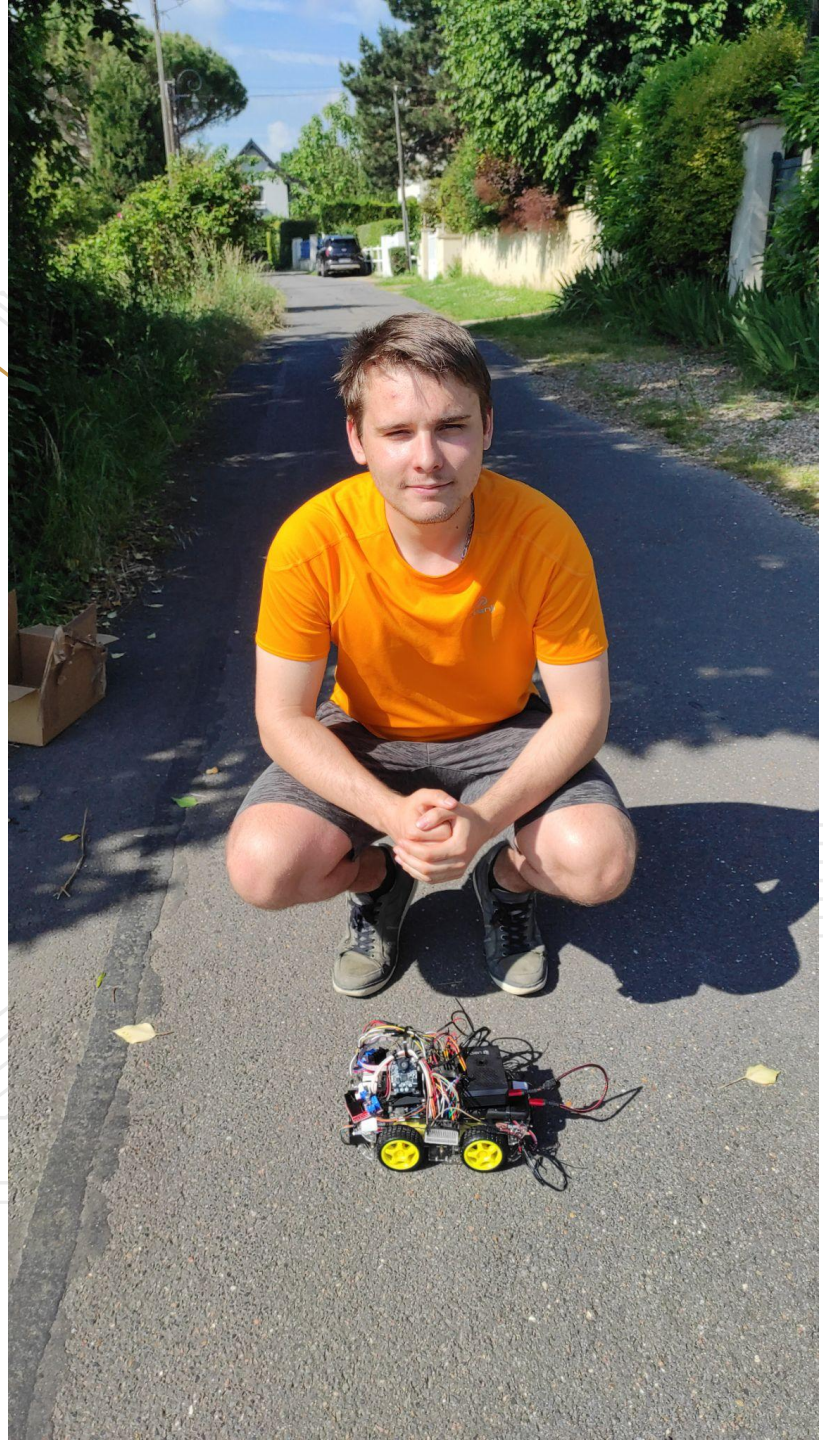
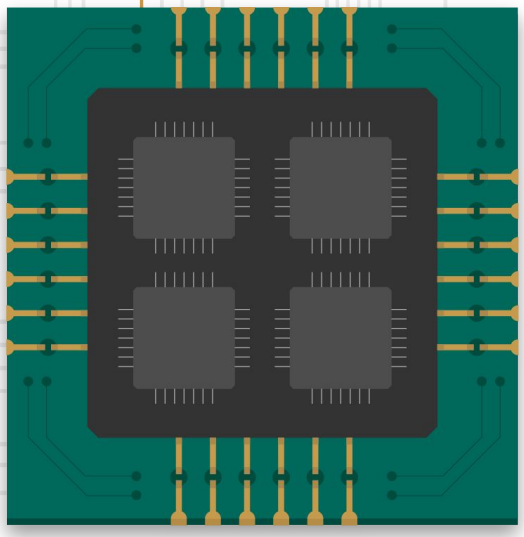
TIPE 2023-2024

