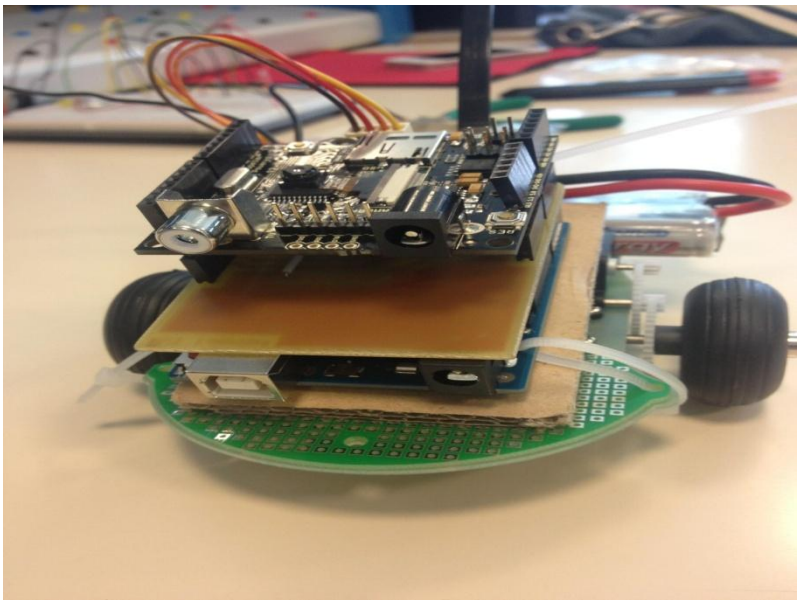
INSA ROUEN
À taille humaine,
à l'échelle du monde

Projet de Physique P6-3
STPI/P6-3/2012 – 53

Mise en œuvre d'un module de reconnaissance vidéo CMU-CAM4



Etudiants: Daming LI Anatole CHAUMONT
Wenyu LI Soukaina MOULAHID

Enseignant-responsable du projet:
Ludovic HENRIET

Date de remise du rapport : **17/06/2013**

Référence du projet : **STPI/P6/2013 – 53**

Intitulé du projet : **Mise en œuvre d'un module de reconnaissance vidéo CMU-CAM4**

Type de projet : **expérimental et programmation**

Objectifs du projet:

- **Savoir programmer sur Arduino**
- **Savoir monter les circuits basés sur Arduino**
- **Comprendre le fonctionnement de la CMUCam4**
- **Savoir programmer sur Arduino avec la bibliothèque CMUCam/Arduino**
- **Realiser un robot simple à l'aide de la CMUcam4 et Arduino**

Mots-clefs du projet :
CMUcam4
Arduino
Robot
Color-Tracking

Notations et acronymes

CMUcam 4:

Un module autonome de vision intégrant une caméra et un microprocesseur.

ARDUINO:

Un circuit imprimé sur lequel se trouve un micro contrôleur qui peut être programmé pour analyser et produire des signaux électriques de manière à effectuer des tâches très diverses.

L293D:

Circuits intégrés qui permettent de diriger les deux moteurs.

TABLE DES MATIERES:

I. INTRODUCTION

1. *Présentation du projet*

Dans le cadre de notre projet P6 de 4 ème semestre, nous avons choisi de nous intéresser au module de reconnaissance vidéo CMUcam4, surtout car nous étions tous très curieux de savoir comment fonctionne ce module que nous ne connaissions pas avant, d'autant plus que ce sujet nous a paru plus original que les autres sujets proposés en P6.

Notre objectif était clair dés la première séance avec le professeur encadrant: nous devions 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 Arduino et chargée avec le programme Arduino et finalement le robot avec ses 2 moteurs.

Après 11 séances de travaux pratiques, nos objectifs ont été atteints. Notre robot réagit à trois couleurs différentes, et reçoit trois ordres différents selon la couleur qu'il voit. S'il voit du rouge, il s'arrête. S'il voit du vert ou du bleu , il tourne respectivement à gauche ou à droite.

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 faisait 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 tout en tenant compte du fait qu'il fallait partager chacun de nos trois projets en une partiel informatique et une autre mécanique.

I. MÉTHODOLOGIE ET ORGANISATION DU TRAVAIL

1. *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.

2. *Objectif de groupe :*

Bien entendu, qui dit projet de groupe, dit aussi travail de groupe. Pour nous organiser, nous avons fait en sorte que pendant chaque séance, nous nous séparions en groupes de deux: un groupe pour traiter la partie informatique et un autre pour la partie mécanique,les montages et la construction du robot...

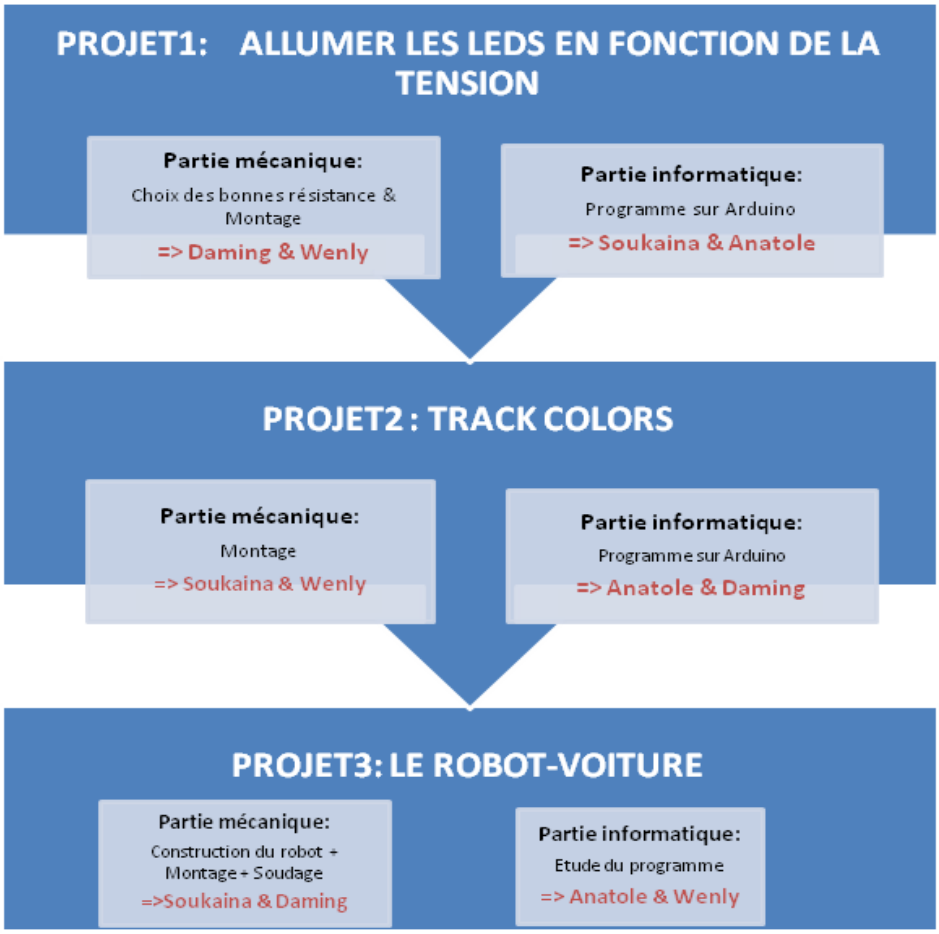


figure 1: répartition des tâches

Finalement, en ce qui concerne le rapport, dés la mi-Avril, nous avons créé un google doc auquel tous les membres du groupe avaient accès et qui nous a permis de bien avancer sur notre rapport.

