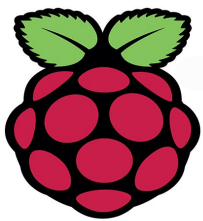


Rapport technique : Projet bras robot



Raspberry Pi

Auteurs:

Tanguy LE DAUPHIN
Ludovic EMMANUEL
Hans TIYE
Hichem BEN YOUSSEF

Client:

Amine RHOUNI

SOMMAIRE

I. Mode d'emploi

- a) Matériels nécessaires
- b) Installer OpenCV
- c) Brancher la raspberry
- d) Connecter ordi et raspberry
- e) Lancer le programme

II. Documentation technique

- a) Robot
- b) TCP
- c) Servo moteur
- d) Caméra

III. Choix techniques

IV. Améliorations

- a) robot
- b) caméra

I. Mode d'emploi

a) Matériels nécessaires

- Dobot-Magician : 1500 euros
https://www.amazon.fr/programmation-DOBOT-imprimante-Porte-Stylo-den%C3%A8vement/dp/B06WW8MR66/ref=sr_1_cc_1?s=aps&ie=UTF8&qid=1527068690&sr=1-1-catcorr&keywords=dobot+magician
- Raspberry PI 3 : 40 euros
https://www.amazon.fr/Raspberry-Pi-Carte-M%C3%A8re-Model/dp/B01CD5VC92/ref=sr_1_1?ie=UTF8&qid=1527068405&sr=8-1&keywords=raspberry+pi3
- Raspberry Pi NoIR Camera V2 : 40 euros
https://www.amazon.fr/LS-Raspberry-Official-Vision-Infrared/dp/B07689J3PL/ref=sr_1_6?ie=UTF8&qid=1527065896&sr=8-6&keywords=camera+raspberry+pi+v2
- Servo-moteur groover: 10 euros
<https://www.gotronic.fr/art-servomoteur-grove-20640.htm>
- TC4427 : 1 euro
<https://www.mouser.com/ds/2/268/20001422G-967549.pdf>
- régulateur 7805 : 1 euro
https://www.amazon.fr/7805-Tension-R%C3%A9gulateur-TO220-lot/dp/B00KM1MXA2/ref=sr_1_1?s=computers&ie=UTF8&qid=1527068960&sr=1-1&keywords=r%C3%A9gulateur+7805
- Composant électronique (2 capa polarisées, 2 LED, 2 Résistance, un connecteur) : 3 euros

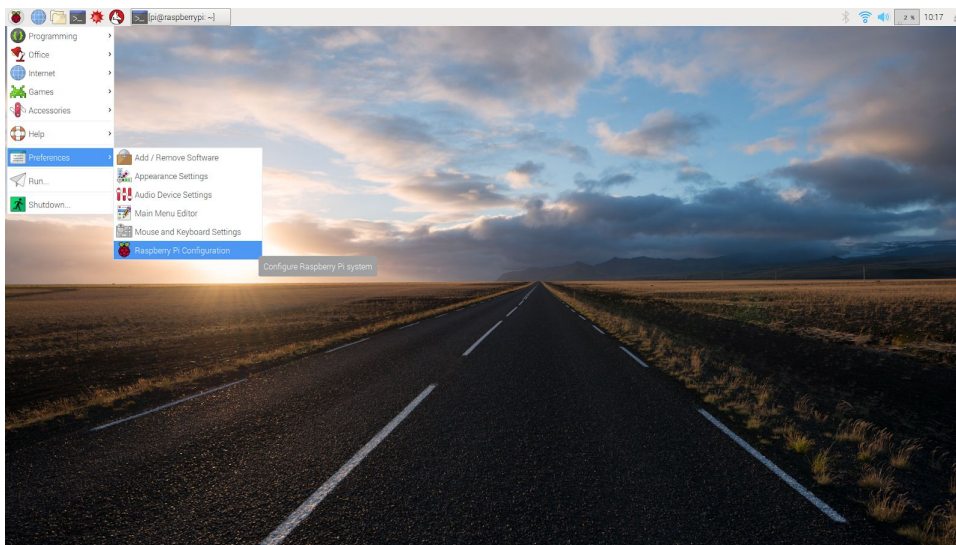
b) Installation OPENCV

Manuel d'utilisation et d'installation sur raspbian (rasberry Pi)

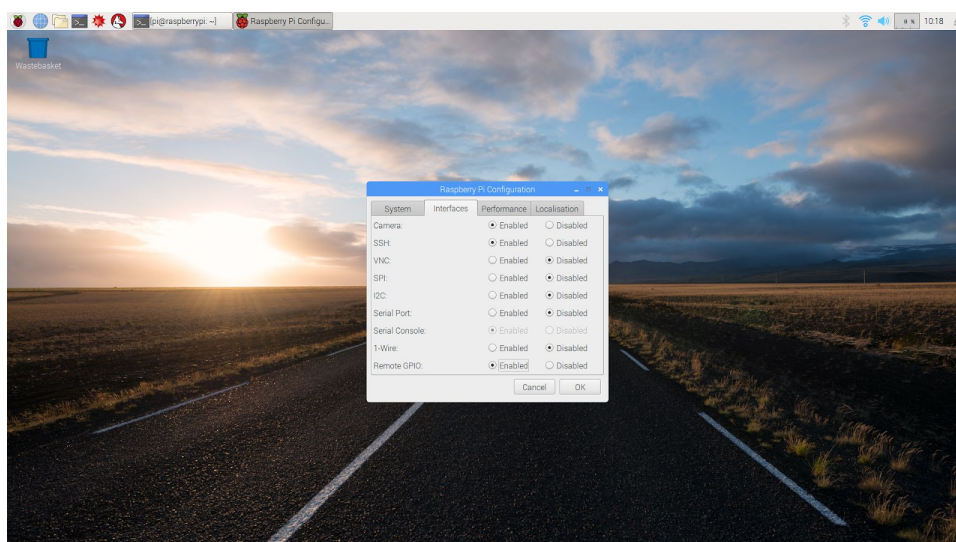
Pour le robot, tous les fichiers sources sont fournies, il faut juste compiler sous Linux (marche parfaitement sur Raspbian).

