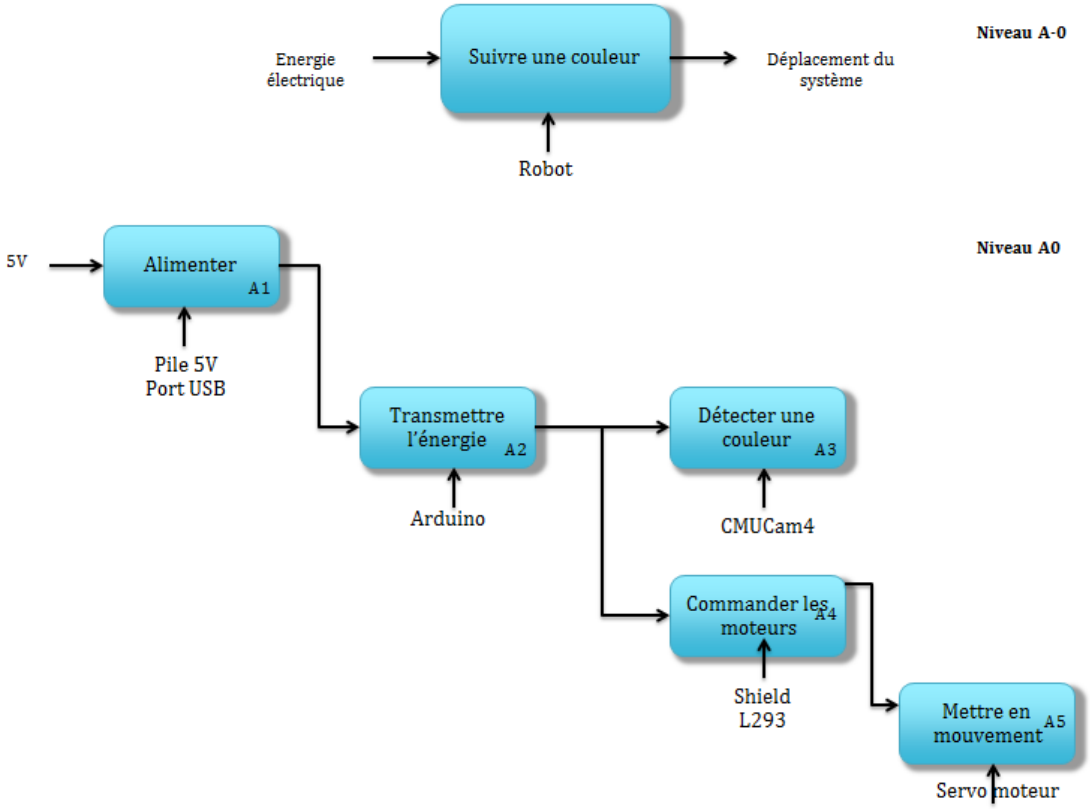


CALENDRIER

04/02	Présentation du projet, découverte des éléments et des outils
11/02 et 18/02	Recherches, documentation et collecte d'informations
11/03 au 25/03	Allumer les leds en fonction de la couleur détectée par la CMUCam (Montage + programme sur Arduino)
08/04 et 15/04	Servomoteurs (Travail sur le programme puis essai expérimental)
26/05	Commencement de la répartition des tâches pour le dossier et initiation du projet Robot voiture
13/05 au 27/05	Projet Robot final
03/05	Finalisation du dossier
10/05	Préparation de la soutenance

3. TRAVAIL REALISE ET RESULTATS

3.1. Analyse fonctionnelle



3.2. Description du matériel

Ce projet traitant de robotique, les enjeux majeurs ont donc été de réaliser des montages électroniques, concevoir des programmes informatiques et surtout veiller à ce que la mise en commun de chaque partie résulte en un robot qui fonctionnerait selon nos attentes. Pour cela, nous avons très vite été mises en contact avec un matériel relativement flexible et simple à utiliser, que nous nous proposons de présenter dans cette partie.

3.2.1. Le système Arduino

Il a été le pilier central de nos montages électroniques. En effet, le projet Arduino a été pensé comme un outil devant permettre aux débutants, amateurs ou professionnels de créer des systèmes électroniques plus ou moins complexes, et cela en alliant les performances de la programmation informatique et celles de l'électronique.

Nous avons pu ainsi constater qu'un des avantages certains de l'électronique programmée, c'est qu'elle simplifie les schémas électroniques et réduit la charge de travail à la conception d'une carte électronique.

Le système Aduino s'avère être compatible avec les systèmes d'exploitation les plus courants, à savoir Windows, Linux et Mac. Le logiciel, gratuit, est disponible en open source. La carte Arduino, quant à elle, est un circuit imprimé basé sur une interface entrée/sortie simple et dont les plans circulent librement sur internet. Son microcontrôleur peut être programmé afin de contribuer à la réalisation d'un large panel de projets électroniques trouvant une application dans des domaines variés (ex: construire un robot, contrôler des appareils dans le cadre de la domotique, communiquer avec un ordinateur…).

Evidemment, il existe plusieurs types de cartes Arduino mais pour notre part, nous nous sommes servies d'une Arduino Uno que nous avons programmée via l'interface Arduino.

