Le rapport intermédiaire de projet fin d'étude

Labyrinthe à bille autonome

Tuteur : M.Blaise CONRARD

Étudiante : YAN Xuelu

Sommaire

Introduction

L'objectif du projet est de créer un labyrinthe dans lequel une bille doit suivre une trajectoire de la surface. L'un des systèmes de contrôle assure le mouvement de la bille grâce au contrôle de la pente de la surface.

Pour cette réalisation, il faut d'abord créer de l'équipement à la découpe laser. Et ensuite on utilise d'un arduino comme système de traitements pour commander le système. Pour contrôler le trajectoire de la bille, il faut détecter ses coordonnées tout le temps. Donc on utilise d'une caméra comme capteur de position Enfin on utilise de servo-moteurs pour le contrôle de l'inclinaison du plan et du mouvement de la bille.

Dans la première moitié de mes études, j'ai créé l'équipement labyrinthe et déterminer le schéma mécanique. Puis j'ai testé le caméra et les moteurs avec l'Arduino. Dans ce cas, j'ai dessiné le schéma électronique dans le logiciel Fritzing.
Pour la seconde moitié du travail, je vais finir la partie électronique et après je vais travailler sur la partie programme. Il faut utiliser le contrôle PI pour contrôler servos. Enfin, je vais faire la fixation des moteurs et faire l'assemblage de tous équipement

Présentation du projet

Cahier des charges

Contexte

C'est le projet de IMA5. Ce projet vise à réaliser un démonstration où une bille parcourt un labyrinthe. Il doit se déplacer en douceur vers la position désignée en fonction de l'itinéraire spécifié. Une fois le projet terminé, il sera exposé lors de la journée portes ouvertes.

Objectif

L'objectif du projet est de créer un labyrinthe dans lequel une bille doit suivre une trajectoire de la surface. L'un des systèmes de contrôle assure le mouvement de la bille grâce au contrôle de la pente de la surface.

Fonctionnement

1. La bille se déplace en douceur

La bille doit éviter les obstacles et se déplacer en douceur dans le labyrinthe.

2. Déterminer les coordonnées de la bille

Pendant le mouvement de la bille, le système doit surveiller ses informations de position en temps réel et obtenir les coordonnées de la bille.

3. La bille se déplace selon la trajectoire spécifiée

Nous utilisons le programme pour définir la trajectoire de la bille. La bille peut répéter le mouvement en fonction de la trajectoire spécifié.

Ressources

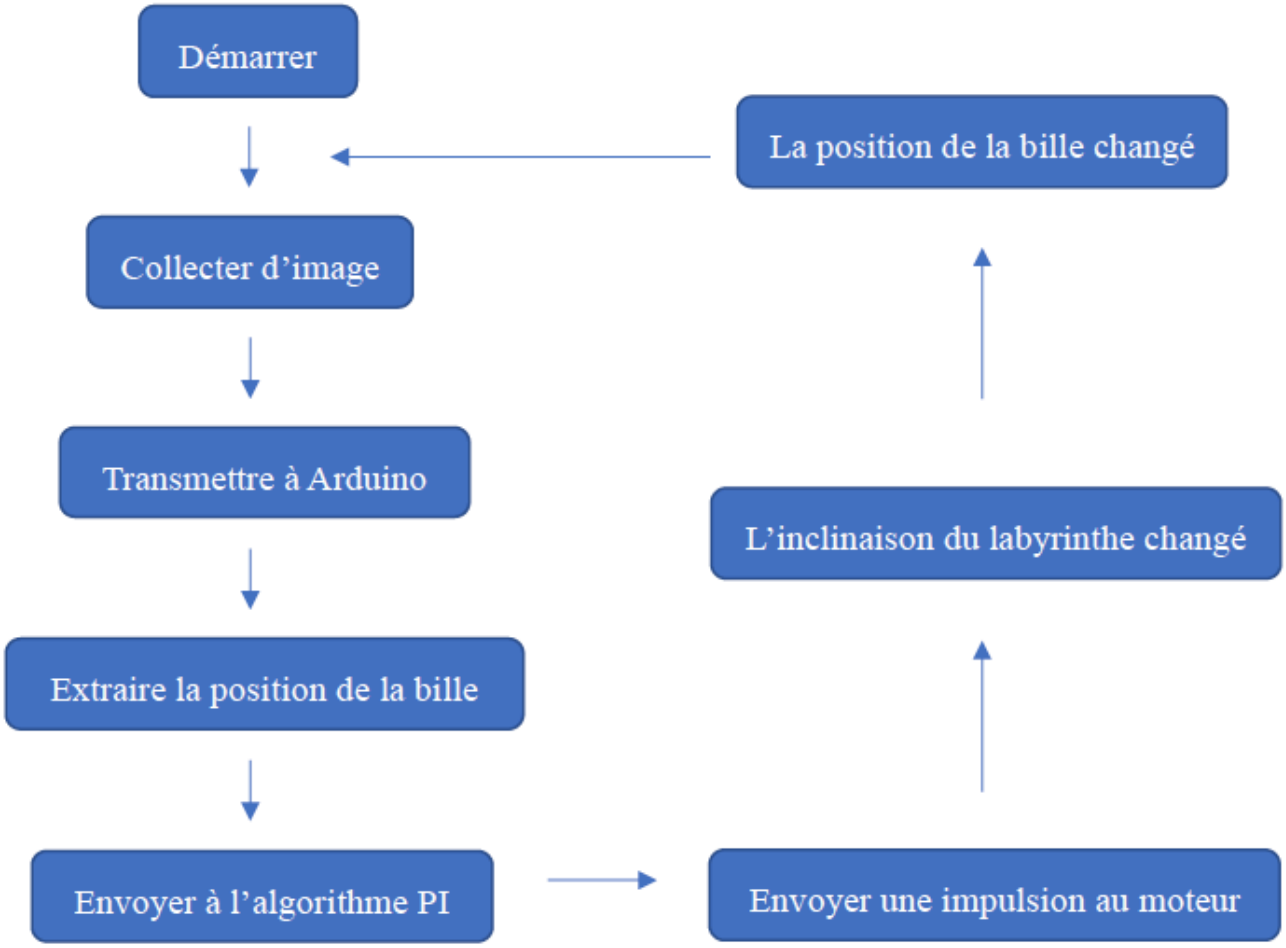
Sous la direction de mon tuteur Blaise Conrard, ce système est réalisé par moi personnellement. Le matériel informatique, les composants et le support technique requis dans ce système sont fournis par l'école.