3. Calendrier

05/02 *12/02*	***DOCUMENTATION & COLLECTE D'INFORMATIONS***	
05/03 *12/03*	***PREMIER PROJET** : Allumer leds en fonction de V* *Montage* *Programme sur Arduino + Réalisation*	
26/03 *9/04* *14/05* *21/05*	***DEUXIEME PROJET :** Track colors* *Documentation sur le langage Arduino* *Programmation*	***RAPPORT :*** • *Création du google doc*
28/05 *04/06* *09/06*	***TROISIEME PROJET :** Le robot suiveur* *Début construction du robot* *Programmation*	*Répartition des tâches + Rédaction*
13/06 *14/06*	*Préparation de la soutenance*	*Finalisation du rapport*

II. TRAVAIL RÉALISÉ ET RÉSULTATS:

1. ELECTRONIQUE:

a. LA CARTE ARDUINO

- ***Qu'est-ce que l'Arduino?***

Arduino est un circuit imprimé en matériel libre *(c'est a dire que les plans de la carte elle-même sont publiés en licence libre)* sur lequel se trouve un microcontrôleur qui peut être programmé pour analyser et produire des signaux électriques. Correctement utilisée, l'Arduino peut effectuer des tâches très diverses comme lle contrôle des appareils domestiques *(éclairage, chauffage…)* ou encore le pilotage d'un robot.C'est une plateforme basée sur une interface entrée/sortie simple. Arduino peut être utilisé pour construire des objets interactifs indépendants, ou bien peut être connecté à un ordinateur pour communiquer avec différents logiciels.

Au cours de notre projet, nous avons utilisé une carte Arduino Uno : La Arduino Uno est le modèle de référence de la série de cartes Arduino USB.

- ***Comment programmer l'Arduino?***

Un module Arduino est construit autour d'un micro contrôleur et de composants complémentaires qui facilitent la programmation et la communication avec d'autres circuits.Le micro contrôleur est préprogrammé avec un chargeur d'amorçage *(bootloader)* de façon à ce qu'un programmateur dédié ne soit pas nécessaire. Les modules sont programmés au travers d'une connexion série : RS-232, bluetooth ou tout simplement par USB.L'Arduino utilise la

plupart des entrées/sorties du microcontrôleur pour l'interfaçage avec les autres circuits. Plusieurs sortes d'extensions sont disponibles dans le commerce.

Le logiciel de programmation des modules Arduino est une application Java, libre et multi-plateformes. Servant d'éditeur de code et de compilateur, elle peut transférer le programme au travers de la liaison série . Il *est également possible de se passer de l'interface Arduino, et de compiler les programmes via l'interface en ligne de commande.*Le langage de programmation utilisé est le C++, lié à la bibliothèque de développement Arduino, qui permet l'utilisation de la carte et de ses entrées/sorties. La mise en place de ce langage standard rend aisé le développement de programmes sur les plates-formes Arduino, à toute personne maîtrisant le C ou le C++.

b. *LA CMUCAM4*

- **Qu'est-ce que la CMUcam4 ?**

La CMUCam4, développée par l'université Carnegie Mellon n'est pas une caméra comme nous l'entendons aujourd'hui. C'est plus un système de vision qu'une simple caméra. En effet, sa fonction principale sera de repérer des couleurs que l'utilisateur aura, au préalable, paramétrées. La CMUCam4 est le quatrième opus de la série CMUCam.

La CMUcam4 a de faibles dimensions (53 x 54 mm), Elle est conçue autour d'un puissant processeur Propeller™ associé à un capteur vidéo couleur CMOS Omnivision™. C'est un système composé également d'un bouton Reset, d'une led pour indiquer la présence d'alimentation, d'un connecteur permettant l'alimentation via une source externe, d'un autre connecteur pour carte mémoire microSD™et aussi un connecteur 6 broches pour Arduino.

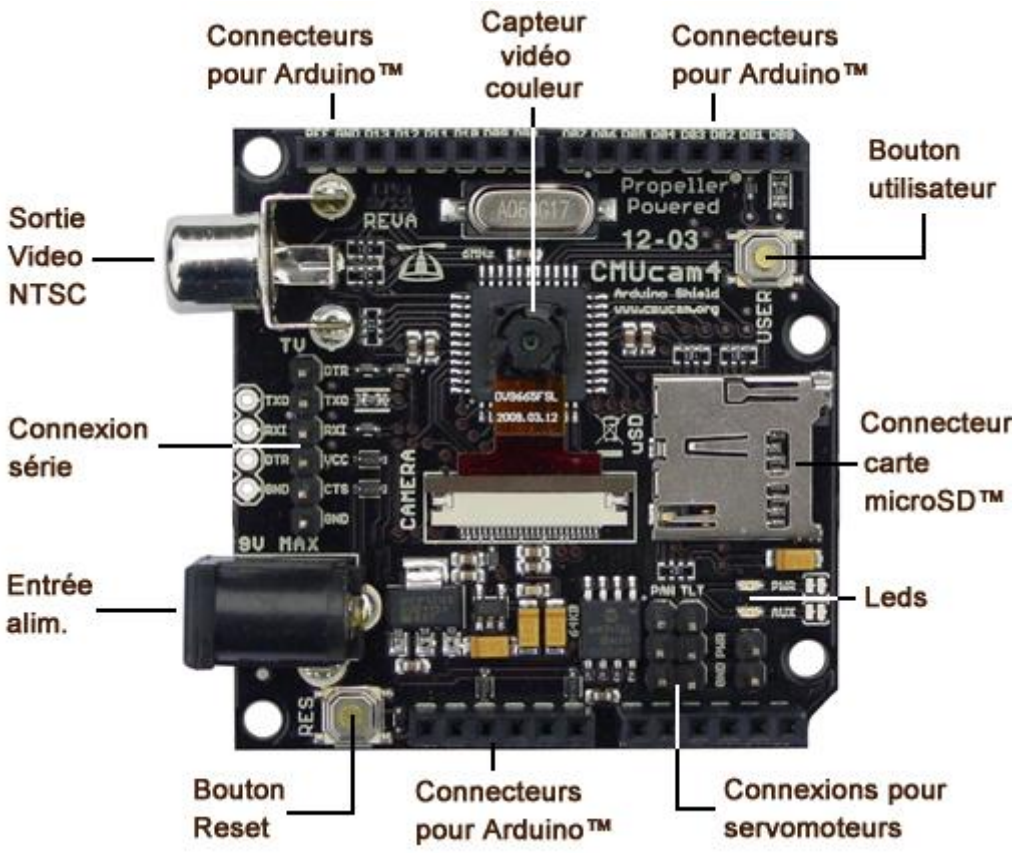


figure 2 : une CMUcam4 et sa légende

- **Comment ça marche ?**

La CMUcam4 n'est utile que si elle est mise en association avec un microcontrôleur (ici Arduino) pour la réalisation de robots mobiles ludiques capables de réagir avec leur environnement afin de pouvoir suivre ou éviter différents objets. Pour ce faire il suffira de lui envoyer via le logiciel Arduino et via son port série une succession de commandes correspondantes aux actions que nous désirons lui faire effectuer.