Sur Raspbian, pour être sûr que tout fonctionne, commencez par taper sur le terminal la commande :

- `sudo rpi-update`
- `sudo apt-get upgrade`
- `sudo apt-get update`
- `sudo apt-get autoremove`



Ensuite aller dans le menu puis sur Préférences et enfin Raspberry Pi configuration comme suit :



Allez dans l'onglet Interface puis activez, si ce n'est pas déjà fait, les options suivantes:

Ouvrez un terminal:

- 1) Tapez “`cd ~`” pour être à la racine de l’user sur raspberry pi
- 2) Tapez sur le terminal la commande :
`sudo apt-get install build-essential cmake cmake-curses-gui`
- 3) Ensuite, tapez la commande suivante :
`sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev libavcodec-dev libavformat-dev libswscale-dev libeigen3-dev libxvidcore-dev libx264-dev`
(ca risque de prendre un peu de temps).
- 4) Tapez :`sudo apt-get install libgtk2.0-dev`
- 5) Tapez à la suite les lignes de commandes comme suit :

```
sudo apt-get -y install libv4l-dev v4l-utils
sudo modprobe bcm2835-v4l2
mkdir opencv
cd opencv
wget https://github.com/opencv/opencv/archive/3.4.1.zip -O opencv_source.zip
wget https://github.com/opencv/opencv\_contrib/archive/3.4.1.zip -O opencv_contrib.zip
```

```
unzip opencv_source.zip
unzip opencv_contrib.zip
cd opencv-3.4.1
mkdir build
cd build
ccmake ./ ( appuyez sur la touche ‘c’ et attendez le chargement, appuyez une
nouvelle fois sur la touche ‘c’ et attendez le chargement. Enfin, appuyez sur la
touche ‘g’.)
```

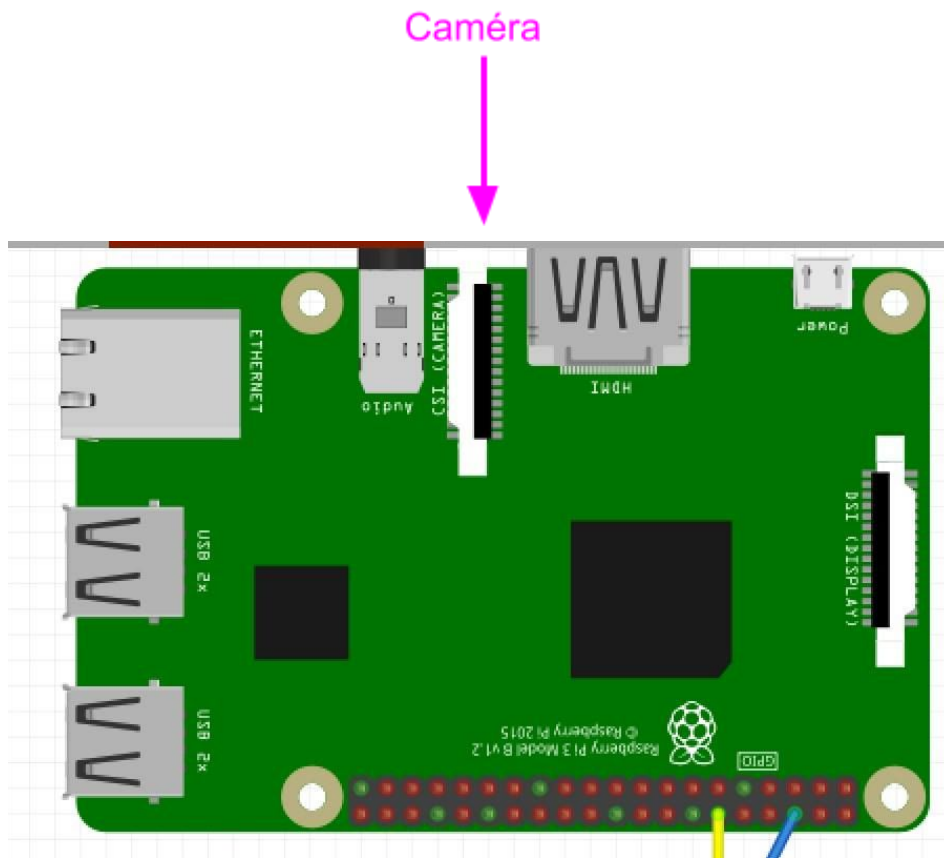
```
make -j4 (environ 1 heure)
```

```
sudo make install
sudo ldconfig
```

c) Brancher la raspberry

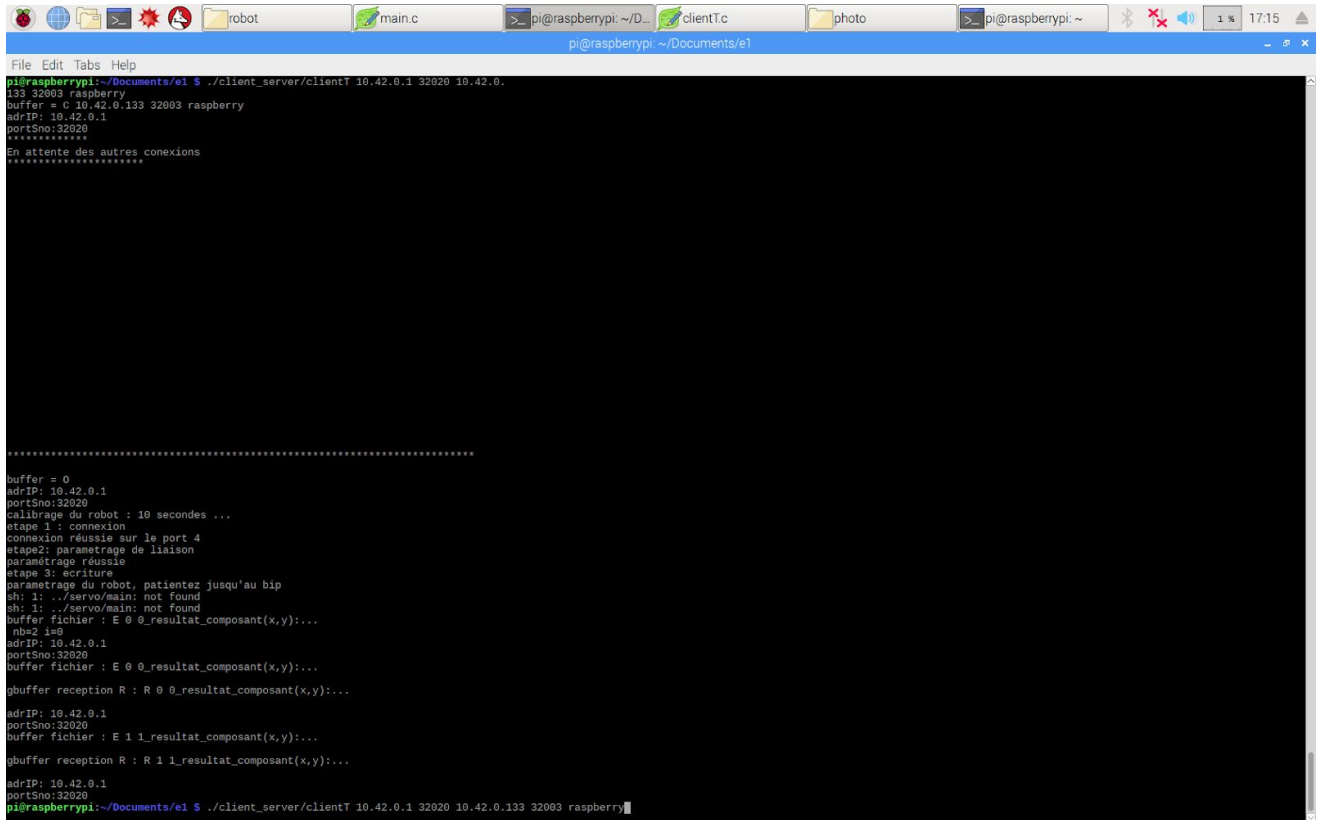
La Raspberry doit être connecté de la manière suivante:

- sa **pin 3** connecté à la **masse** du circuit externe (**fil bleu du circuit**).
- sa **pin 6** connecté à la **commande** du circuit externe (**fil jaune du circuit**).
- La **Caméra** (Raspberry Pi NoIR Caméra V2) dans le **port caméra**.



d) Connecter ordinateur et raspberry

Afin de pouvoir envoyer des données en wi-fi, il faut que l'ordinateur client, la raspberry et le serveur soient sur le même réseau.



```
pi@raspberrypi: ~/Documents/e1
pi@raspberrypi:~/Documents/e1 $ ./client_server/clientT 10.42.0.1 32020 10.42.0.133 32003 raspberry
buffer = c 10.42.0.133 32003 raspberry
adrIP: 10.42.0.1
portSno:32020
*****
En attente des autres conexions
*****

buffer = 0
adrIP: 10.42.0.1
portSno:32020
Calibrage du robot : 10 secondes ...
etape 1 : connexion
connexion réussie sur le port 4
etape2: parametrag de liaison
paramétrage réussie
etape 3: écriture
parametrag du robot, patientez jusqu'au bip
sh: 1: ../servo/main: not found
sh: 1: ../servo/main: not found
buffer fichier : E 0 0_resultat_composant(x,y):...
nb=2 i=0
adrIP: 10.42.0.1
portSno:32020
buffer fichier : E 0 0_resultat_composant(x,y):...
gbuffer reception R : R 0 0_resultat_composant(x,y):...

adrIP: 10.42.0.1
portSno:32020
buffer fichier : E 1 1_resultat_composant(x,y):...
gbuffer reception R : R 1 1_resultat_composant(x,y):...

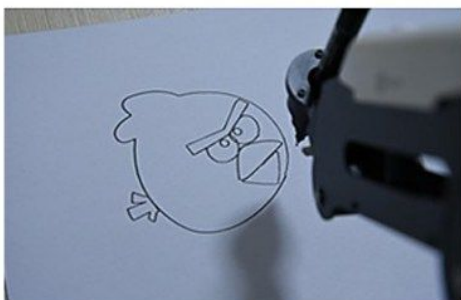
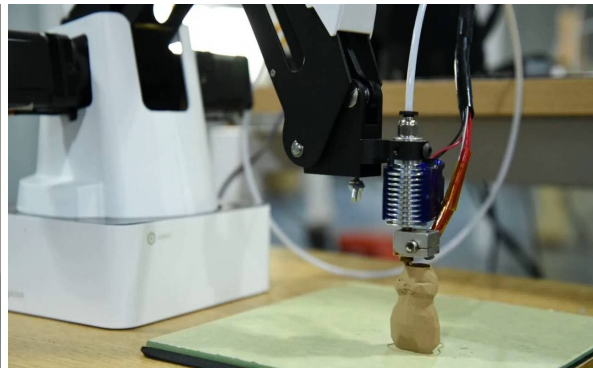
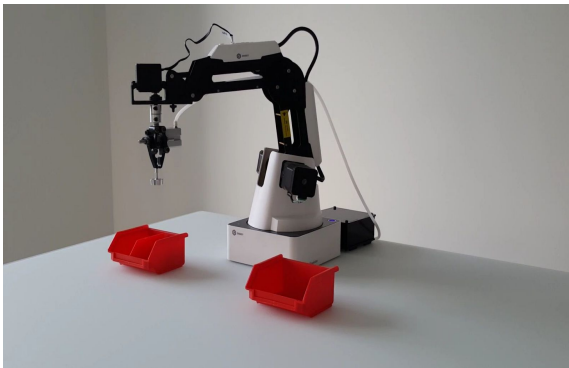
adrIP: 10.42.0.1
portSno:32020
pi@raspberrypi:~/Documents/e1 $ ./client_server/clientT 10.42.0.1 32020 10.42.0.133 32003 raspberry
```


II. Documentation technique

a) Le robot

Nous disposons pour ce projet du **DOBOT Magician**. Il s'agit d'un bras robotisé multifonctionnel pour la formation pratique. Il est installé avec plusieurs outils, DOBOT Magician peut réaliser différentes fonctions intéressantes telles que l'impression 3D, la gravure laser, l'écriture et le dessin. Il prend en charge le développement secondaire par 13 interfaces extensibles et plus de **20 langages de programmation**, ce qui augmente réellement la créativité et l'imagination sans aucune limitation.