Delai

Il est prévu de mettre en place le fonctionnement et l'utilisation du système au début du mois de février 2020. La date de livraison est fin février 2020.

Diagramme de processus

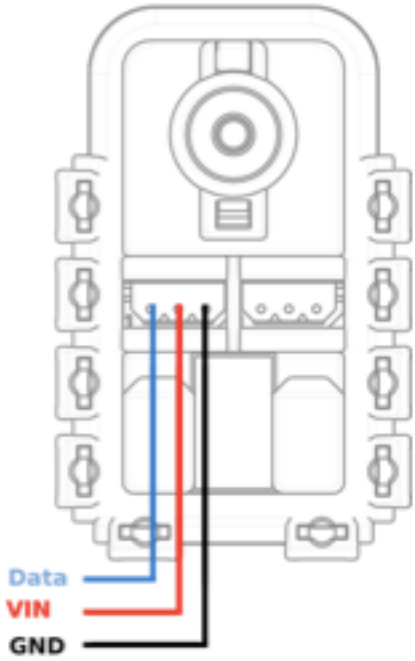
Avant de commencer ce projet j'ai cherché les informations sur ce projet et enfin j'ai dessiné le diagramme de processus.



Après le démarrage, la caméra commence à acquérir des images en temps réel. Les informations de coordonnées collectées par la caméra seront transmises à Arduino pour traitement. Les informations de position traitées sont ensuite transmises à un algorithme PI, on l'utilise pour contrôler la rotation du servomoteur. Par conséquent, l'inclinaison de la plaque de labyrinthe et le changement de la position de la petite bille sont contrôlés, de sorte que la bille peut répéter le mouvement en fonction de la trajectoire spécifié.

Choix du matériel

Servomoteurs



Principe :

Le système servo est un système de contrôle automatique qui permet à une quantité de sortie contrôlée d'un objet (comme la position, l'orientation, le statut...) de suivre un changement arbitraire d'une cible d'entrée. Le servomoteur est principalement positionné par impulsions. Lorsque le servomoteur reçoit une impulsion, il fait pivoter l'angle correspondant à une impulsion pour obtenir un déplacement. Le servo-moteur lui-même a pour fonction d'émettre une impulsion. Ainsi, chaque fois que le servo-moteur tourne d'un angle, un nombre correspondant d'impulsions est émis. De cette manière, l'impulsion reçue par le servomoteur forme un écho ou une boucle fermée. En conséquence, le système sait combien d'impulsions ont été envoyées au servomoteur et combien d'impulsions ont été reçues. De cette manière, la rotation du moteur peut être contrôlée avec une très grande précision, ce qui permet d'obtenir un positionnement précis pouvant atteindre 0,001 mm.

Avantage:

1. Précision: réalise un contrôle en boucle fermée de la position, de la vitesse et du couple, résout le problème du moteur pas à pas en déséquilibre;

2, Vitesse: la performance à haute vitesse est bonne, la vitesse nominale générale peut atteindre 2000 ~ 3000 tr / min;

3. Adaptabilité: il possède une forte capacité anti-surcharge et peut résister à trois fois le couple nominal. Il est particulièrement adapté aux occasions avec des fluctuations de charge transitoires et un démarrage rapide.

4. Stabilité: Il fonctionne sans heurts à basse vitesse et ne produit pas d'opération pas à pas similaire à celle d'un moteur pas à pas lorsqu'il tourne à basse vitesse. Convient pour les occasions avec des exigences de réponse à grande vitesse;

5. Rapidité d'exécution: La dynamique d'accélération et de décélération motrices est courte, généralement en quelques dizaines de millisecondes.

6. Confort: la chaleur et le bruit sont considérablement réduits.

Caméra Pixy

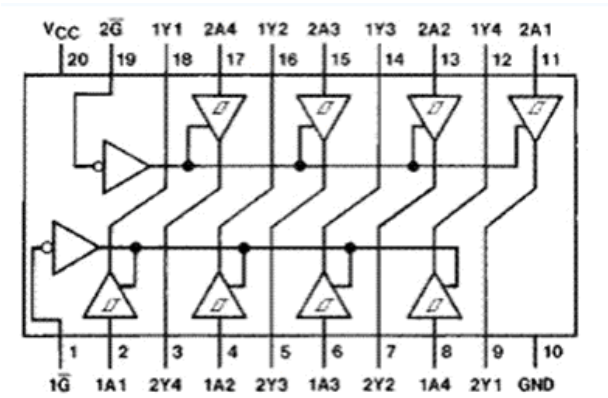
Fonctions :