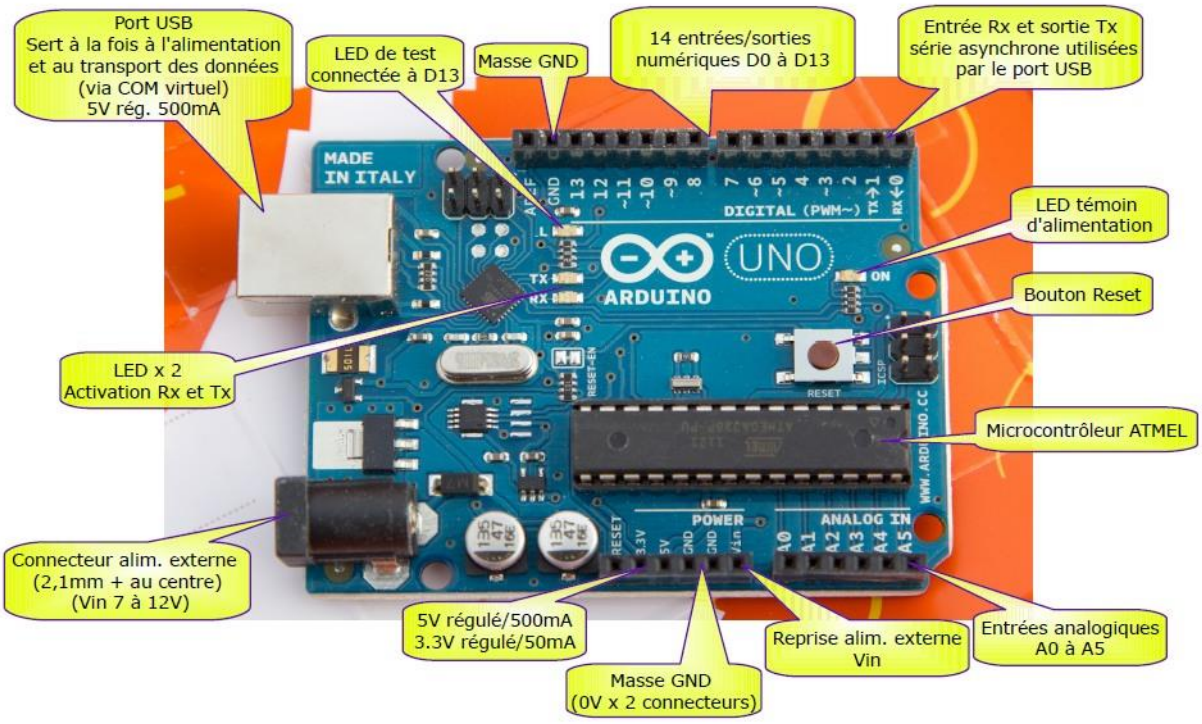
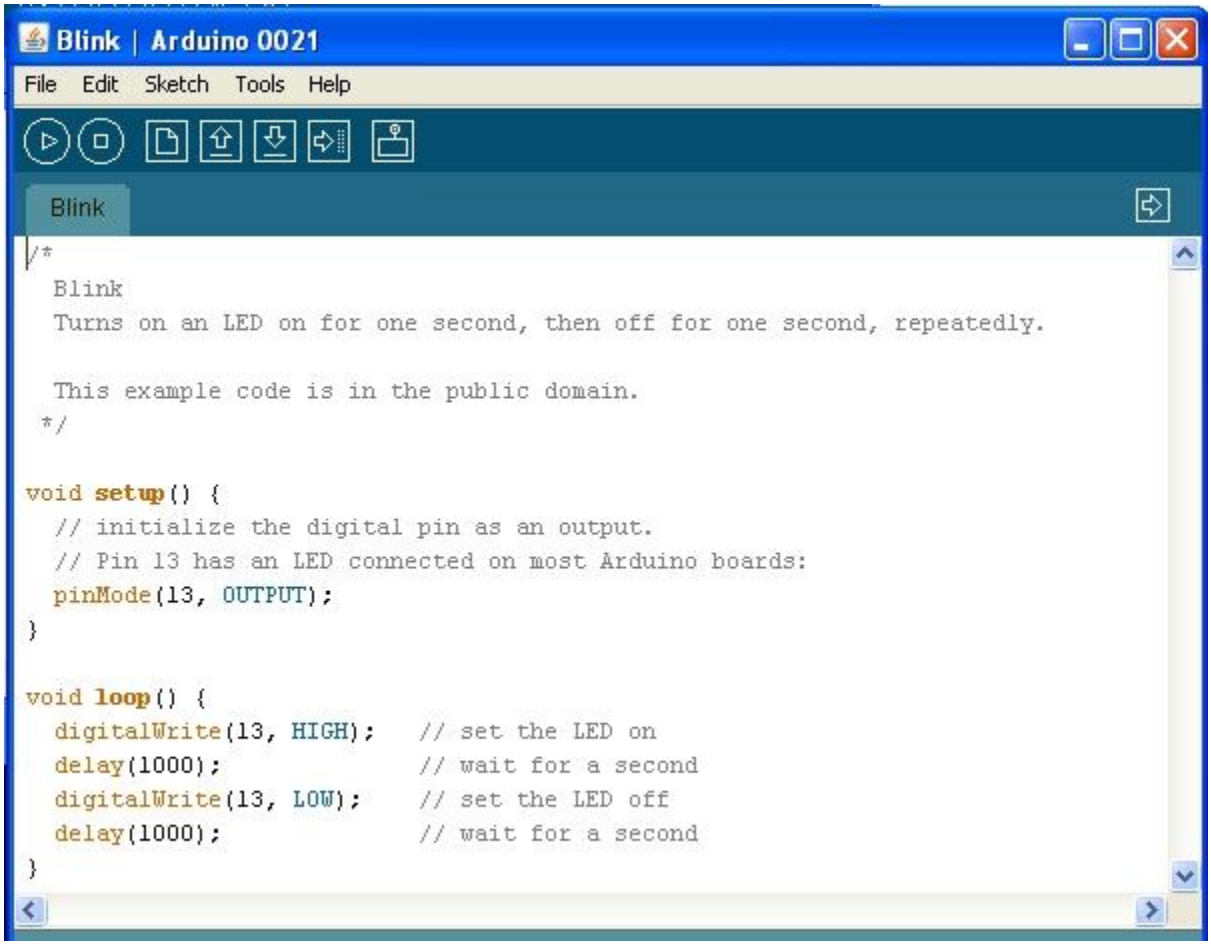


Figure 1 : Carte Arduino légendée tirée du site developpez.com



Un aperçu de l'interface Arduino (programme "Blink")

3.2.2. La CMUcam4

La problématique de notre projet était la vision par ordinateur, ou comment permettre à une machine de comprendre ce qu'elle « voit » lorsqu'on la connecte à une ou plusieurs caméras. La vision artificielle nécessite une grande puissance de traitement, et c'est exactement ce que nous apporte un matériel comme la CMUcam4.

1. Fonctionnalités et présentation matérielle

La CMUcam est un projet open source réalisé par la Carnegie-Mellon University. Il s'agit d'un système présentant une caméra et un microprocesseur intégré et qui dispose d'un port série qui permet de créer une interface avec une machine. Plus particulièrement, ce matériel est utilisé pour suivre les couleurs ou collecter des statistiques d'images de base. Nous avons constaté que les meilleures conditions pour une CMUcam performante sont obtenues lorsqu'il y a des couleurs intenses et très contrastées. Par exemple, on pourra très facilement suivre une boule rouge sur un fond blanc.