- **Pourquoi Arduino ?**

Bien que le module "CMUcam4" soit compatible avec n'importe quel microcontrôleur doté d'une liaison série (PIC®, module PICBASIC, CUBLOC ..), le format de son circuit imprimé est identique à celui des platines d'extension "Shield" pour Arduino™. Il est donc possible d'enficher directement le module "CMUcam4" sur un module Arduino, ce qui est trés pratique pour l'experimentation.

figure 3*: *La compatibilité de la carte Arduino et de la CMUcam4

- **La bibliothèque d'interface de CMUCam4/Arduino**

Il y a une bibliothèque spécialement pour la CMUCam4, et on peut la mettre dans le dossier de la bibliothèque d'interface d'Arduino .Cette bibliothèque inclus toutes les commandes connues par CMUCam4. Ici, on présente les fonctions les plus utiles:

Tracking data t: Cette structure contient des membres comme 'int pixels', 'int confidence'.La valeur des pixels peut être utilisée pour déduire la taille ou la distance de la couleur du pixel suivi: plus le pixel est grand plus grande ou plus proche sera la couleur. La valeur de confiance peut être utilisée pour déduire la densité ou la certitude sur la couleur cible du pixel suivi: plus la confiance est grande plus on est sûr que c'est la bonne couleur.

Camera.trackColor: Cette commande provoque pour la CMUcam4 le début du mode "flux" et de l'envoi des paquets de données de type T.Seules les fonctions de getType_DataPackets peuvent être utilisées après l'appel de cette fonction pour obtenir des paquets. La CMUcam4 quitte automatiquement le mode flux si une fonction autre que les fonctions getType_DataPackets est appelé. Cette fonction ne modifie pas les paramètres du suivi de couleur lorsqu'il est appelé. Retours: 0 en cas de succès et une valeur négative en cas d'échec.

Camera.getTypeTDataPacket: Cette commande attend qu'un paquet de données de type T vienne de la CMUcam4 et stocke ce paquet de données dans une structure de données 'CMUcam4_tracking_data_t'.Un flux de paquets de données de type T doit être d'abord lancé

pour que cette fonction n'échoue pas et prolonge le temps pour un T-paquet de données de type.Retours: 0 en cas de succès et une valeur négative en cas d'échec.

c. *Le L293*

Pour commander les deux moteurs du robot, nous devions utiliser un L293. Ce circuit imprimé numérique peut aisément dialoguer avec carte Arduino et la CMUcam4 Ce circuit comprend deux ponts en H eux-mêmes constitués de 4 transistors chacun qui servent à contrôler la polarité des moteurs.

A l'aide de sa documentation, nous avons pu connaitre les branchements à réaliser sur ce composant.

- Les branches 1 et 9 étant des « enable », nous les avons utilisé (pour faire tourner les moteurs) en les reliant aux broches d'Arduino.

- Les branches 4, 5, 12 et 13 ont été reliées à la masse (GROUND).

- Nous avons soudé 4 fils entre L293 et les deux moteurs.

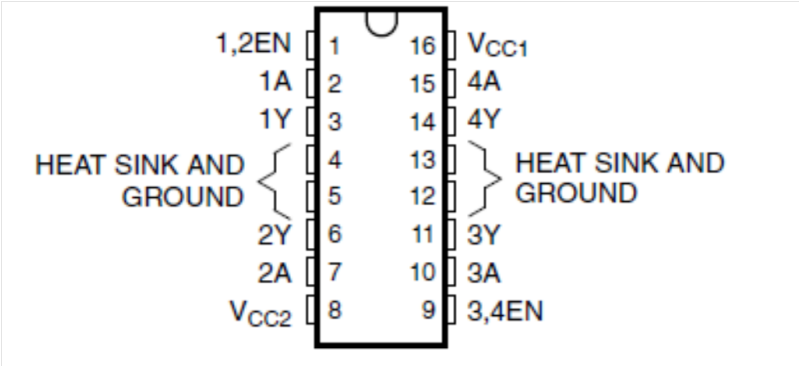


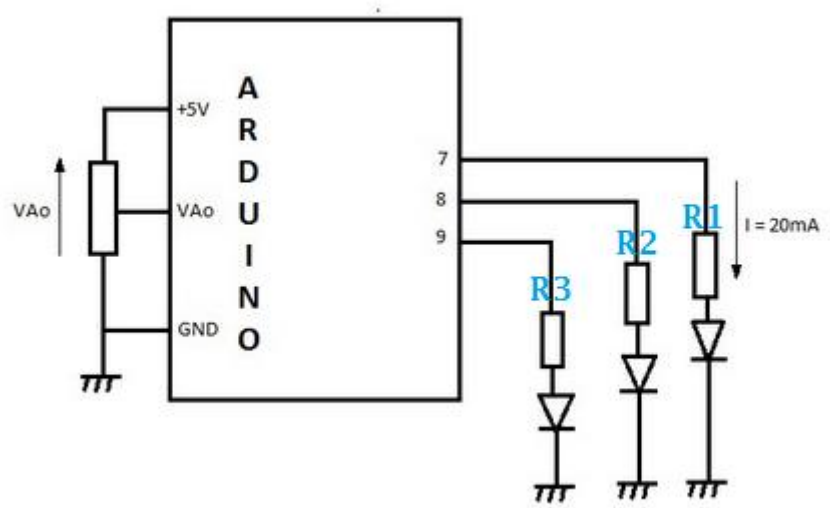
figure 4: Le L293

2. LES PROJETS:

a. ***Premier Projet:*** *Allumer des LED en fonction du potentiel*

Notre premier objectif était de créer un programme pour l'Arduino Uno permettant d'allumer une LED de couleur différente en fonction de la valeur du potentiel, que l'on a fait varier à l'aide d'un potentiomètre.

- Si VAo < 1V => Allumer la Led7
- Si 1V <VAo < 2V => Allumer la Led8
- Si VAo > 3V => Allumer la Led9



De plus, pour retrouver les valeurs de R1, R2 et R3, nous avons utilisé la loi d'Ohm avec I=20mA.