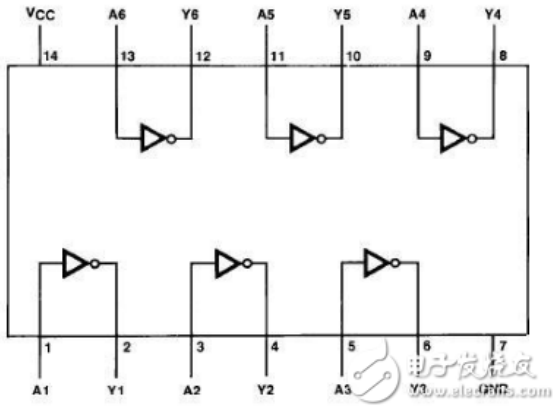
Pixy est un capteur de reconnaissance d'image open source qui prend en charge la reconnaissance de couleur multi-objet et multicolore et prend en charge jusqu'à 7 couleurs. Vous pouvez lui dire la couleur que vous voulez et lui apprendre à trouver quelque chose. Pixy prend en charge plusieurs méthodes de communication, telles que SPI, I2C, etc., qui peuvent être directement connectées à la carte de commande Arduino. Installez-le sur votre robot et ajoutez une paire d'yeux à votre petit robot. Il est équipé d'un capteur d'image doté d'un matériel puissant pouvant être utilisé avec un PC pour suivre et analyser des données multicolores. Pixy peut communiquer directement avec l'Arduino. Il envoie des informations de bloc à l'Arduino à 1 Mbit / s, ce qui signifie que le Pixy peut envoyer plus de 6 000 objets identifiés par seconde ou 135 objets identifiés par frame (Pixy peut traiter 50 images par seconde).

Communications :

Pas besoin de jouer avec de minuscules fils - Pixy est livré avec un câble spécial pour se connecter directement à un Arduino et un câble USB pour se connecter à un Raspberry Pi, afin que vous puissiez commencer rapidement. Pixy possède plusieurs interfaces (SPI, I2C, UART et USB) et des communications simples, de sorte que votre contrôleur choisi parle à Pixy en peu de temps.

74LS244N & 74HC04N





74LS244N :

Description :

74LS244N est un buffer à huit voies à trois états.
Ces tampons octaux et pilotes de ligne sont spécialement conçus pour améliorer à la fois les performances et la densité des pilotes d'adresse mémoire à trois états, des pilotes d'horloge et des récepteurs et transmetteurs orientés bus. Le concepteur dispose d'un choix de combinaisons sélectionnées de sorties inversées et non inversées, d'entrées symétriques de contrôle de sortie actif (G) et d'entrées de contrôle de sortie complémentaires (G et G). Ces appareils présentent un fort fan-out, un fan-in amélioré et une marge de bruit de 400 mV. Les appareils SN74LS et SN74S peuvent être utilisés pour conduire des lignes terminées vers 133 Ohm.

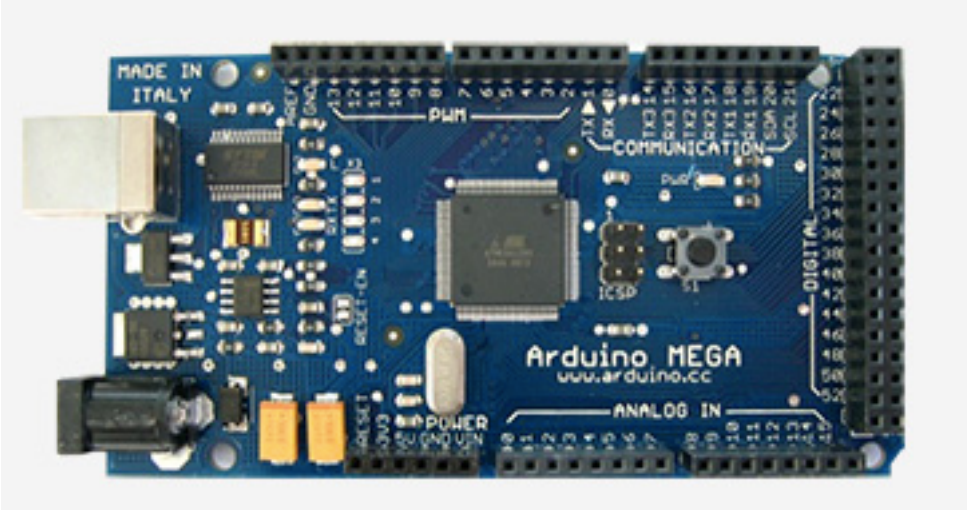
Fonctions :

Dans ce projet, le composant 74LS244N peut établir une communication entre le servomoteur et l'Arduino. Après la connexion par lui, on peut contrôler la rotation du moteur directement via Arduino.

74HC04N :

Les appareils 74HC04N contiennent six inverseurs indépendants. Ils exécutent la fonction booléenne Y = $\bar{A}$ en logique positive.
Ce composant est pour inverser la commande.

Arduino Mega



Présentation :

L'Arduino Mega est une carte microcontrôleur basée sur l'ATmega1280 (fiche technique). Il dispose de 54 broches d'entrée / sortie numériques (dont 14 peuvent être utilisées comme sorties PWM), 16 entrées analogiques, 4 UART (ports série matériels), un oscillateur à cristal de 16 MHz, une connexion USB, une prise d'alimentation, un en-tête ICSP, et un bouton de réinitialisation. Il contient tout le nécessaire pour supporter le microcontrôleur; connectez-le simplement à un ordinateur avec un câble USB ou alimentez-le avec un adaptateur AC-DC ou une batterie pour commencer.