La CMUcam est destinée à des projets de robotique personnelle (dans le cadre de l'enseignement ou du loisir). Ainsi la simplicité d'utilisation entrait d'office dans le

cahier des charge initial du système: afin de pouvoir être aisément embarquée sur un petit robot elle présente des dimensions plutôt faible. La CMUcam4 que nous avons, pour notre part, utilisée est la quatrième génération de CMUcam. Elle mesure 53 mm de long pour 54 mm largeur et présente un processeur dédié à la reconnaissance de formes simples: une parallaxe Propeller P8X32A relié à un module caméra à base d'un capteur Omnivision 9665 CMOS.

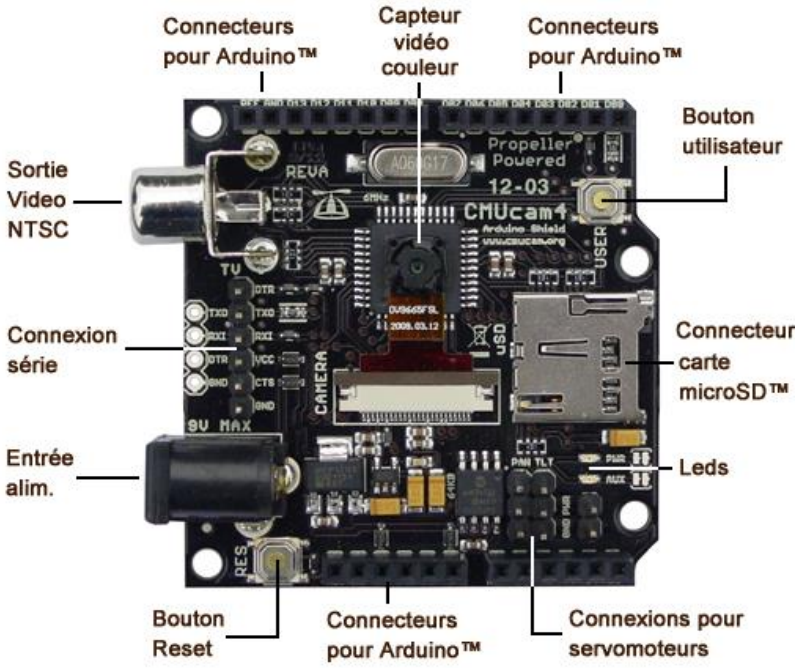


Figure 2 : CMUCam4 légendée tirée du site lextronic.fr

2. Utilisation et compatibilité avec Arduino

La configuration la plus courante est de faire la CMUcam communiquer avec un microcontrôleur par le biais d'un un port série standard TTL. Il peut s'agir d'un ordinateur (et donc la connexion se fait par USB ou RS232) ou de toute carte de développement possédant un port adapté. Nous avons utilisé une carte Arduino pour ses performances évidentes mais aussi parce que le format du circuit imprimé de la CMUcam4 est identique à celui des platines d'extension "Shield" pour Arduino.

Figure 3 : Assemblage Arduino + CMUCam4 - image de lextronic.fr

Le microprocesseur de la CMUcam va s'occuper du lourd traitement des données qu'il transmettra ensuite à l'Arduino.

3.3. Projet 1 : détection des couleurs

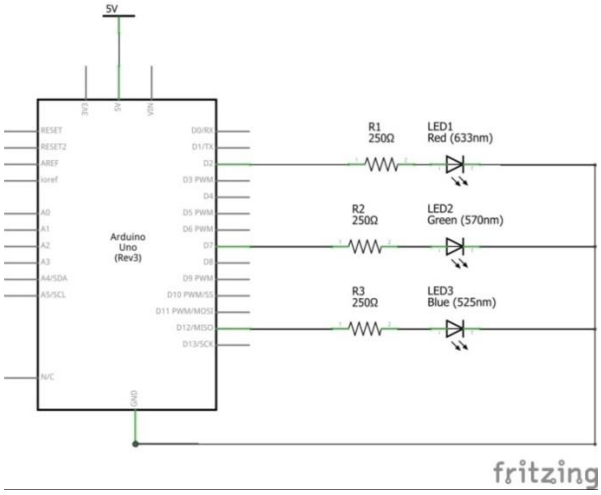
Lors de notre premier projet, notre objectif était de nous familiariser avec la carte ARDUINO Uno, la carte CMUcam4, ainsi que la programmation en C, grâce au logiciel Arduino. Pour cela, nous devions faire fonctionner un programme (voir annexe 1) ainsi que mettre en place un circuit électrique permettant à une Led (rouge, verte ou jaune) de s'allumer lorsque la CMUcam4 reconnaissait la couleur correspondante (respectivement rouge, verte ou bleue).

Tout d'abord, pour que le programme fonctionne, nous avons dû chercher quels étaient les codes RVB (Rouge, Vert, Bleu) qui correspondaient à chaque couleur recherchée. Nous avons donc déterminé que nos couleurs se situeraient dans ces intervalles :
- Bleu : (0,60 ; 0,60 ; 120,255)
- Vert : (0,60 ; 100,255 ; 0,60)
- Rouge : (120,255 ; 0,60 ; 0,60)

```
// BLEU
cam.trackColor(0,60,0,60,120,255);
cam.getTypeTDataPacket(&data);
if(data.pixels > 1) {
  lightDown = HIGH;    // Si la couleur repérée appartient au domaine definit pour le bleu, alors la led s'allume.
}
```