<u>Remarque</u>: sur Arduino, 'Serial.print' permet d'afficher les valeurs actuelles des variables (ici VAo)sur l'écran.

figure 5 : *shéma du montage test des potentiels*

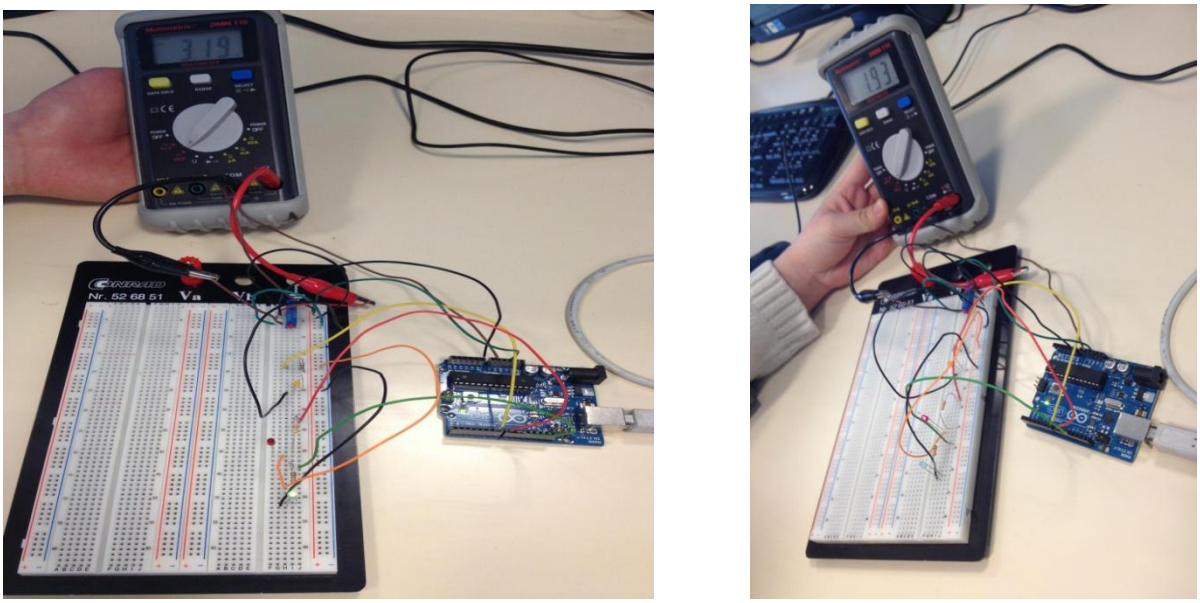


figure 6 : *photographies du montage*

b. **Second Projet:** Track Color

A. Le projet

Afin de mieux comprendre le fonctionnement de la carte CMUCam4, nous avons réalisé une deuxiéme expérience qui consistait à faire en sorte que la LED correspondante à la couleur (rouge , jaune ou verte) vue et détectée par la caméra s'allume. Cette expérience nécessitait d'abord une bonne connaissance de la carte Arduino, du langage Arduino et des différentes composantes de la CMUcam4.

Pour cela, nous avons eu besoin de :

- une CMUcam4
- une carte Arduino
- Le logiciel Arduino
- un câble USB
- 3 LEDS: rouge, verte, jaune
- Un breadboard
- des fils

Ensuite, pour le montage, on a utilisé le même que celui qui a servi pour la premiére expérience. En effet, la majeur partie du temps consacré à ce projet a été utilisée pour réaliser le code Arduino. Pour ce faire,nous avons dû apprendre les bases de ce code, et nous familiariser avec les commandes basiques pour ensuite, s'inspirer d'un modéle proposé par Arduino et l'adapter à notre expérience. Ainsi, nous avons travaillé pendant 3 semaines sur le code, et nous avons fini par proposer 2 programmes [Prog 2, Prog3] que Arduino compilait sans probléme mais qui ne fonctionnait pas dans la pratique pour des raisons que nous expliquerons plus tard.

▪ Qu'est-ce que c'est que 'track color'?

Le track color ou suivi de couleur est la capacité de prendre une image, isoler une couleur particulière et extraire des informations de la localisation d'une région de cette image qui ne contient que cette couleur. Par exemple, supposons que vous avez une photographie qui contient une boule rouge posée sur le sol. Si quelqu'un nous demandait de dessiner une boîte autour de tout ce qui était de la couleur rouge dans l'image, nous aurions assez facilement dessiné un rectangle autour de la balle. C'est l'idée de base du suivi de couleur. Vous n'avez pas besoin de savoir que l'objet est un boule. Il suffisait d'avoir une notion de la couleur rouge afin d'isoler l'objet dans l'image.

▪ Comment le CMUcam4 utilise-t-elle des informations contenues dans une image de la caméra pour réaliser le suivi de la couleur ?

Afin de préciser la couleur, nous devions indiquer par exemple les limites minimales et maximales des composantes RGB de la couleur que l'on souhaite suivre ou rechercher vous devez définir une valeur minimale et maximale pour trois chaînes de couleurs. Chaque couleur

pure est représentée par une valeur de rouge, vert et bleu qui indique la quantité de chaque chaîne pour former cette couleur pure. La partie la plus délicate pour préciser une couleur, c'est la définition des valeurs admissibles pour les trois chaînes de couleur. Puisque la lumière n'est pas parfaitement uniforme et la couleur d'un objet n'est pas parfaitement uniforme, nous devions tenir compte de ces variations. Cependant, nous ne voulions pas trop assouplir ces limites, sinon beaucoup de couleurs indésirables seraient acceptées. Ainsi, dans le cas de la CMUcam4, chaque chaîne de couleur est convertie en un nombre entre 0 et 255, il est donc possible de caractériser chaque couleur par une limite supérieure et inférieure. Si nous avons besoin de deux limites pour définir chacune des trois couleurs, cela signifie que six valeurs peuvent être utilisées pour caractériser l'espace entier de couleurs. Ainsi chaque pixel se trouvant dans le champ de vision du capteur vidéo de la CMUcam4 sera analysé et comparé avec les valeurs prédéfinies, afin de renvoyer les coordonnées du pixel le plus en haut à gauche et le plus en bas à droite du rectangle entourant de plus prés l'objet de la couleur choisie. Ces informations permettront ainsi à la carte Arduino de réagir en conséquence et par exemple commander le robot pour suivre l'objet ou l'éviter en tournant à gauche ou à droite.

B. Les contraintes et les difficultés

Nous avons rencontré quelques difficultés, notamment concernant:

- **L'alimentation** puisque à chaque fois qu'on télécharge le code sur l'Arduino, il faut soit séparer l'Arduino et la carte CMUCam4, soit mettre la CMUCam4 en mode arrêté.
- **Les programmes** qui ne fonctionnaient pas même compilés. Nous nous sommes demandés si le non fonctionnement de nos programmes était dû à la caméra et nous avons compris par la suite que le problème résidait dans la bibliothèque. En effet, c'est dû à un problème de communication en série, car la carte CMUcam4, la carte Arduino et le PC communiquent en même temps via les ports 1 et 2. Pour faire communiquer les deux cartes, on doit connecter la CMUCam4 à un des ports série de l'Arduino. En faisant cela, nous établissons une communication en série et simultanée entre la CMUcam4, l'Arduino et l'ordinateur, ce qui n'est possible que deux à deux. C'est pour cela qu'il faut soir télécharger le code sur la carte Arduino, débrancher le fil permettant la communication avec l'ordinateur pour ensuite exécuter le programme, soit mettre la CMUcam4 en mode arrêté, soit ne pas faire communiquer l'Arduino ,la CMUCam4 et l'ordinateur simultanément sur le même port .