Communication :

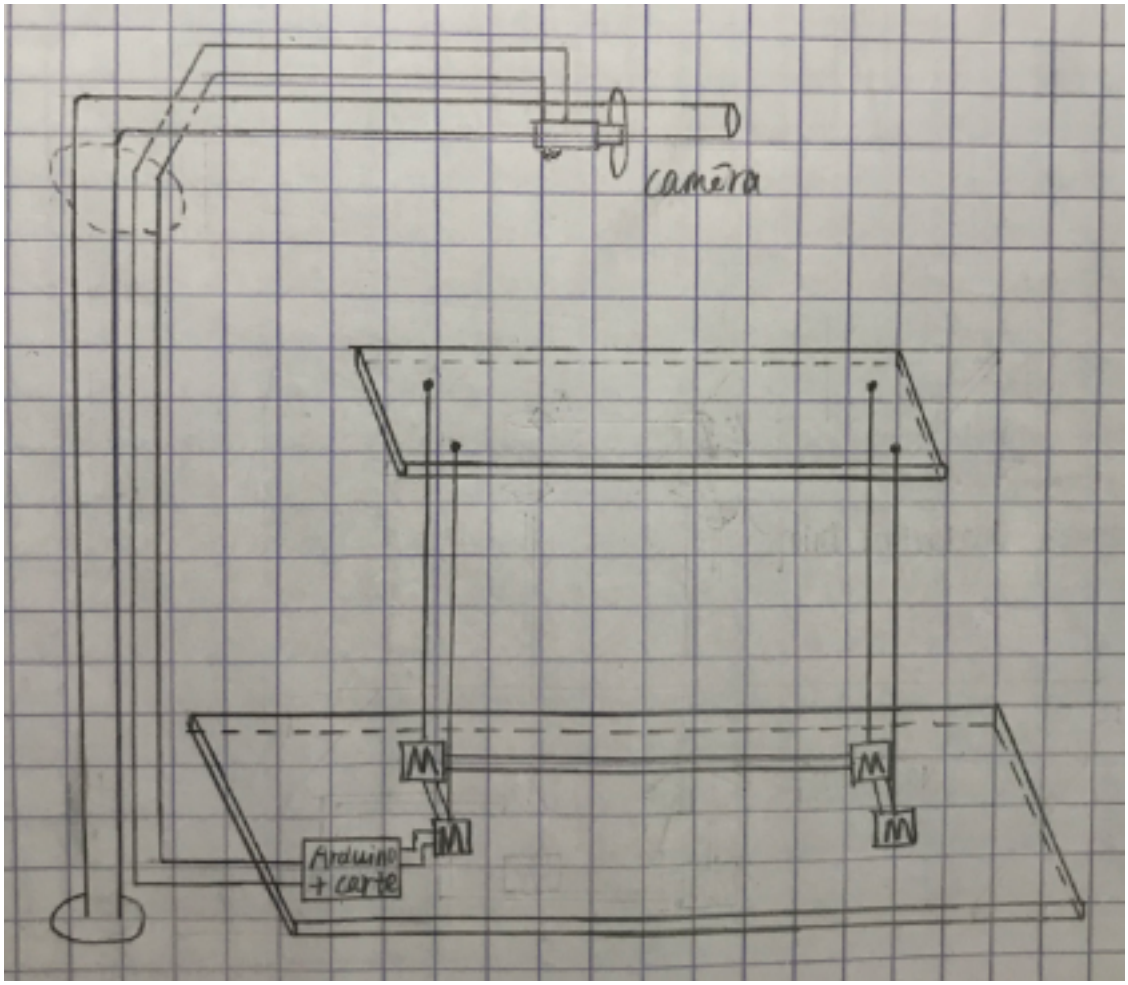
L'Arduino Mega dispose d'un certain nombre d'installations pour communiquer avec un ordinateur, un autre Arduino ou d'autres microcontrôleurs. L'ATmega1280 fournit quatre UART matériels pour la communication série TTL (5V). Un FTDI FT232RL sur la carte canalise l'un d'entre eux via USB et les pilotes FTDI (inclus avec le logiciel Arduino) fournissent un port de communication virtuel aux logiciels de l'ordinateur. Le logiciel Arduino comprend un moniteur série qui permet d'envoyer des données textuelles simples vers et depuis la carte Arduino. Les LED RX et TX de la carte clignotent lorsque des données sont transmises via la puce FTDI et la connexion USB à l'ordinateur (mais pas pour la communication série sur les broches 0 et 1).