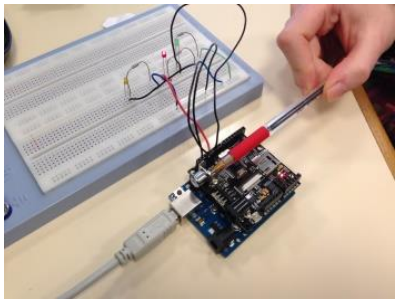
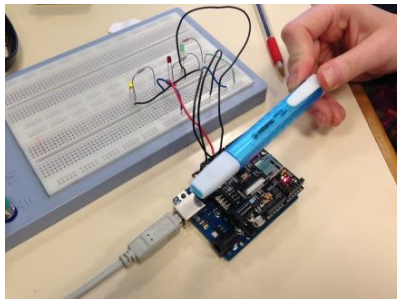
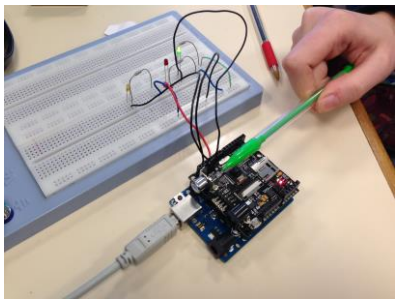
On va définir une couleur en déterminant l'intensité de chaque couleur primaire présente (sur une échelle de 0 à 255). Les paires de nombres, séparés par une virgule, correspondent aux limites minimales et maximales des composantes rouge, vert, et bleu. C'est à dire, par exemple, que pour reconnaître du bleu, l'intensité de notre objet devra se situer dans un intervalle de 0 à 60 pour le rouge et le vert, et entre 120 et 255 pour le bleu.

Ensuite, nous avons réalisé le montage suivant :

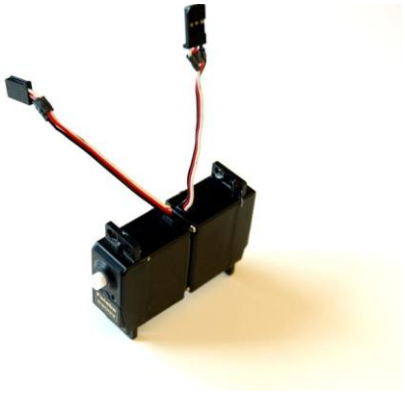


Pour y parvenir, nous avons dû trouver la bonne résistance, notamment en utilisant la loi d'Ohm, avec I=10mA, V=5V, on a donc R=250 Ohm.

Une fois le programme créé, les résistances choisies convenablement, et le montage effectué, il nous restait à téléverser le programme sur la Arduino. Nous avons cependant rencontré un problème, nous nous sommes rendu compte que pour téléverser correctement le programme sur la carte Arduino, il fallait soit mettre la CMUCam4 en mode arrêté, soit déconnecter la carte CMUCam4 de la Arduino. Nous avons donc séparé les deux cartes, et une fois le programme téléversé sur la carte, nous avons pu reconnecter la CMUCam4. Finalement, on peut voir sur les photos ci-dessous que lorsqu'on passe un objet rouge (ou bleu ou vert), la led rouge (ou jaune ou verte respectivement) s'allume.

3.4. Projet 2 : fonctionnement des servomoteurs



"Un servomoteur (souvent abrégé en « servo », provenant du latin *servus* qui signifie « esclave ») est un moteur capable de maintenir une opposition à un effort statique et dont la position est vérifiée en continu et corrigée en fonction de la mesure. C'est donc un système asservi."

- Wikipédia

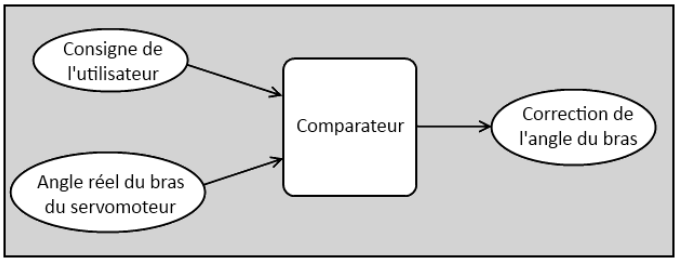
3.4.1. L'électronique d'asservissement du servomoteur

L'asservissement est un moyen de gérer une consigne de régulation selon une commande d'entrée. Dans le cas du servomoteur, on l'alimente et on lui envoie un signal de commande qui permet de définir à quel angle va se positionner le bras du servomoteur. Ce dernier va s'exécuter et faire en sorte de toujours garder la position de son bras à l'angle voulu. Pour pouvoir assurer ce maintien de position, le servo utilise l'électronique d'asservissement.

Il faut savoir qu'un servomoteur est un ensemble de mécanique et d'électronique composé de :

- un moteur à courant continu (CC) souvent de petite taille ;
- une carte électronique d'asservissement ;
- un réducteur de vitesse ;
- un potentiomètre, pour contrôler la position de l'axe du moteur ;
- un axe dépassant hors du boîtier avec différents bras ou roues de fixation.

Le capteur de position du bras, qui n'est autre qu'un potentiomètre couplé à l'axe du moteur, mesure la tension au point milieu et permet d'obtenir une tension image de l'angle d'orientation du bras. Cette position est ensuite comparée, à la consigne (le signal de commande) qui est ensuite transmise au servomoteur. Après une rapide comparaison entre la consigne et la valeur réelle de position du bras, l'électronique de commande du servo va appliquer une correction si le bras n'est pas orienté à l'angle imposé par la consigne.



3.4.2. La commande d'un servomoteur avec Arduino

Dans le cadre de notre projet, nous avons utilisé un servomoteur à électronique de commande de type analogique. C'est le type le plus courant et le moins chers. En l'occurrence, il s'agit d'un servomoteur de la marque Futuba version "rotation continue": le servo permet de tourner en permanence dans les 2 sens suivant le signal envoyé (la valeur du rapport cyclique dont nous parlerons un peu plus loin).

Commander un servo avec Arduino est chose facile, du moins à la portée d'un débutant grâce à une bibliothèque intégrée à l'environnement Arduino. En fait, la connectique d'un servomoteur se résume à trois fils : deux pour l'alimentation positive et la masse et le dernier pour le signal de commande. Lors de la confection de notre carte électronique, nous avons choisi d'utiiser la pin 5V de la carte Arduino pour alimenter nos servomoteurs.