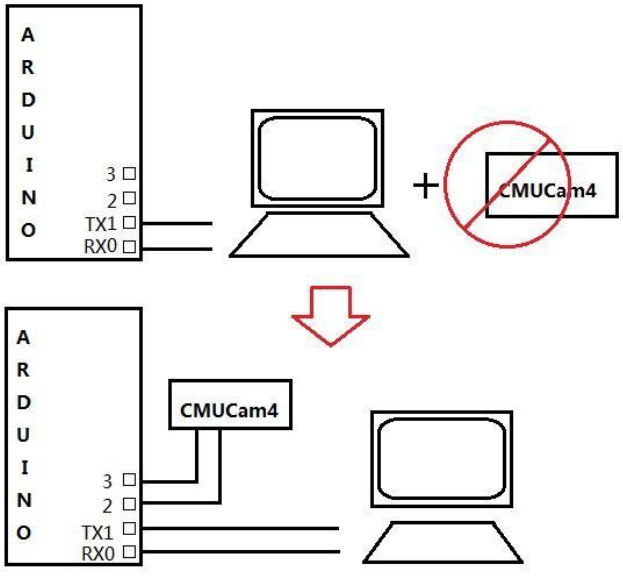


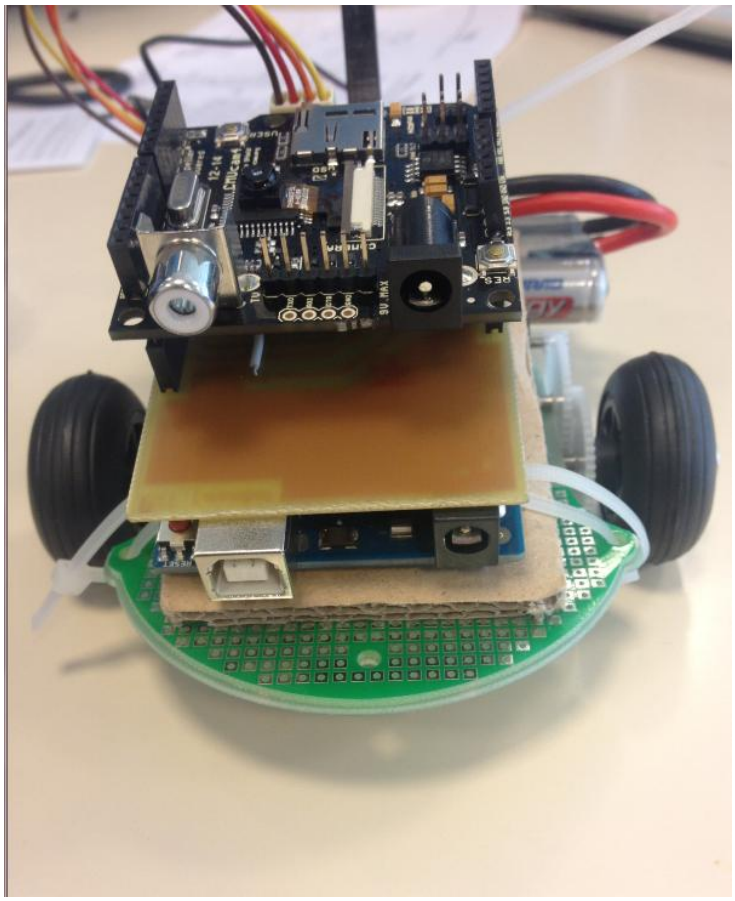
figure 7 : Shéma de connexion de la Cmucam 4

Toutefois, puisque nous avions besoin de la fonction 'serial. Monitor', nous ne pouvions pas utiliser le mode arrêté. Nous avons donc tenté de modifier le fichier 'cmucom4.h' pour changer les ports qui les connectent en modifiant les ports de communication de la CMUcam4 et de la Arduino de 0 et 1 à 2 et .

c. **Troisième projet** : Le robot voiture

▪ Qu'est-ce qu'un robot?

Un robot intelligent est un assemblage complexe de pièces mécaniques et de pièces électroniques, le tout pouvant être piloté par une intelligence artificielle. Il est souvent constitué de trois parties. La première partie est celle des capteurs. La CMUCam4 par exemple, sert à distinguer les différentes couleurs, c'est donc l'équivalent des yeux du robot. La puissance de la collection des informations du robot dépend de la puissance des capteurs. La deuxième partie est celle des circuits électroniques. Le micro-contrôleur est le plus important. Il est comme le CPU de l'ordinateur. L'intelligence du robot vient d'ici. La carte Arduino est un microcontrôleur simple qui suffit de compléter des tâches faciles. La dernière partie est celle de la mécanique, qui est dirigée par le microcontrôleur et qui complète directement les travaux.



Les robots plus compliqués ont souvent un système électronique très puissant. De plus, il est substantiellement intégré à une taille petite. L'industrie microélectronique facilite les développements de la robotique. Les robots deviennent plus intelligents, moins con sommants et moins chers.

figure 8: *Photographie du robot voiture*

Notre troisième projet consistait en la réalisation d'un robot-voiture équipé d'une CMUcam4, et qui ,s'il voit du rouge, du bleu ou du vert , va respectivement s'arrêter, tourner à gauche et tourner à droite.

Pour cela, nous avons eu besoin de

- la CMUcam4
- La carte Arduino + le programme Arduino pour diriger le robot
- Un kit de construction de robot
- L293D Motor Driver

Notre robot est constitué d'une plaque verte qui lui sert de support et est mobile grâce à trois roues, il est composé de 2 moteurs situés au centre, et une batterie située à l'arrière du robot qui assure l'alimentation de la carte Arduino.A l'avant du robot, il y'a dans cet ordre: la CMUcam4 , la L293D Motor Driver et la carte Arduino. Il y a 4 fils sortant de L293D qui doivent être liés avec les 4 pôles des deux moteurs.