Réalisation du projet

Partie mécanique

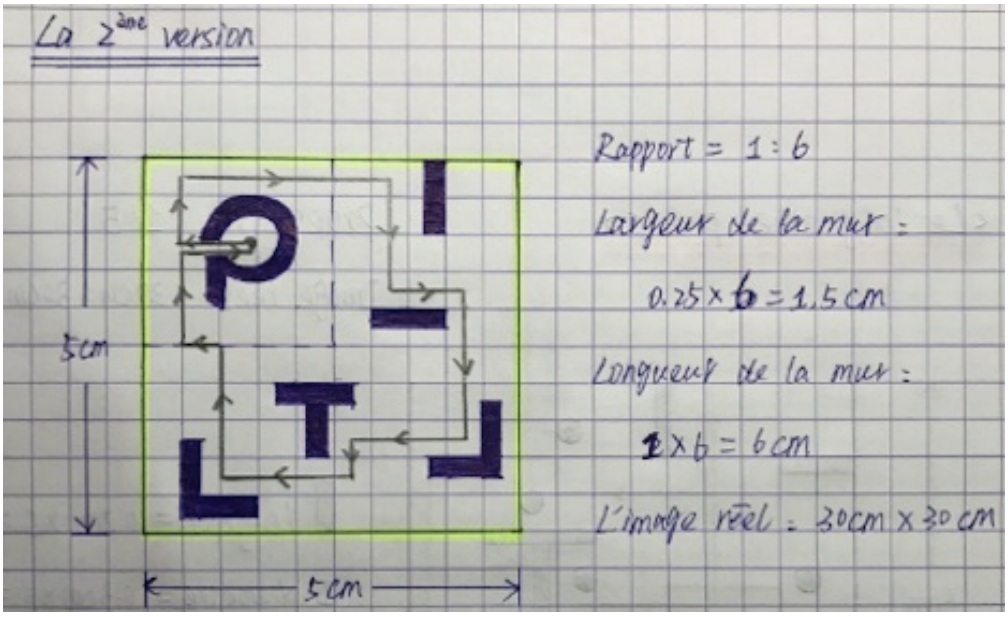
Design du schéma mécanique

Tout d'abord, il y a quatre moteurs en bas pour contrôler la direction. Fixez une tige au culbuteur de chaque moteur. La rotation du moteur fait bouger la tige d'haut en bas pour contrôler l'inclinaison de la planche. J'utilise 4 moteurs pour stabiliser la structure, et parce que la bille se déplace en ligne droite, chaque fois que l'inclinaison de la planche doit être contrôlée par deux moteurs du même côté. Enfin, il faut fixer la caméra sur une tige utilitaire directement au-dessus de la carte. La caméra et les pièces de connexion Arduino sont également fixées à la tige. Cela garantira la stabilité de la caméra.

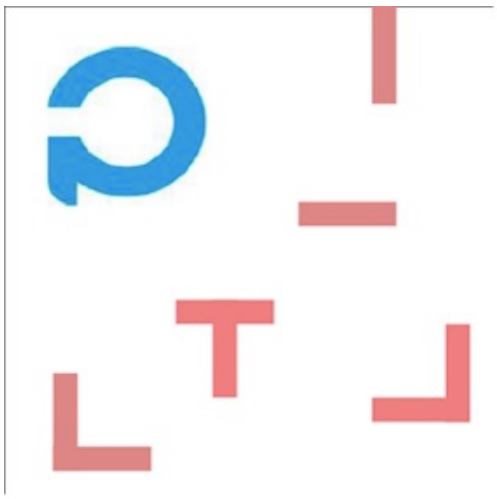


Le labyrinthe

Tout d'abord, j'ai dessiné le labyrinthe. Dans ma conception, je règle la taille du labyrinthe (plache du haut) à 30 cm x 30 cm. La taille de la planche du bas est de 40 cm x 40 cm. Je mets quelques obstacles, le largeur est 1.5cm et la longueur est 6cm. J'ai dessiné la trajectoire de la bille. La bille va partir du symbole. Et elle va répéter le mouvement en fonction de la trajectoire spécifié.



Après j'ai déterminé le dessin en papier, je l'ai crée dans le logiciel 'Inkscape'. Ensuite, j'ai réservé la machine 'Découpeuse Laser' pour faire ma planche de labyrinthe. Avant de couper, je dois ajuster divers paramètres. J'ai importé le fichier conçus à l'ordinateur. Avant de couper, je dois ajuster divers paramètres. Le rouge représente la coupe et le noir représente le gravure. Donc, j'ai défini la bordure en rouge , le chemin du milieu et la zone du logo en noir. Enfin, j'ai obtenu la planche du labyrinthe.

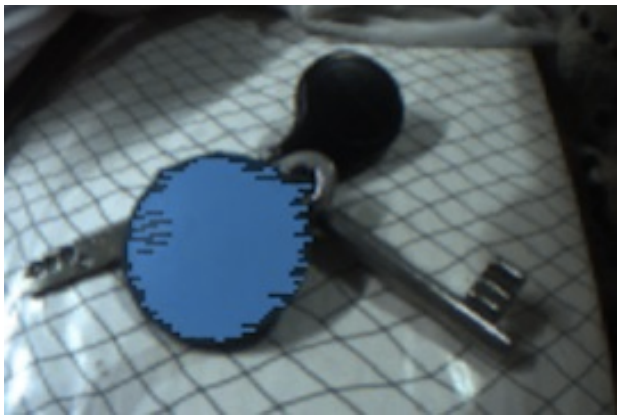


Partie de caméra

Avant de tester la caméra, il faut installer le logiciel ‘PixyMon’.

Détection d’objets

Je l’ai connecté sur le ordinateur pour le tester. Lorsque je l'ai connecté, l'image capturée par la caméra est très floue. J'ai essayé d'ajuster la caméra pour la faire la mise au point. Ensuite, j'ai commencé à apprendre à Pixy à reconnaître un objet. Dans pixymon, j'ai choisi ‘Signature 1’ pour enregistrer la cible du suivi de la caméra. J'ai utilisé le trousseau à titre d'exemple pour tester l'appareil photo. Après l'objet est sélectionné, la caméra suit la position de l'objet bleu sur le trousseau.



Tester avec l'Arduino

Après tester la caméra, je l'ai connecté avec Arduino.