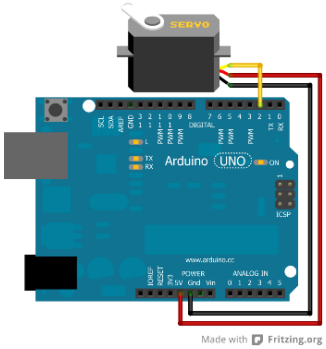


Figure 4 : Exemple de montage Arduino + servomoteur

Voyons par exemple ce que nous avons pris en compte dans la conception d'un programme pour faire tourner les roues:

```
int cmdmoteur = 11;
void setup() {
  pinMode(cmdmoteur, OUTPUT);
}

void loop() {
  analogWrite(cmdmoteur, 250);
}
```

Nous avons utilisé la PWM (Pulse Width Modulation) de l'Arduino et connecté nos quatre servomoteurs aux broches 7, 9, ? et ? (j'oublie). Et c'est avec la fonction analogWrite() de Arduino que nous envoyons la commande aux servos. Elle prend deux arguments:

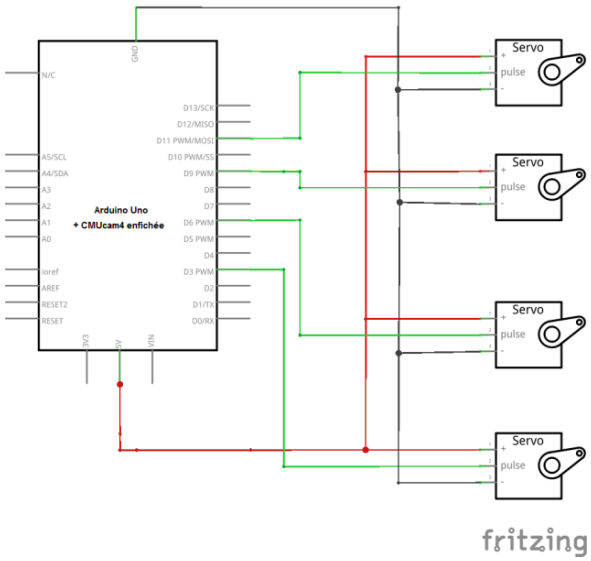
- le numéro de la broche où l'on veut générer la PWM;
- la valeur du rapport cyclique à appliquer. Cette valeur est en fait un nombre entier compris entre 0 et 255.

Il est nécessaire d'apporter quelques concernant le second argument. En effet, normalement, le rapport cyclique s'exprime de 0 à 100 % cependant, dans cette fonction il s'exprimera de 0 à 255 (sur 8 bits). Ainsi, pour un rapport cyclique de 0% nous enverrons la valeur 0, pour un rapport de 50% on enverra 127 et pour 100% ce sera 255. Les autres valeurs sont considérées de manière proportionnelle entre les deux.

Ainsi, eentre 0 et 127, la roue tournera plus ou moins vite en marche avant. Au delà de 127, elle ira en marche arrière, la vitesse de rotation augmentant avec la valeur envoyée. L'avantage du modèle, c'est qu'avec un seul pin du microcontrôleur Arduino, nous pouvons contrôler la vitesse et le sens de rotation et ainsi piloter les roues du robot.

3.5. Projet 3 : le robot voiture

Voici le schéma électrique général du robot que nous avions pour but de réaliser :



3.5.1. Partie programmation

Pour la partie programmation du robot, nous avons pu réutiliser tout ce que nous avions codé dans les parties précédantes du projet (c'est-à-dire reconnaître une couleur et commander les servomoteurs). Nous avions donc déjà la syntaxe et les instructions propres à ces fonctions particulières.Il restait à mettre en relation ces fonctions pour avoir un programme complet : pouvoir par exemple commander les servomoteurs pour faire aller le robot à droite si la caméra détecte du rouge dans la partie droite de l'image.

Pour obtenir un algorithme qui nous permette de mettre en relation les deux fonctions pour commander le robot comme nous le voulions, nous nous sommes appuyées sur un programme que nous a donné notre enseignant, réalisé lors d'un projet d'étudiants de troisième année du département Mécanique. Celui-ci commande un robot Spoonbot, dont les objectifs sont les mêmes que les nôtres. La différence est que ce programme est destiné à commander une carte Arduino et une caméra CMUcam3, la version précédente de la CMUcam4 que nous utilisions.

Nous avons donc repris le corps de ce programme en l'adaptant à notre matériel. Les fonctions disponibles pour les deux caméras n'étant pas les mêmes, nous avons dû « traduire » les instructions propres à la CMUcam3 pour la CMUcam4 (par exemple pour tous les réglages d'initialisation de la caméra).

Il a fallu modifier toute la partie commande des servomoteurs de notre programme « modèle », puisque dans celui-ci, ils sont directement commandés par la caméra, alors que nous voulions les commander depuis la carte Arduino. L'ancien programme fait d'ailleurs appel à des fonctions définies dans un fichier .c et inclus dans le programme principal, pour commander les servomoteurs avec les instructions propres à la CMUcam3. Dans notre cas, le programme s'est beaucoup simplifié puisque les instructions pour commander les servomoteurs sont très simples (une ligne), comme nous l'avons vu dans la partie 3. Notre programme final ne fait plus appel à d'autres fichiers inclus.

Après ce travail, nous avons obtenu un programme .ino beaucoup plus simple au niveau de la syntaxe et de la structure que le programme précédant, un grand nombre d'instructions se simplifiant grâce aux fonctions déjà définies pour la CMUcam4. Le plus difficile du travail a finalement été l'analyse du premier programme, qui n'est pas forcément évident à comprendre, pour en comprendre le fonctionnement et sélectionner ce qui était à garder pour notre version.

3.5.2. Commentaires du programme réalisé (code en annexe 2) :

Bibliothèques utilisées :

- cmucam4.h : définit les fonctions propres à la CMUcam4
- cmucom4.h : pour définir la communication CMUcam4-Arduino

Avant d'implémenter les deux fonctions *setup* et *loop*, qui sont la base de la syntaxe de tout programme .ino pour commander une carte Arduino, nous avons défini quelques fonctions pour être capable de diriger le robot avec les servomoteurs. Il y en a donc cinq:

- spoonBot_tourelle_pos(int position) : dirige la tourelle de gauche à droite
- spoonBot_spoon_pos(int position) : dirige la tourelle de haut en bas
- spoonBot_left(int speed) : tourne le robot à gauche
- spoonBot_right(int speed) : tourne le robot à droite

- spoonBoot_stop() : stoppe les roues du robot

Dans chacune de ces fonctions, nous utilisons la commande *analogWrite(pin, value)* qui permet d'envoyer une valeur sur les sorties *pwm* utilisées pour la communication avec les servomoteurs et donc de les diriger. Il nous faut également effectuer quelques tests dans chacune : par exemple, pour vérifier que l'on va faire tourner le moteur dans la bonne direction, ce qui dépend du sens de montage de celui-ci (pour cela il faut utiliser des constantes que l'on définit pour chaque moteurs après le montage), ou bien pour vérifier que l'on pas déjà atteint une « position maximale » (si la tourelle ne peut pas tourner plus, par exemple).