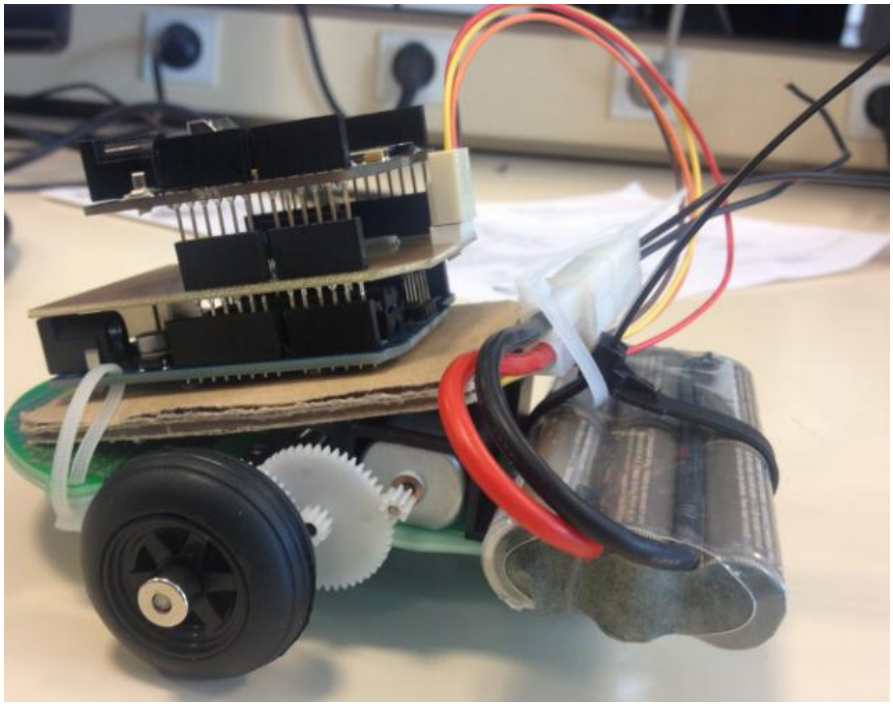


figure 9: *photographie de l'assemblage du robot*

Nous avons tout d'abord étudié le programme que le professeur nous a donné. Ensuite on a modifié les numéros des broches correspondant aux moteurs. Pour cela, nous avons bien lu les documents techniques de la L293D et le schéma des circuits. On a donc pu trouver les numéros des broches correctes correspondant à Arduino. Après, on a monté les parties mécaniques et les circuits. Après le téléchargement du programme et le soudage des broches, nous avons pu le tester en posant des couleurs différentes devant le capteur CMUCam4. Ici, la carte Arduino est alimentée par les batteries à 7 volts.LA L293D dirige les deux moteurs. Pour

tourner à gauche par exemple , il suffit d’arrêter le moteur à gauche, même chose pour tourner à droite.

Grâce à ce projet, nous avons appris la procédure entière de la réalisation d’un robot simple. On a vu clairement comment les trois parties fonctionnent simultanément. Les robots plus compliqués se réalisent selon le même principe. Nous avons également testé chaque partie séparément pour vérifier le bon fonctionnement de tous les composants de notre robot.

CONCLUSIONS ET PERSPECTIVES

Conclusion

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. Globalement, nous nous sommes bien réparti les rôles, chaque membre du groupe a pu travailler de manière efficace.

De plus, le projet a été remis dans les temps, même si nous avons eu besoin de quelques séances supplémentaires.

Perspectives et Améliorations

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. Ainsi, la CMUCAM4 offre de nombreuses possibilités, qu'il pourrait être intéressant d'exploiter plus en détails. On pourrait par exemple réaliser des robots mobiles ludiques capables de réagir avec leur environnement afin de pouvoir suivre ou éviter différents objets. Et plus précisément, localiser l'objet coloré et réagir en conséquence. Et Dans la pratique, un trieur de couleur qui nous permet de cribler du riz ou des fruits serait très intéressant.

III. Bibliographie

[1] http://www.lextronic.fr/P1890-module-de-reconnaissance-video-cmucam4.html (valide à la date du 05/02/2013
[2]*http://www.cmucam.org/* (valide à la date du 31/08/2012)
[3]*http://www.arduino.cc/* (valide à la date du 20/05/2013
[4] *Christian Tavernier, "Arduino: Applications avancées", DUNOD, 2012.*

IV. ANNEXES

PROGRAMME 1 :

```
int sensorPin = A0;                  // select the input pin for the potentiometer
int jaune = 7;                       // select the pin for the LED
int rouge = 8;                      // variable to store the value coming from the sensor
int vert = 9;
void setup() {
  // declare the ledPin as an OUTPUT:
  Serial.begin(9600);
  pinMode(jaune, OUTPUT);
  pinMode(rouge, OUTPUT);
  pinMode(vert, OUTPUT);
}
void loop() {
   long sensorValue = analogRead(sensorPin);     // read the value from the sensor
  delay(1000);
   Serial.print(sensorValue);                          //float volt= (sensorValue/1023)*5;
 if (sensorValue<205)
 {
   digitalWrite(jaune, HIGH);                        // Illuminate the yellow led
  digitalWrite(rouge, LOW);
   digitalWrite(vert, LOW);
 }
 else if(sensorValue<410) {
   digitalWrite(rouge, HIGH);                        // Illuminate the red led
   digitalWrite(jaune, LOW);
   digitalWrite(vert, LOW);
 }
 else if(sensorValue>615){

   digitalWrite(vert, HIGH);                         // Illuminate the green led
   digitalWrite(jaune, LOW);
   digitalWrite(rouge, LOW);
 }
}
```

Programme ***2 :***

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


int LED_jaune = 7;                    // The pin corresponding to the yellow led
int LED_rouge = 8;                    // The pin corresponding to the red led
int LED_vert = 9;                     // The pin corresponding to the green led
int red;                              // 0 if red is found, negative otherwise
int green;                            // 0 if green is found, negative otherwise
int yellow;                           // 0 if yellow is found, negative otherwise
int mode;

const int RRED_MIN = 170;             // To define the RGB channels for red, 6 parameters
const int RRED_MAX = 255;
const int RGREEN_MIN = 0;
const int RGREEN_MAX =80;
const int RBLUE_MIN =0;
const int RBLUE_MAX =80;

const int GRED_MIN = 0;               // To define the RGB channels for green, 6 parameters
const int GRED_MAX = 100;
const int GGREEN_MIN =130;
const int GGREEN_MAX =255;
const int GBLUE_MIN =30;
const int GBLUE_MAX =100;

const int YRED_MIN = 180;          // To define the RGB channels for yellow, 6 parameters
const int YRED_MAX = 255;
const int YGREEN_MIN =200;
const int YGREEN_MAX =255;
const int YBLUE_MIN =0;
const int YBLUE_MAX =100;