J'ai téléchargé la bibliothèque Arduino sur le site et puis j'ai ouvert le logiciel Arduino et importé le fichier de bibliothèque Pixy correspondant. Pour tester le programme, il faut le compiler et l'importer.

```
#include <PixyI2C.h>
#include <Pixy.h>
#include <TPixy.h>

#include <SPI.h>
// Author: Scott Robinson
// charmedlabs.com
//
// Continuously prints blob data
// using the Pixy library.
#include <SPI.h>
#include <Pixy.h>

Pixy pixy;

void setup()
{

  Serial.begin(9600);
  Serial.print("Starting...\n");
}

void loop()
{
  static int i = 0;
  int j;
  uint16_t blocks;
  char buf[16];

  blocks = pixy.getBlocks();

  if (blocks)
  {
    i++;

    if (i%50==0)
    {
      sprintf(buf, "Detected %d:\n", blocks);
      Serial.print(buf);
      for (j=0; j<blocks; j++)
      {
        sprintf(buf, "  block %d: ", j);
        Serial.print(buf);
        pixy.blocks[j].print();
      }
    }
  }
}
```

Ensuite j'ai ouvert le moniteur série et il a affiché :

Detected 1: block 0: sig: 1 x: 159 y: 109 width: 61 height: 61Detected 1: block 0: sig: 1 x: 173 y: 114 width: 60 height: 61Detected 1: block 0: sig: 1 x: 146 y: 111 width: 70 height: 65…

Il faut communiquer les informations d'objet identifiées par pixy à Arduino via le port série. Arduino analysera ensuite ces informations pour contrôler les moteurs.

getBlocks () : Cette fonction retournera le nombre d'objets reconnus par Pixy. Ensuite, on peut obtenir les données de chaque objet identifié via le tableau de *pixy.blocks [ ]* (chaque membre du tableau correspond à un objet identifié). Chaque membre (i) contient les éléments suivants:

pixy.blocks [i] .signature: le numéro d'étiquette de l'objet identifié;

pixy.blocks [i] .x: les coordonnées de la position centrale de l'objet identifié dans la direction x;

pixy.blocks [i] .y: les coordonnées de la position centrale de l'objet identifié dans la direction y;
pixy.blocks [i] .width: la largeur de l'objet identifié (1 à 320);
pixy.blocks [i] .height: hauteur de l'objet identifié (1 200);
pixy.blocks [i] .print (): une fonction membre utilisée pour imprimer les informations de l'objet identifié sur le port série.

Partie électronique

Les fonctions réalisées

Tout d'abord, je dois déterminer ce que le circuit doit réaliser. J'ai décidé d'ajouter des boutons et des commutateurs pour contrôler le fonctionnement ou l'arrêt du système. Dans le même temps, j'ai ajouté trois LED pour observer l'état du système.

1). LEDs

- LED rouge: indique si le programme est mis en pause.

- LED orange: indique si le système est alimenté (couplée à l'interrupteur).

- LED vert: indique si le programme tourne.

2). Bouton

Le bouton à contact temporaire contrôle la mise en route du système. Une fois le bouton appuyé le programme de gestion du labyrinthe se met en marche. Si le bouton est appuyé à nouveau le programme se met en pause.

3). Interrupteur

L'interrupteur contrôle l'alimentation du système. Activer l'interrupteur signifie que le système est déjà en place et que le courant circule.

4). Servomoteurs

Je me réfère au projet IMA4 de 2015 concernant la commande de servos bioloid pilotés par Arduino. J'utilise un buffer permettant de commander les servos.

Tester le servomoteur

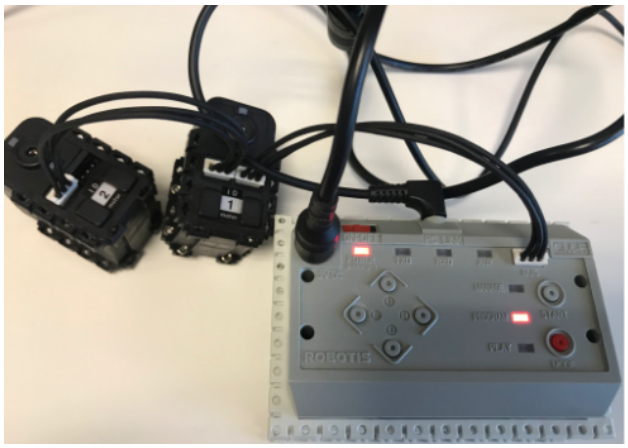
Pour tester les servomoteurs, je m'ai abord référé aux instructions d'utilisation et puis j'ai connecté les deux moteurs avec la source. Et ensuite pour les programmer, il faut installer le logiciel 'Behavior Control Programmer' avec le disque. J'ai essayé de réaliser l'action facile. Je règle séparément la vitesse et l'angle des deux moteurs, puis observe leur comportement. Il y a deux problèmes :

- Le premier moteur tourne à la vitesse spécifiée, mais le deuxième moteur ne tourne pas. J'ai essayé de changer l'ordre des connexions ou de changer le programme, mais il y a rien qui change.

- Le premier moteur tourne éternellement, l'angle cible j'ai défini est invalide.

J'ai consulté le livre de référence et demandé à mon tuteur. J'ai trouvé que la raison pour laquelle le moteur n°2 ne tournait pas était que son numéro de série était devenu le n°6. Ensuite, le programme a changé son numéro en n°2. Testez à nouveau et le moteur n°2 fonctionne correctement. Deuxièmement, il faut précédemment défini la vitesse de rotation et l'angle de rotation du moteur en sélectionnant différentes fonctions.