Dans la fonction *setup*, on effectue simplement tous les réglages initiaux de la caméra CMUcam4de façon automatique, et l'on définit les modes de sorties des pins correspondantes aux moteurs sur la carte Arduino.

Enfin, la fonction *loop* constitue le corps de notre programme : c'est une boucle répétée indéfiniment. On commence par chercher du rouge sur l'image détectée par la CMUcam4, avec la fonction que l'on a vu précédemment. Ces instructions renvoient différentes informations sous forment d'une structure de données, qui nous permettent de déterminer dans quelle partie de l'image les pixels rouges ont été détectés. Avec cette information, on peut ensuite diriger les moteurs selon les différents cas, grâce aux fonctions que l'on a défini auparavant. Et en particulier, la variable track_flag est utilisée pour savoir si on doit mettre en mouvement ou non le moteur : elle est 0 si il n'y en a pas besoin (pas de rouge détecté, ou s'il est déjà centré), et sinon à 1, dans le cas où l'on doit diriger les servomoteurs.

3.5.3. Mise en place

Afin de faire communiquer notre carte Arduino avec la CMUcam4, nous avons utilisé une plaque shield servant d'interface entre ces deux composants majeurs. Pour cela nous avons soudé des fils électriques sur cette plaque, chaque fil électrique correspondant a une sortie de la arduino. Ainsi chaque sortie commandait un des quatre servo moteurs de notre robot. La sortie 7 faisait tourner la roue droite, la sortie 5 la roue gauche, la sortie 9 le mouvement horizontal et la sortie 11 le mouvement vertical, couvrant ainsi la totalité des mouvements dans l'espace.

Pour souder nos composants nous avons utilisé un fer à souder et de l'étain en essayant de faire en sorte que chaque soudure ne se chevauche pas pour éviter tout faux contact. Cette opération a été la même pour l'entrée et la masse. Une fois les soudures terminées, nous avons téléversé notre programme spoonbot sur notre arduino pour vérifier qu'il fonctionnait bien. Cependant nous nous sommes alors rendu compte qu'il y avait de mauvaises transmissions et avons, pour détecter les mauvaises soudures, utilisé un voltmètre entre chaque composant vérifiant donc l'existence d'une tension.

Malgré tout il subsistait des problèmes de déplacement d'un des servo moteurs, nous empêchant donc d'assurer le bon fonctionnement de notre robot et nous obligeant à abandonner le travail effectué sur ce dernier. Nous nous sommes alors lancées sur un nouveau robot n'utilisant pas des servo moteurs mais des moteurs utilisant un courant continu.

3.6. Problèmes rencontrés

Le problème majeur que nous avons rencontré est la disparité dans les compétences de chaque membre du groupe notamment en matière de programmation ou de connaissance de la mécanique. Cependant, avec le temps et à force de travailler ensemble nous avons appris à utiliser les forces de chacune au mieux.

Aussi, le professeur encadrant a souvent comblé nos lacunes dans ces domaines en nous aidant et en étant présent dès que nous en avions besoin, ce qui nous permis d'avancer au mieux dans notre projet.

Lorsque nous avons voulu tester notre projet final, nous avons réalisé qu'il ne fonctionnait pas, le problème était certainement dû à une erreur de soudure. En effet, après quelques tests, nous avons réalisé que, bien que le programme compile, l'information n'était pas transmise par le circuit imprimé. Nous avions déjà perdu plusieurs heures à trouver l'erreur, c'est pourquoi nous avons choisi, et ce par peur de manque de temps, de présenter un modèle très proche du notre, dont les moteurs sont à courant continu (tandis que le nôtre possédait des servomoteurs).

On a remplacé les servomoteurs par des moteurs simples, à courant continu, comme il est facile d'en trouver dans le commerce. Voici une photo du nouveau matériel :

Figure 5 : Moteurs à courant continu - photo tirée du site shop.robotee.com/

Il s'agit d'un twin moteur gearbox tamiya dont voici un schéma détaillé :

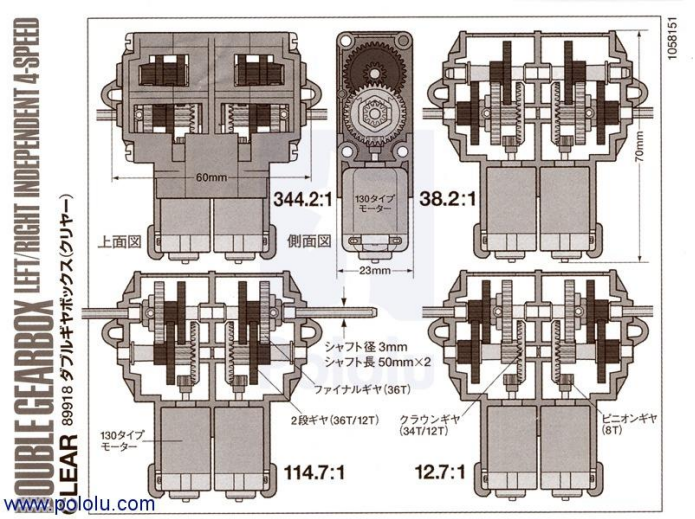


Figure 6 : Schéma d'un moteur à courant continu - image tirée du site pololu.com

C’est la seule différence avec notre projet précédent, et cela nous permet d’éviter les problèmes de montage rencontrés auparavant. Les programmes utilisés sont les mêmes et fonctionnent sur la même base que le projet en servomoteurs. Seule la partie informatique qui contrôle les servomoteurs change. Elle est ici beaucoup plus facile à manipuler. En effet, un servomoteur est un type de moteur particulier utilisé en robotique ou véhicules télécommandés. Il est en général moins puissant qu'un moteur pas à pas.