CMUcam4 camera(CMUCOM4_SERIAL);
void setup(){
 camera.begin();
 pinMode(LED_jaune, OUTPUT);
 pinMode(LED_rouge, OUTPUT);
 pinMode(LED_vert, OUTPUT);

 camera.LEDOn(CMUCAM4_LED_ON);
```

```
 delay(1000);
}

void loop(){

 CMUcam4_tracking_data_t thedata;       /* To define a data structure for storing the tracked information*/

 if(mode==0) {

camera.setTrackingParameters(RRED_MIN,RRED_MAX,RGREEN_MIN,RGREEN_MAX,RBLUE_MIN,RBLUE_MAX);

camera.trackColor(RRED_MIN,RRED_MAX,RGREEN_MIN,RGREEN_MAX,RBLUE_MIN,RBLUE_MAX);                                                  // tracking red
  red=camera.getTypeTDataPacket(&thedata);
     if(red==0){                                                  // if red is found, illuminate the red led
       digitalWrite(LED_jaune,LOW);
       digitalWrite(LED_rouge,HIGH);
       digitalWrite(LED_vert,LOW);
            }

     else {                                                  // if red is not found, enter mode 1
       mode=1;
          }
    }

  else if(mode==1) {

camera.setTrackingParameters(YRED_MIN,YRED_MAX,YGREEN_MIN,YGREEN_MAX,YBLUE_MIN,YBLUE_MAX);

camera.trackColor(YRED_MIN,YRED_MAX,YGREEN_MIN,YGREEN_MAX,YBLUE_MIN,YBLUE_MAX);                                                  // tracking yellow
   yellow=camera.getTypeTDataPacket(&thedata);
        if(yellow==0){                                                  // if yellow is found, illuminate the red led
          digitalWrite(LED_jaune,HIGH);
          digitalWrite(LED_rouge,LOW);
          digitalWrite(LED_vert,LOW);
                }
        else {                                                  // if yellow is not found, enter mode 2
          mode=2;
            }
      }
     else if(mode==2){
```

```
camera.setTrackingParameters(GRED_MIN,GRED_MAX,GGREEN_MIN,GGREEN_MAX,GB
LUE_MIN,GBLUE_MAX);

camera.trackColor(GRED_MIN,GRED_MAX,GGREEN_MIN,GGREEN_MAX,GBLUE_MIN,
GBLUE_MAX);                                                    // tracking green
        green=camera.getTypeTDataPacket(&thedata);
           if(green==0){                                       // if green is found, illuminate the red
led
                digitalWrite(LED_jaune,LOW);
                digitalWrite(LED_rouge,LOW);
                digitalWrite(LED_vert,HIGH);
                     }
           else {                                              // if green is not found, enter mode 0
                 mode=0;
                }
         }

  Serial.print(red);                                          // to show is information got in the monitor
  delay(500);
  Serial.print(yellow);
  delay(500);
  Serial.print(green);
  delay(500);

}
```

Programme 3:

```
/*
int       mx
       The middle mass X position of the tracked pixels - [0:159].
int       my
       The middle mass Y position of the tracked pixels - [0:119].
int       x1
       The upper left X coordinate of the bounding box - [0:159].
int       y1
       The upper left Y coordinate of the bounding box - [0:119].
int       x2
       The bottom right X coordinate of the bounding box - [X1:159].
int       y2
       The bottom right Y coordinate of the bounding box - [Y1:119].
int       pixels
       The percentage of tracked pixels in the image - [0:255].
int       confidence
       The density of tracked pixels in the image - [0:255].
```

```
 Plus pixels est grand et plus l'objet recherch?? est proche
*/

#include <CMUcam4.h>
#include <CMUcom4.h>

#define COULEUR_CHERCHEE_MIN 20
#define COULEUR_CHERCHEE_MAX 255
#define COULEUR_IGNOREE_MIN  0
#define COULEUR_IGNOREE_MAX  50
#define PIXELS               50
#define CONFIDENCE           50 // sorte d'interval de confiance
#define TROP_PROCHE          251
#define VITESSE_MAXIMALE     255
#define VITESSE_ELEVEE       200
#define VITESSE_MOYENNE      130
#define VITESSE_BASSE        30
#define LED_BLINK            5 // 5 Hz
#define WAIT_TIME            5000 // 5 seconds


int enableG = 13, poleAG = 6, poleBG = 5, enableD = 8, poleAD = 10, poleBD = 11;  //
voir schema

CMUcam4 camera(CMUCOM4_SERIAL);
inline void digital_allumer_mot(int &mot){
  digitalWrite(mot, HIGH);
}

inline void digital_eteindre_mot(int &mot){
  digitalWrite(mot, LOW);
}

inline void analog_set_mot(int &mot, const int &vitesse){
  analogWrite(mot, vitesse);
}


inline void arreter(){
 analog_set_mot(poleAG, 255);
 analog_set_mot(poleBG, 255);
 analog_set_mot(poleAD, 255);
 analog_set_mot(poleBD, 255);
 digital_allumer_mot(enableG);
 digital_allumer_mot(enableD);
```

```
}
inline void avancer(const int vitesse){
  digital_allumer_mot(enableG);
  digital_allumer_mot(enableD);
  analog_set_mot(poleBG, 0);
  analog_set_mot(poleBD, 0);
  analog_set_mot(poleAG, vitesse);
  analog_set_mot(poleAD, vitesse);
}
inline void reculer(const int vitesse){
  analog_set_mot(poleAG, 0);
  analog_set_mot(poleAD, 0);
  analog_set_mot(poleBG, vitesse);
  analog_set_mot(poleBD, vitesse);
}
inline void tourner_droite(const int vitesse){
  analog_set_mot(poleAD, 0);
  analog_set_mot(poleBG, 0);
  analog_set_mot(poleBD, vitesse);
  analog_set_mot(poleAG, vitesse);

}
inline void tourner_gauche(const int vitesse){
  analog_set_mot(poleAG, 0);
  analog_set_mot(poleAD, 0);
  analog_set_mot(poleBD, 0);
  digital_eteindre_mot(enableG);
  digital_eteindre_mot(enableD);
  analog_set_mot(poleBG, vitesse);
}

inline void ordre_rouge(const CMUcam4_tracking_data_t &rouge){
  if(rouge.pixels > PIXELS && rouge.confidence > CONFIDENCE){ // on voit du rouge
    if(rouge.pixels > TROP_PROCHE){
      arreter();
    }
    else{
      avancer(VITESSE_MAXIMALE);
    }
  }
  else{
     arreter();
  }
}

inline void ordre_vert(const CMUcam4_tracking_data_t &vert){
```

```
  if(vert.pixels > PIXELS && vert.confidence > CONFIDENCE){ // on voit du vert
     tourner_gauche(VITESSE_MAXIMALE);
  }
  else{
     arreter();
  }
}

inline void ordre_bleu(const CMUcam4_tracking_data_t &bleu){
  if(bleu.pixels > PIXELS && bleu.confidence > CONFIDENCE){ // on voit du bleu
    tourner_gauche(VITESSE_MAXIMALE);
    delay(1500);
    avancer(VITESSE_MAXIMALE);
    delay(2000);
  }
  else{
    arreter();
  }
}

inline char priorite(const CMUcam4_tracking_data_t &rouge, const
CMUcam4_tracking_data_t &vert, const CMUcam4_tracking_data_t &bleu){
  unsigned r = rouge.pixels + rouge.confidence;
  unsigned v = vert.pixels + vert.confidence;
  unsigned b = bleu.pixels + bleu.confidence;
  if(r >= v && r >= b){
   return 'r';
  }
  else if(v >= r && v >= b){
   return 'v';
  }
  else{
   return 'b';
  }
}

inline void traiter_coordonnees(const CMUcam4_tracking_data_t &rouge, const
CMUcam4_tracking_data_t &vert, const CMUcam4_tracking_data_t &bleu){
  char prio = priorite(rouge, vert, bleu);
  if(prio == 'r'){
    ordre_rouge(rouge);
  }
  else if(prio == 'v'){
    ordre_vert(vert);
  }
  else{
```