Cependant, j'ai découvert plus tard que le moteur ne tournait pas à la vitesse et à l'angle spécifiés. J'ai ensuite essayé de contrôler la rotation des deux moteurs via les boutons. Les deux boutons de haut et bas sont pour contrôler le moteur n°1, les deux de gauche et droite sont pour contrôler le moteur n°2. Et le programme est comme ci-dessous :

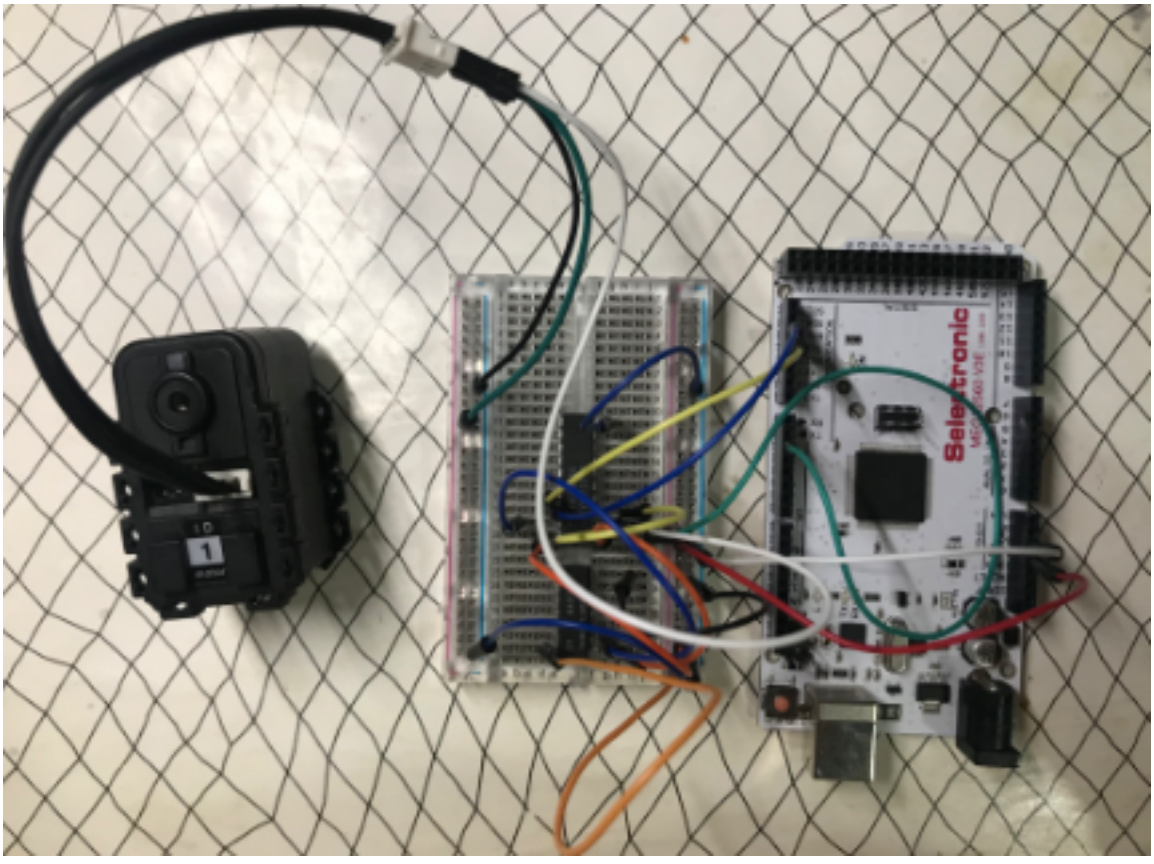


Pour ce projet, je dois remplacer le CM5 par une carte Arduino, ceci afin d'avoir des possibilités de programmation étendue. Après avoir étudié ces caractéristiques, j'ai remarqué que l'on pourrait directement passer par les ports RX et TX de l'Arduino pour le contrôle des servomoteurs, il suffirait juste d'ajouter un composant 74LS241N

pour les contrôles alternatifs de TX et RX. En effet ce buffer trois états permettrait de faire passer alternativement Rx ou Tx en état d'haute impédance, ceci dans le but de n'avoir qu'une unique information sur le bus data vers le servo.

En raison du composant manquant 74LS241N, j'ai choisi de remplacer ce composant par un buffer 74LS244N. J'ai réalisé les branchements entre l'Arduino et les servos via ce composant en nous aidant de les datasheets des ces composants. De cette façon, il sera possible de se substituer du CM5 et de commander directement les servos via l'Arduino. J'ai effectué quelques tests en connectant les ports TX et RX de l'Arduino sur les différentes entrées du buffer 3 états. Le test a échoué et la connexion entre les servos et Arduino n'a pas été établie avec succès.