Notre objectif principal étant de déplacer un composant, nous allons essentiellement nous servir de la ventouse et d'une pompe pour aspirer les composants.



Codes sources

Nous avons choisi de coder en C. Nous avons donc écrit plusieurs scripts afin de réaliser ce que le robot était capable de faire via le soft fourni avec.

Notre main est principalement constitué de trois parties: **l'initialisation**, **l'envoi** et enfin **la récupération**.

Initialisation:

Il existe une fonction "init()" qui permet donc de connecter le robot et de faire l'initialisation.

```
void init()
{
    printf("calibrage du robot : 10 secondes ...\n");
    sleep(10);

    connect();
    printf("parametrage du robot, patientez jusqu'au bip\n");
    sleep(1);
    suction_on();
    sleep(4);
    suction_off();
    sleep(1);
    bouger(166.4899,0.00001,-14.2097,0.0);
    sleep(2);
    home();
    bzero(text,23);
}
```

L'initialisation se fait en 10 secondes environ. La fonction connect permet ainsi de paramétrer le robot (Ceci remplace le bouton connect présent dans le soft fourni). La fonction home est une fonction qui sert enfin à faire une remise à zéro des distances et définit un point de départ afin de ne pas se tromper sur nos mesures.

Nous parlerons plus tard des fonctions "suction_on()", "suction_off()" et "bouger()"

Les points :

Afin d'enregistrer l'emplacement des composants, nous avons choisis de faire un tableau de points. Chaque point a quatre caractéristiques : la position en x,y,z et la rotation de la ventouse. Nous avons donc pour ce cas créé une structure nommée "point".

```
typedef struct point
{
    float x;
    float y;
    float z;
    float r;
} point;
```

Structure point

```

tableau_point[0].x = 198.1627;
tableau_point[0].y = -70.1703;
tableau_point[0].z = -64.8000;
tableau_point[0].r = 0.0001;

```

```

tableau_point[1].x = 196.9675;
tableau_point[1].y = -45.0118;
tableau_point[1].z = -64.7733;
tableau_point[1].r = 0.0001;

```

```

tableau_point[2].x = 194.8586;
tableau_point[2].y = -16.8152;
tableau_point[2].z = -64.0068;
tableau_point[2].r = 0.0001;

```

Tableau de point

L'envoi:

Cette partie consiste à récupérer le composant à une position précise et le déposer sur la plaque de test.

```

void envoyer(void)
{
    bouger(191.0451,-69.6064,1,-30.15);

    bouger(198.6411,-70.1703,-62.9907,-30.15);
    suction_on();
    wait_s(1);
    bouger(191.0451,-69.6064,1,-30.15);

    bouger(55.3118,209.5573,1.1547,-28.7470);

    bouger(55.3118,209.5573,-51.01,-28.7470);
    sleep(32);
    suction_off();
    sleep(2);
    bouger(55.3118,209.5573,1.1547,-28.7470);
}

```

La fonction "bouger()" prend en paramètres les valeurs de x,y,z et r. Comme vous pouvez le deviner, cette fonction permet de bouger le robot à la position définie.

La fonction "suction_on()" permet d'activer la succion et ainsi de récupérer le composant. De ce fait, la fonction "suction_off()" permet de l'éteindre et ainsi relâcher le composant.

```

void suction_on()
{
    bzero(text,23);
    htoa("AAAA043E010101BF");
    write (fd, text, 8);
    tcdrain(fd);
    bzero(text,23);
}

```

```

void suction_off()
{
    bzero(text,23);
    htoa("AAAA043E010000C1");

    write (fd, text, 8);
    tcdrain(fd);
    bzero(text,23);
}

```

La récupération:

Le principe est le même que pour l'envoi sauf qu'on fait la démarche inverse. Le code est donc aussi inversé car on déplace maintenant le composant de la plaque de test à sa position initiale.

```
void recuperer()
{
    sleep(1);
    bouger(55.3118,209.5573,-52.4279,-28.7470);
    wait_s(1);
    suction_on();
    bouger(55.3118,209.5573,1.1547,-28.7470);
    bouger(191.0451,-69.6064,1,-30.15);
    bouger(198.6411,-70.1703,-62.9907,-30.15);
    sleep(7);
    suction_off();
    bouger(191.0451,-69.6064,1,-28.7470);
}
```

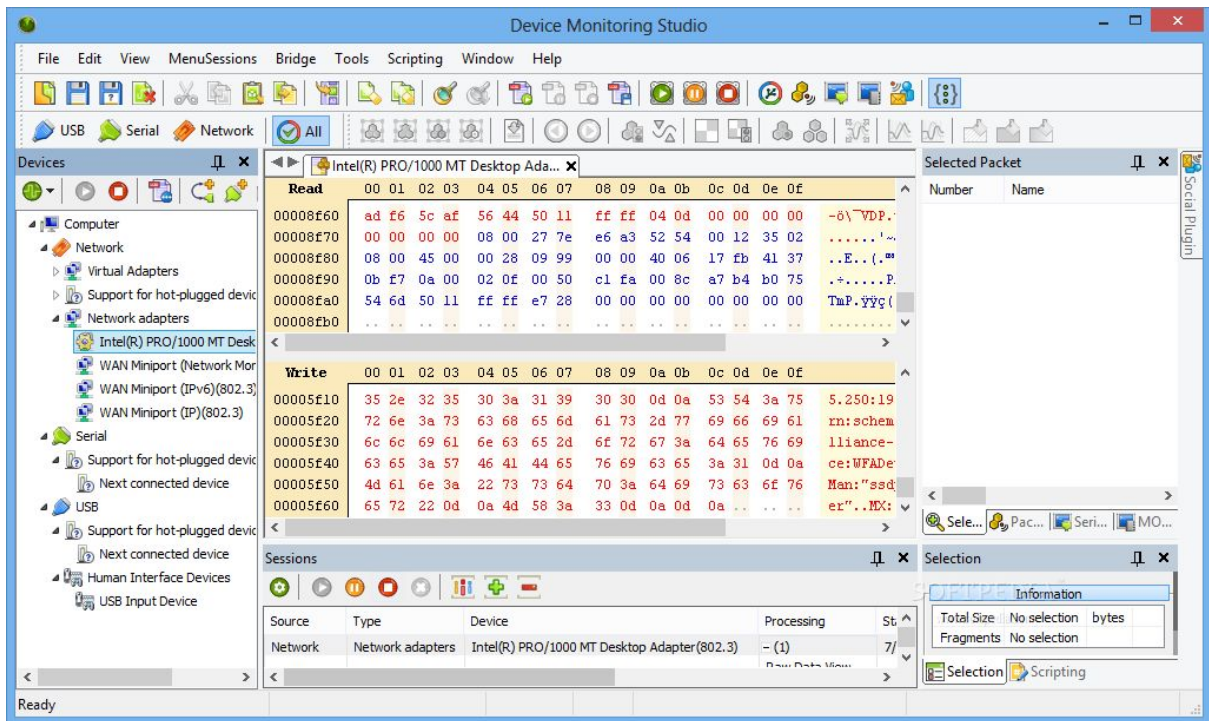
Conception du logiciel:

Pour coder ce logiciel, nous avons dû hacker le robot, c'est-à-dire observer les trames envoyées depuis le logiciel officiel vers le robot. Avec plusieurs trames, nous avons pu décoder le protocole utilisé, il est de ce type :

Header	Len	Payload				Checksum
		ID	Ctrl		Params	
			rw	isQueued		
0xAA 0xAA	2+0	0	1	0	Empty	Payload checksum

Une documentation technique avait été fournie concernant ces trames mais elles étaient d'une grande incompréhension. Nous avons donc dû "sniffer" les trames pour les réutiliser en utilisant le logiciel Device Monitoring Studio.

Interface de Device Monitoring Studio



Exemple de trames

Structure des trames:

0xAAAA : Header, toutes les trames commence par cela.

Len : C'est la taille de la Payload en octet, minimum 2 car il faut compter ID et CTRL, ils sont toujours présent.

Payload : partie importante, information de la trame

ID: identificateur de la trame, chaque instruction a un ID.

Ctrl: Pour contrôler si c'est un immédiat ou pas.

Params: data pour certains paramètres (coordonnées par exemple).

Checksum : sécurité pour vérifier si les données précédentes se sont bien envoyées. Si le checksum est faux, le code ne sera pas exécuté par le robot. Il doit être calculé comme ceci :

A partir de ID, on additionne chaque octet, en rajoutant le checksum l'octet de poids faible doit valoir 0.

Exemple: On envoie une trame :

0xAA AA (header) 0A (Len) 02 (ID) 03 (Ctrl) 5C 00 00 A1 2F 00 76 A0 (Params)

Pour calculer le checksum, on ne prend que les octets à partir de ID soit:

02 03 5C 00 00 A1 2F 00 76 A0

on additionne un par un les octets : $02+03+5C+00+00+A1+2F+00+76+A0 = 0x247$.

Le checksum est donc B9 car $0x247 + 0xB9 = 0x300$, le dernier octet est bien égale à 0.

b) Connexion sans fil (TCP)

Pour notre projet, nous devons avoir un moyen de communiquer entre l'ordinateur du client et la carte raspberry qui contrôle le bras robotisé. Pour ce faire, nous avons décidé d'utiliser la méthode TCP. Voici ci-dessous le diagramme des échanges :

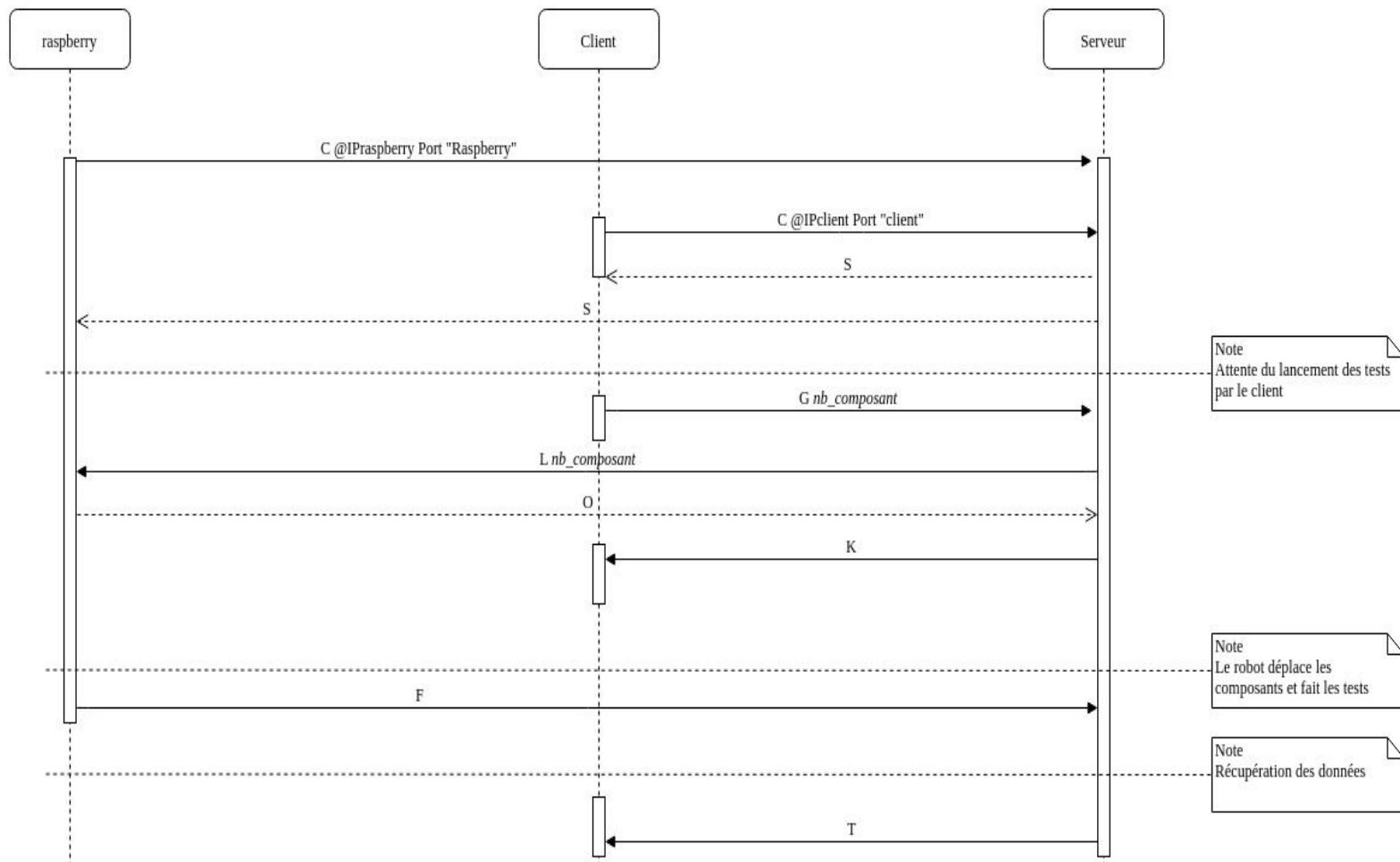


Figure : Échange TCP

Pour permettre une bonne communication entre la raspberry et le client, nous avons intégré un serveur qui sert d'intermédiaire dans les échanges.