4. CONCLUSION ET PERSPECTIVES

Une amélioration qu'il faudrait apporter au robot, serait d'exploiter d'avantage la caméra. En effet, la CMUCam4 a des caractéristiques très avantageuses telles que l'environnement de développement très flexible avec des sources ouvertes (open source), une plate-forme à faible coût et un capteur vidéo intelligent incorporé. Aussi, il serait intéressant et demanderait peu de travail de rajouter des couleurs détectées au projet présenté (par exemple, nous avions étudié la possibilité d’un projet inspiré des bateaux, qui, s’il détectait une lumière verte, irait à bâbord (gauche) et s’il détectait une lumière rouge, se dirigerait vers la droite afin de rentrer dans le port.

En conclusion, nous pouvons dire que nous sommes satisfaits de l'aboutissement de ce projet. Au début du semestre, nous n'avions que de vagues notions d'électronique, et n'avions pour la plupart d'entre nous jamais touché de près le monde de la robotique.

Ce projet nous aura non seulement permis de travailler en équipe, mais également de développer notre autonomie, face à du matériel qui nous était inconnu. Nous avons su palier les problèmes que nous avons rencontrés au cours du projet, un point très important pour notre futur métier d'ingénieur.

Nous avons su faire preuve d'une bonne organisation, pendant que les uns travaillaient sur la partie la plus mécanique du projet, les autres s'occupaient de la partie logicielle, ou encore de l'écriture du rapport. La répartition des rôles fut bien réalisée, chaque membre du groupe a pu travailler de manière efficace.

5. BIBLIOGRAPHIE ET CREDITS D'ILLUSTRATION

Site developpez.com : http://www.developpez.com/ (Valide à la date du 05/06/2014)

Site couleurs RVB : http://www.toutes-les-couleurs.com/code-couleur-rvb.php (Valide à la date du 11/03/2014)

Les schémas ont été réalisés grâce au logiciel Fritzing.

6. ANNEXES

6.1. Programme « Track Color » (Projet 1)

```
#include <CMUcam4.h>  // Bibliothèques
#include <CMUcom4.h>

#define LED_BLINK 5 // 5 Hz
#define WAIT_TIME 5000 // 5 seconds

#define LED_UP 11 // Les nombres correspondent à la sortie sur la
#define LED_MID 10    Arduino.
#define LED_DOWN 12
#define LED_DEBUG 13

CMUcam4 cam(CMUCOM4_SERIAL);

void setup()
{
  cam.begin();

  cam.LEDOn(LED_BLINK);
  delay(WAIT_TIME);

  cam.autoGainControl(false);
  cam.autoWhiteBalance(false);

  cam.LEDOn(CMUCAM4_LED_ON);


  // Déclaration des sorties numériques
  pinMode(LED_UP, OUTPUT);
  pinMode(LED_MID, OUTPUT);
  pinMode(LED_DOWN, OUTPUT);
  pinMode(LED_DEBUG, OUTPUT);
}

void loop()
{
  CMUcam4_tracking_data_t data;

  // l'état des LED.
  int lightUp, lightMid, lightDown, lightDebug;

  for(;;)
  {
    // par défaut, on reset toutes les valeurs
    lightUp = LOW;
    lightMid = LOW;
    lightDown = LOW;
    lightDebug = LOW;

    // ROUGE
    cam.trackColor(120,255,0,60,0,60); // chercher dans le rouge
```

```
    cam.getTypeTDataPacket(&data);          //  récupérer  les  infos
transmises par la cam
    if(data.pixels > 1) {                 // si on détecte des pixels
de cette couleur
      lightUp = HIGH;                   // on allumera la LED UP
    }

    // VERT
    cam.trackColor(0,60,0,255,0,120);
    cam.getTypeTDataPacket(&data);
    if(data.pixels > 1) {
      lightMid = HIGH;
    }


    // BLEU
    cam.trackColor(0,60,0,60,120,255);
    cam.getTypeTDataPacket(&data);
    if(data.pixels > 1) {
      lightDown = HIGH;
    }

    digitalWrite(LED_UP, lightUp);
    digitalWrite(LED_MID, lightMid);
    digitalWrite(LED_DOWN, lightDown);
    digitalWrite(LED_DEBUG, lightDebug);
  }

}
```