```
    ordre_bleu(bleu);
  }
}


void setup(){
  pinMode(enableG, OUTPUT);
  pinMode(enableD, OUTPUT);
  pinMode(poleAG, OUTPUT);
  pinMode(poleBG, OUTPUT);
  pinMode(poleAD, OUTPUT);
  pinMode(poleBD, OUTPUT);

  arreter();

  camera.begin();

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


  camera.autoGainControl(false);
  camera.autoWhiteBalance(false);

  camera.LEDOn(CMUCAM4_LED_ON);



}


void loop(){


  CMUcam4_tracking_data_t rouge;
  CMUcam4_tracking_data_t bleu;
  CMUcam4_tracking_data_t vert;
  camera.pollMode(true);

  for(;;){
    camera.trackColor(COULEUR_CHERCHEE_MIN, COULEUR_CHERCHEE_MAX,
COULEUR_IGNOREE_MIN, COULEUR_IGNOREE_MAX, COULEUR_IGNOREE_MIN,
COULEUR_IGNOREE_MAX);
    camera.getTypeTDataPacket(&rouge);
    camera.trackColor(COULEUR_IGNOREE_MIN, COULEUR_IGNOREE_MAX,
```

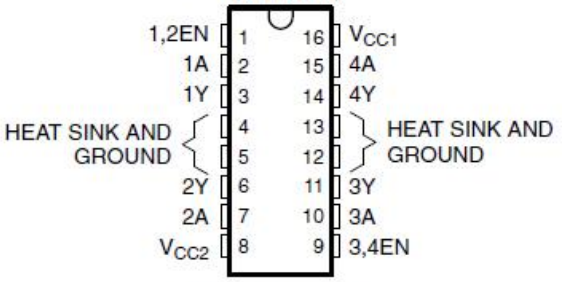
```
COULEUR_CHERCHEE_MIN,                    COULEUR_CHERCHEE_MAX,
COULEUR_IGNOREE_MIN,
COULEUR_IGNOREE_MAX);
   camera.getTypeTDataPacket(&vert);
   camera.trackColor(COULEUR_IGNOREE_MIN,    COULEUR_IGNOREE_MAX,
COULEUR_IGNOREE_MIN,
COULEUR_IGNOREE_MAX,                     COULEUR_CHERCHEE_MIN,
COULEUR_CHERCHEE_MAX);
   camera.getTypeTDataPacket(&bleu);
   traiter_coordonnees(rouge, vert, bleu);
  }

}
```

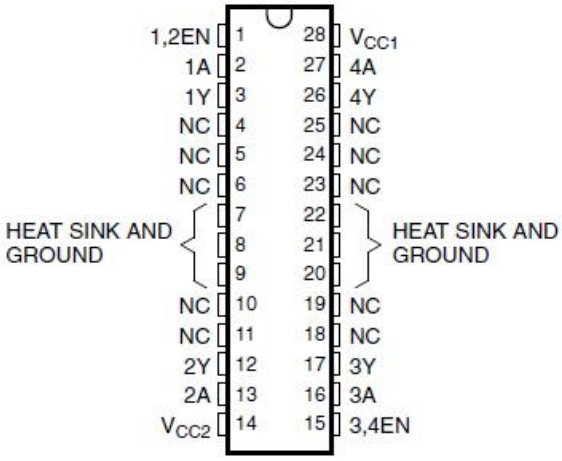
IV. DOCUMENTATION TECHNIQUE:

- **Featuring Unitrode L293 and L293D Products Now From Texas Instruments**
- **Wide Supply-Voltage Range: 4.5 V to 36 V**
- **Separate Input-Logic Supply**
- **Internal ESD Protection**
- **Thermal Shutdown**
- **High-Noise-Immunity Inputs**
- **Functionally Similar to SGS L293 and SGS L293D**
- **Output Current 1 A Per Channel (600 mA for L293D)**
- **Peak Output Current 2 A Per Channel (1.2 A for L293D)**
- **Output Clamp Diodes for Inductive Transient Suppression (L293D)**

L293 . . . N OR NE PACKAGE
L293D . . . NE PACKAGE
(TOP VIEW)



L293 . . . DWP PACKAGE
(TOP VIEW)



description/ordering information

The L293 and L293D are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

All inputs are TTL compatible. Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled, and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled, and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

ORDERING INFORMATION

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	HSOP (DWP)	Tube of 20	L293DWP	L293DWP
	PDIP (N)	Tube of 25	L293N	L293N
	PDIP (NE)	Tube of 25	L293NE	L293NE
		Tube of 25	L293DNE	L293DNE

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

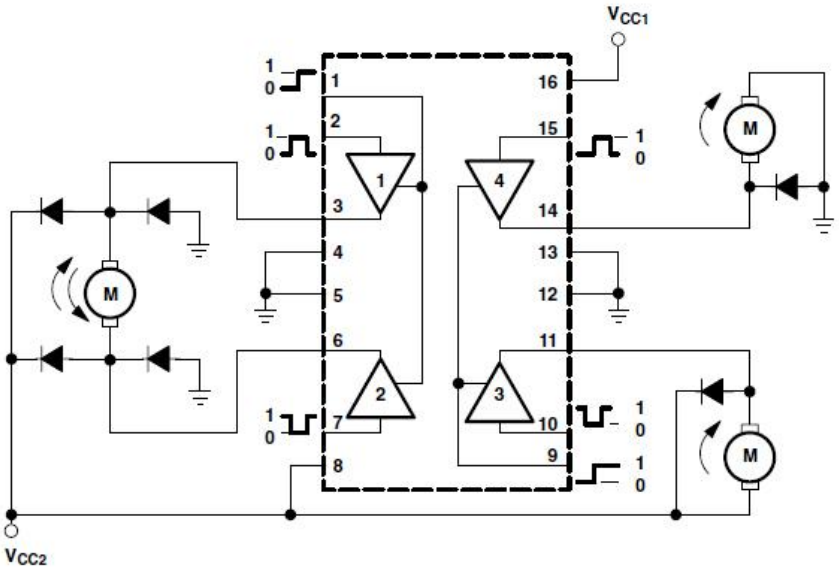
description/ordering information (continued)

On the L293, external high-speed output clamp diodes should be used for inductive transient suppression.

A V_{CC1} terminal, separate from V_{CC2}, is provided for the logic inputs to minimize device power dissipation.

The L293and L293D are characterized for operation from 0°C to 70°C.

block diagram



NOTE: Output diodes are internal in L293D.

FUNCTION TABLE
(each driver)

INPUTS†		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = high level, L = low level, X = irrelevant,
Z = high impedance (off)

† In the thermal shutdown mode, the output is in the high-impedance state, regardless of the input levels.