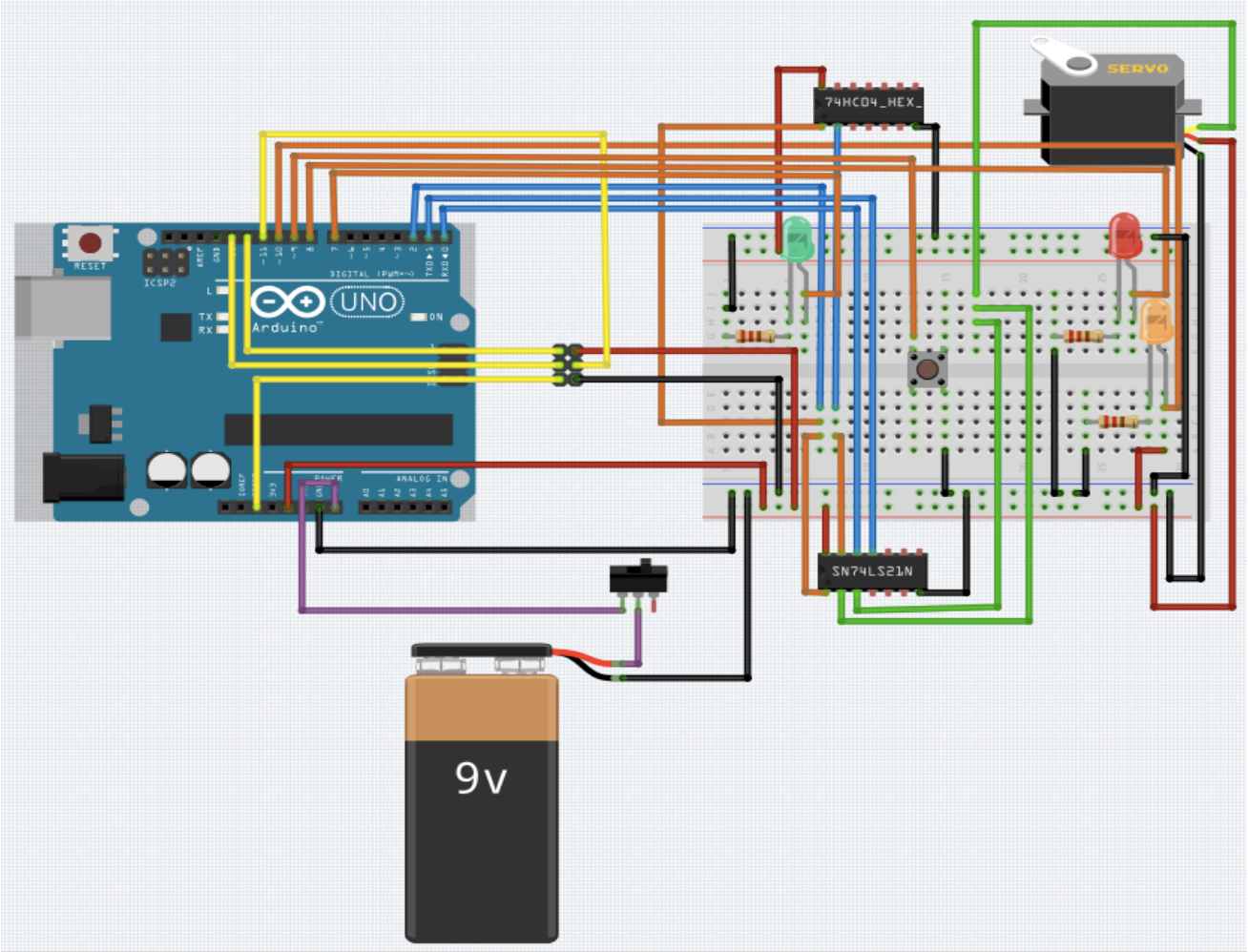
En recherchant les informations, j'ai trouvé que la différence entre 74LS241N et 74LS244N est que le port 2G de ce dernier doit être inversé. En d'autres termes, 74LS241N est un buffer non inverseur et ne peut pas être remplacée directement. Cela signifie que je dois ajouter un inverseur pour obtenir le même effet.
J'ai donc ajouté l'inverseur 74HC04N, ce qui donne pour le circuit, la figure ci-dessous :



Après avoir ajouté inverseur, j'ai testé le moteur avec un exemple de code. Le moteur a tourné avec succès. J'ai ouvert le moniteur série en même temps, on peut voir les informations imprimées et connaître la rotation du moteur. On peut utiliser des programmes pour ajuster différents moteurs, y compris le sens de rotation, la vitesse de rotation et l'angle de rotation.

Dessiner de la carte électronique

Une fois le test terminé, j'ai dessiné le schéma dans le logiciel et ajouté des LEDs, bouton et interrupteur pour contrôler et afficher l'état du système. J'ai également ajouté un connecteur entre la caméra et l'Arduino. Il faut tester davantage ce circuit.



Mon tuteur m'a conseillé de choisir d'utiliser la carte électronique d'essai à bande. C'est plus concis et pratique. Cet objectif sera atteint dans les travaux futurs.

Dans le futur

1. Partie électronique :

Dans un travail ultérieur, je vais choisir le type de la carte. Il y a deux choix :

- Dessiner le PCB, et puis faire la soudure
- Utiliser la carte électronique d'essai à bande

Après déterminer, je vais finir la partie électronique. Dans ce cas, je peut tester le programme qui contrôler les moteurs.

2. Partie programmation :

Dans cette partie, je dois chercher les informations sur le contrôle PI et ensuite écrire le programme pour commander les servos.

3. Partie mécanique (Assemblage) :

Après avoir soudé toutes les cartes et implémenté les fonctions de lecture de la caméra et de contrôle des servomoteurs, je vais ensuite tout assembler sur les planche. Enfin je vais tester le système et ajuster.

Annexes

Bibliothèque Arduino – Servomoteur Dynamixel :

https://www.savageelectronics.com/blog/arduino-biblioteca-dynamixel

Bibliothèque Arduino – La camèra Pixy :

https://pan.baidu.com/s/1dDpDlvV - list/path=%2F