Commandes pour lancer les différents programmes :

- raspberry : `./client @serv portServ @iplocale portlocal client`
- client : `./client @serv portServ @iplocale portlocal raspberry`
- serveur : `./server port`

c) Servomoteur

Le servomoteur doit être commandé par un signal PWM d'amplitude 5V.

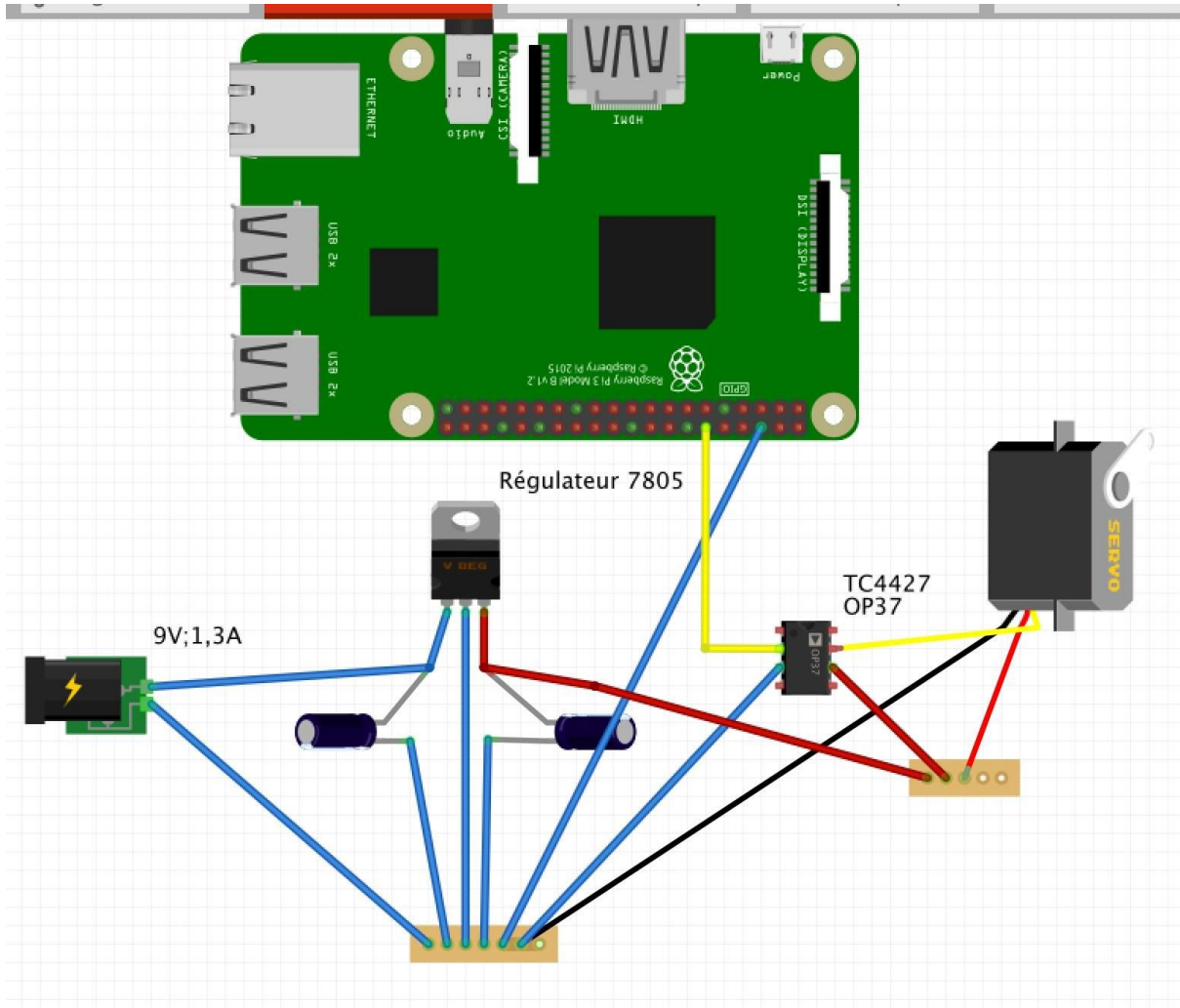
(<https://eskimon.fr/tuto-arduino-602-un-moteur-qui-a-de-la-t%C3%A0te-le-servomoteur>).

Cependant, ce signal provenant de la Raspberry PI peut être déformé lors de chute de tension dû aux calculs de la carte. Il nous faut donc passer par un chip TC4427 qui transforme le signal de commande de 3,3V en 5V sans fluctuation.

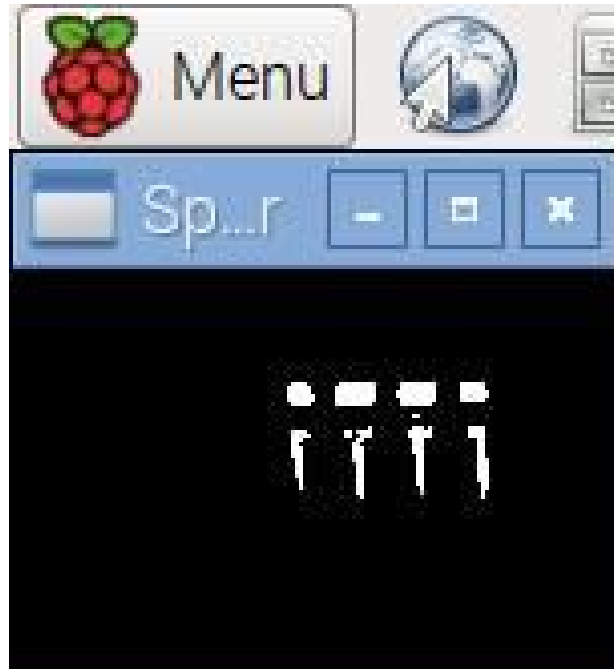
(<http://ww1.microchip.com/downloads/en/DeviceDoc/20001422G.pdf>).