6.2. Programme du Robot final (Projet 3)

```
#include <CMUcam4.h> // bibliothèque de fonctions propres à la CMUcam4
#include <CMUcom4.h> // bibliothèque de fonctions pour la communication CMUcam4-Arduino
#define LED_BLINK 5 // 5 Hz Constante pour le clignotement de la led lors du réglage de la CMUcam4
#define WAIT_TIME 5000 // 5 seconds Constante pour attendre le temps du réglage automatique de la caméra
CMUcam4 cam(CMUCOM4_SERIAL); // definit la communication entre cmucam4 et Arduino

//Définition des pins de sortie pour les moteurs
const int MOTEUR_GAUCHE = 1;
const int MOTEUR_DROITE = 2;
const int TOURELLE_HORIZONTAL = 3;
const int TOURELLE_VERTICAL = 4;

//Constantes definissant certaine positions des moteurs : milieu/droit/centre
const int32_t SBOT_LEFT_MID = 71, SBOT_RIGHT_MID = 77; //constantes pour que le moteur ne bouge pas
const int32_t SBOT_SPOON_MID = 128, SBOT_SPOON_DOWN = 0, SBOT_SPOON_UP = 255;
const int32_t SBOT_TOURELLE_MID = 128, SBOT_TOURELLE_GAUCHE = 0,
SBOT_TOURELLE_DROITE = 255;

//Constantes directions moteurs (si on monte le moteur dans un sens ou dans l'autre) à adapter selon le montage
const int8_t SBOT_LEFT_DIR = 1, SBOT_RIGHT_DIR = -1, SBOT_SPOON_DIR = 1,
SPBOT_TOURELLE_DIR = 1;
```

```
void spoonBot_spoon_pos(int position) // pour faire bouger la tourelle (de haut en bas)

{
	//Le moteur est-il monté dans le bon sens ?
	if (SBOT_SPOON_DIR < 0)
	{
	//La position demandée dépasse-t-elle la position minimale ?
	if (SBOT_SPOON_MID - position < SBOT_SPOON_DOWN)
	analogWrite(TOURELLE_VERTICAL, SBOT_SPOON_DOWN);//Si oui, on met le servomoteur à la position minimale
	else
	analogWrite(TOURELLE_VERTICAL, SBOT_SPOON_MID - position); //Si non, on le met à la bonne position (différence avec la valeur "milieu")
	}
	else
	{
	//Même chose à l'envers
	if (SBOT_SPOON_MID + position > SBOT_SPOON_UP)
	analogWrite(TOURELLE_VERTICAL, SBOT_SPOON_UP);
	else
	analogWrite(TOURELLE_VERTICAL, SBOT_SPOON_MID + position);
	}
}
void spoonBot_tourelle_pos(int position) // pour faire tourner la tourelle (de droite à gauche)
{
	//Voir fonction précédente
	if (SBOT_TOURELLE_DIR < 0)
	{
	//A changer si la tourelle tourne dans le mauvais sens
	if (SBOT_TOURELLE_MID - position < SBOT_TOURELLE_GAUCHE)
	analogWrite(TOURELLE_HORIZONTAL, SBOT_TOURELLE_GAUCHE);
	else
	analogWrite(TOURELLE_HORIZONTAL, SBOT_TOURELLE_MID - position);
	}
	else
	{
	if (SBOT_TOURELLE_MID + position > SBOT_TOURELLE_DROITE)
	analogWrite(TOURELLE_HORIZONTAL, SBOT_TOURELLE_DROITE);
	else
	analogWrite(TOURELLE_HORIZONTAL, SBOT_TOURELLE_MID + position);
	}
}
void spoonBot_left(int speed) // pour faire tourner le robot à gauche avec les roues
{
	//Moteur monté dans le "bon" sens ?
	if (SBOT_LEFT_DIR > 0)
	analogWrite(MOTEUR_GAUCHE, SBOT_LEFT_MID - speed);
	else
	analogWrite(MOTEUR_GAUCHE, SBOT_LEFT_MID + speed); //Même chose mais à l'envers
	if (SBOT_RIGHT_DIR > 0)
	analogWrite(MOTEUR_DROITE, SBOT_RIGHT_MID + speed);
	else
	analogWrite(MOTEUR_DROITE, SBOT_RIGHT_MID - speed);
}

void spoonBot_right(int speed) // Pour faire tourner le robot à droite
{
	//Voir fonction précédente, mais les roues vont dans l'autre sens
```

```
            if (SBOT_LEFT_DIR > 0)
            analogWrite(MOTEUR_GAUCHE, SBOT_LEFT_MID + speed);
            else
            analogWrite(MOTEUR_GAUCHE, SBOT_LEFT_MID - speed);
            if (SBOT_RIGHT_DIR > 0)
            analogWrite(MOTEUR_DROITE, SBOT_RIGHT_MID - speed);
            else
            analogWrite(MOTEUR_DROITE, SBOT_RIGHT_MID + speed);
}
void spoonBoot_stop(); // pour stopper tout mouvement des roues
{
  analogWrite(MOTEUR_GAUCHE, SBOT_LEFT_MID);
  analogWrite(MOTEUR_DROITE, SBOT_RIGHT_MID);

void setup() // initialisation de la caméra et des sorties de la carte Arduino
{
            cam.begin();

            // Wait for auto gain and auto white balance to run.

            cam.LEDOn(LED_BLINK);
            delay(WAIT_TIME);

            // Turn auto gain and auto white balance off.

            cam.autoGainControl(false);
            cam.autoWhiteBalance(false);

            cam.LEDOn(CMUCAM4_LED_ON);

            //Configuration des pins (mode sortie)
            pinMode(MOTEUR_GAUCHE, OUTPUT);
            pinMode(MOTEUR_DROITE, OUTPUT);
            pinMode(TOURELLE_HORIZONTAL, OUTPUT);
            pinMode(TOURELLE_VERTICAL, OUTPUT);
}

void loop() // fonction détection du rouge et commande du déplacement du robot en fonction de l'image détectée
{
// déclarations :
            CMUcam4_tracking_data_t data; // type des données obtenues par le track color
            int32_t spoon_loc = 0, tourelle_loc = 0; // pour diriger les moteurs
            const uint8_t midx = 156 / 2; // coordonnées du milieu de l'image
            const uint8_t midy = 120 / 2; // pareil
  uint8_t track_flag = 0; // prend la valeur 0 si pas de rouge (pas besoin de bouger le robot), 1 si il ya du rouge
(on va bouger le robot)

            //Boucle infinie
            for(;;)
            {
             cam.trackColor(120, 255, 0, 60, 0, 60); // chercher dans le rouge
            cam.getTypeTDataPacket(&data);  // récupérer les infos transmises par la cam
            if(data.pixels > 1)           // si on détecte des pixels de cette couleur (à augmenter si la couleur est mal
détectée)
            {
     //Si le centre de la balle est trop à droite, on envoie la tourelle vers la droite
            if(data.mx > midx + 20)
```

```
        {
        track_flag = 1;
        tourelle_loc += 10;
        }
        //Même chose à gauche
        else if(data.mx < midx - 20)
        {
        track_flag = 1;
        tourelle_loc -= 10;
        }
        //Si le centre de la balle est trop en bas, on met la tourelle vers le bas
        if(data.my < midy - 10)
        {
        track_flag = 1;
        spoon_loc += 3;
        }
        //Même chose en haut
        else if(data.my > midy + 10)
        {
        track_flag = 1;
        spoon_loc -= 3;
        }
        //Si la tourelle est à sa position la plus tournée, on essaye pas de la tourner plus : on tourne les roues :
        if (tourelle_loc > 100)
        {
        track_flag = 1;
        spoonBot_right(10); // fonction définie plus haut pour aller à droite
        delay(10);
        }
        //Pareil dans l'autre sens
        if (tourelle_loc < -100)
        {
        track_flag = 1;
        spoonBot_left(10); // fonction définie plus haut pour aller à gauche
        delay(10);
        }
        //On a tourné les roues, donc on n'essaie pas de tourner plus la tourelle
        if(spoon_loc > 100)
        spoon_loc = 100;
        if(spoon_loc < -100)
        spoon_loc = -100;

        if(tourelle_loc > 100)
        tourelle_loc = 100;
        if(tourelle_loc < -100)
        tourelle_loc = -100;

        //On met la position de la tourelle à la nouvelle position calculée
        spoonBot_spoon_pos(spoon_loc);
        spoonBot_tourelle_pos(tourelle_loc);
        }
        //Si le robot ne doit pas bouger (balle au centre de l'image ou en dehors), on l'arrête
        if(track_flag == 0)
        spoonBot_stop();
        }

}
```