De plus pour alimenter ce chip et notre servomoteur en 5V et pour avoir un ampérage suffisant ($>1A$), on utilise un régulateur 7805 qui transforme notre signal d'entrée de 9V en 5V.

(<http://www.electrokits.com/downloads/pdf/7805-datasheet-fairchild.pdf>).



d) Caméra



La caméra permet de récupérer les pixels sous forme de matrice grâce à la bibliothèque openCV. nous pouvons accéder à chaque point de la matrice grâce à la commande:

créer le rectangle qui va accueillir l'image

```
Rect rectangle(375,350,150,100);
```

récupère le flux vidéo et le met dans une matrice.

```
cap >> frame;
//////////
petitframe = frame(rectangle);
```

Puis on redimensionne la matrice à la taille de rectangle.

On transforme la matrice contenant des pixels colorés vers une matrice ne contenant que des pixels blancs ou noirs.

```
cv::inRange(petitframe, grism, grisp, frameGris);
```


Ici les 1 signifie que l'on n'a pas buté sur une patte.
Plus la valeur est grande, plus on bute rapidement sur le composant.
Dans notre exemple la patte la plus basse dans la matrice est la quatrième car on peut compter 3 succession de grand nombre (donc 3 pattes avant) et que la valeur max est de 74 (donc le plus près de la caméra).

Une fois ce tableau obtenu, on utilise la fonction de détection de pattes de composant.

```

////////////////////////////////////DETECTION DE DIFFERENT TYPE COMPOSANT

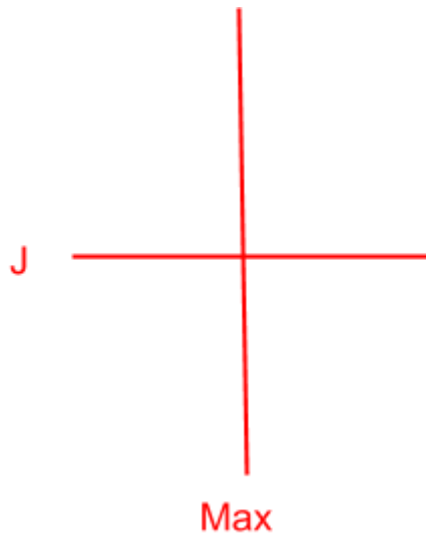
nb_patte =0;
memj=0;
for(j=0;j<frameGris.cols;j++) //on parcour les colone de buté
{
    if(tab[j]>40){ //on détecte quelque chose
        if(tab[j]>max){
            max=tab[j];
            stop_patte = 1;
        }
    }
    else{
        if(stop_patte == 1){
            //printf("ligne : %d ; pos dans la ligne : %d\n",j,max);
            remplir(j,max,nb_patte);
            nb_patte++;
            memj =j;
            max = 30;
        }
        stop_patte=0;
    }
}
}

```

Fonctionnement de ce code:

1. on parcourt le tableau de butée jusqu'à trouver le début d'une patte.
2. Le début trouvé, si la valeur de butée suivante est supérieure, on actualise le **max** de la patte.





3. Quand on trouve une case du tableau de butée =1, alors on se trouve entre 2 pattes d'un composant, la valeur **max** contient la position verticale de la patte et la valeur **J** la valeur horizontale de la patte.
4. On utilise la fonction remplir pour remplir un tableau contenant les positions en X **tabX** et un autre contenant les positions en Y **tabY** de nos pattes.

- Création de la droite de régression linéaire et segment de référence

Grâce à **tabX** et **tabY** on peut maintenant avoir les coefficients a et b de notre droite affine ($y=a*x+b$) passant par les sommets des pattes grâce à la méthode des moindres carrés.

```
void regLin(int x[],int y[], double *a, double*b,int n)
{
    int i;
    double xs,ys,xys,xxs;
    double ai,bi;
    xs=0.0;
    ys=0.0;
    xys=0.0;
    xxs=0.0;
    for(i=0;i<4;i++)
    {
        xs=xs+x[i];
        ys=ys+y[i];
        xys=xys+x[i]*y[i];
        xxs=xxs+x[i]*x[i];
    }
    ai=(4*xys-xs*ys)/(4*xxs-xs*xs);
    bi=(ys-ai*xs)/4;
    *a=ai;
    *b=bi;
    return;
}
```

Il faut ici savoir combien de pattes sont analysées pour effectuer cette opération. Ici, la régression linéaire est toujours effectuée sur les 4 premiers points de **tabX** et **tabY**. On ne peut pas définir la taille des tableaux après avoir compté le nombre de pattes (problème de définition de la taille d'un tableau en C).

Enfin une fois la droite obtenue, on modifie la matrice de l'image pour afficher le segment passant par les points.

```
////////////////////////// affichage ligne reg lin

abcisse =0;
ordonne =0;

for(j=0;j<frameGris.cols;j++)
{
    abcisse = j;
    ordonne = (int)(a*(float)j+b);
    if(abcisse>tabx[0] && abcisse<=memj)//tabx[nb_patte-1])
    {
        if((ordonne>0)&&(ordonne<frameGris.rows)){
            frameGris.at<uchar>(ordonne,abcisse)=255;
        }
    }
}
}
```

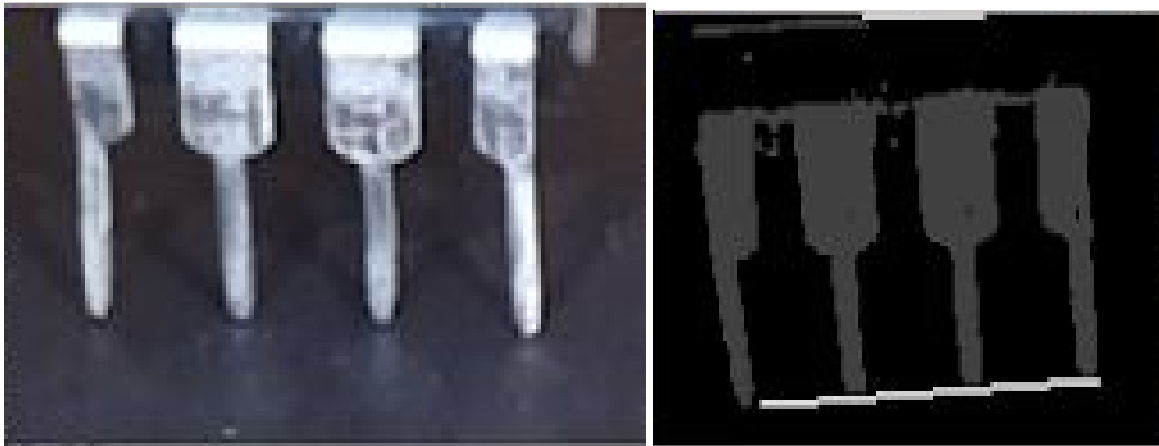
De plus on fait attention à ne tracer la droite que entre la première et la dernière patte, Cela nous permet d'obtenir un segment.

Enfin ce segment peut être comparé à un segment de référence afin de définir si le composant est aligné ou pas avec la position idéale:

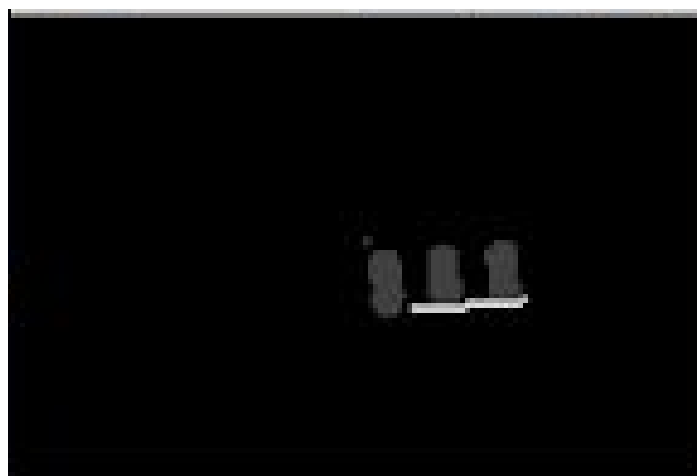
1. A nous permet de changer l'orientation du composant
2. B nous permet de le monter ou le descendre selon l'axe x.
3. Le début du segment et la fin nous permettent de le déplacer selon l'axe Y.

Enfin pour la définition de la droite de référence, nous allons en reparler dans la partie suivante.

Voici le résultat finale:



Il est important de noter que la détection fonctionne sur de plus petit composant et avec un nombre de patte inférieur, ici un composant de 1 mm à 3 pattes.



- Manipulation des options caméras

```
Choix des options
'ESC'  -> quitter
'f't'  ->afficher la camera
'z'    ->zoomer sur le composant
'g'    ->affichage en niveau de gris
'p'    ->augmenter le seuil de detection
'm'    ->diminuer le seuil de detection
's'    ->afficher le segment de reference
'r'    ->prendre une reference
```


III. Choix techniques

Pour mener à bien notre projet, il nous a fallu utiliser un ordinateur pour centraliser toutes nos tâches, tel que l'envoi de données sans fil au robot, asservir un servomoteur et traiter une vidéo.

Nous avons décidé au début du projet de se concentrer sur une carte type microcontrôleur (arduino) mais nous avons dû vite abandonner dû au manque de puissance de calcul et la complexité d'utilisation d'une caméra.

Nous nous sommes donc concentrés sur un ordinateur embarqués: la raspberry Pi. La raspberry Pi (OS Raspbian) nous permet d'avoir toute la puissance de calcul nécessaire et introduire une caméra avec une grande facilité et et avoir plusieurs bibliothèques disponible pour le traitement d'image.

De plus, la raspberry possédant de base tout les modules de communication sans fil (Wi-fi,Bluetooth), possédant nativement un port pour une caméra embarqué et le tout sous environnement Linux, nous avons donc décidés d'adopter la Raspberry définitivement.

Le choix de la communication sans fil était simple, il fallait un protocole assez rapide pour envoyer nos données, le Wi-fi (2,4 GHz) était donc le meilleur choix, malheureusement, le module Wi-fi appartenant au robot ne fonctionnait pas, nous nous sommes donc rapidement concentrer sur le Wi-fi intégré au raspberry.

Pour le test de composant, nous avons choisi d'utiliser un port ZIF pour palier à l'éventualité d'abîmer les pattes, ce type de port accueille les composants avec des larges ouvertures pour ensuite resserrer et créer un contact avec les pattes.

IV. Amélioration

a) Robot

Lors de la conception de notre logiciel, nous voulions mettre plusieurs options tels que l'arrêt d'urgence pour stopper notre robot à tout moment.

Nous voulions aussi améliorer la rapidité et la stabilité de l'action du robot en implémentant plusieurs options tels que la détection d'erreur de placement ou le retour à l'utilisateur des données de position dans l'espace en temps réel.

b) Caméra

Pour améliorer le projet, il faudrait réussir à communiquer le déplacement à effectuer de la caméra au robot en se basant sur la référence définie par l'utilisateur.

1. Pour se faire l'algorithme de la caméra compare la position actuelle du composant avec celle de la référence.
2. Ensuite l'algorithme demande au robot de bouger ou tourner le composant jusqu'à atteindre la référence.
3. Enfin, le robot devra amener le composant correctement positionné vers la matrice de placement.

Cette étape intermédiaire de la matrice permet de fractionner le travail et nous permet de bypasser le fait qu'un composant n'est pas été bien placé sur la matrice avant de le mettre dans le port zif.

Une autre amélioration possible serait d'utiliser un algorithme de reconnaissance de lettre pour lire la référence des composants testés.

On peut imaginer plusieurs types de test que le robot effectuera en fonction du type de composant: différent type de port zif ou de connecteur, différent type de test (tension, température..).