Robot suiveur d'athlète



Candidat n° **47180**
DESCHARLES Guillaume

SOMMAIRE



01

Mise en situation du projet

- Inspiration
- Cahier des charges fonctionnel

Inspiration

Son principe est similaire à la caméra de poursuite Speedcam permettant la diffusion d'évènements sportifs comme les courses de 100 mètres

https://docplayer.fr/74425743-Td-asservissements-camera-de-poursuite-speedcam.html

Cahier des charges fonctionnels

-Temps moyen pour que le coureur soit centré : 1 s
-Lors d'un grand changement de vitesse : 5 s
-Vitesse maximum du robot: 3 km/h

-Masse du robot maximal : 1,5 kg

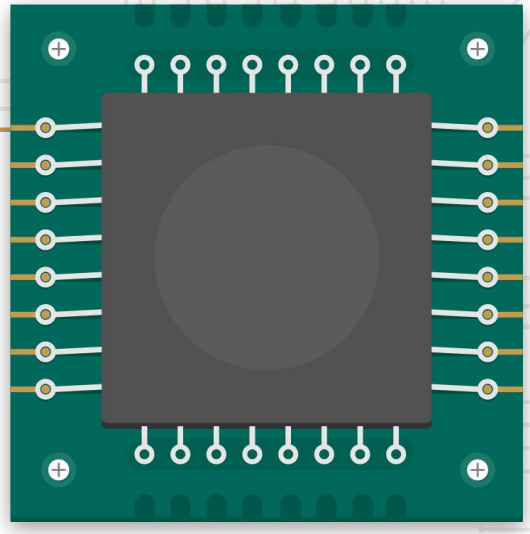
-Fonctionnement sur sol plat

-Temps pour atteindre la vitesse maximum : 500 ms

Principe :

Le robot devra s'adapter à la vitesse du piéton pour qu'il soit centré par rapport au robot.

02 Présentation des motoréducteurs



- Données constructeur
- Dimensionnement d’un motoréducteur

Données du constructeur

https://www.gotronic.fr/pj2-35326-robot03-montage-1601.pdf

Pourquoi 4 roues ?

- **Augmente la stabilité du système**
- **Le centre de gravité reste à l'intérieur d'un rectangle formé par les quatre roues.**
- **Le centre de gravité doit se placer au milieu du rectangle**,

Tension d'alimentation (V)	4,5	6	7,2	9
Consommation (mA)	190	160	180	200
Vitesse de rotation à vide (tour/min)	90	190	230	300
Couple (g/cm)	800	800	1000	1200

Dimensionnement d'un motoréducteur

Données :

- -Alimentation : 9 Vcc
- -Vitesse de rotation à vide : 300 tr/min
- -Couple : 0,12 N.m
- -Dimensions roue : ⌀ 67 x 27 mm

Capacités attendus :

- -Vitesse maximum du robot : 3 km/h
- -Masse du robot maximal : 1,5 kg
- -Fonctionnement sur sol plat
- -Temps pour atteindre vitesse maximum : 500 ms
- -Accélération du robot : 1.67 $m.s^{-2}$

Vitesse angulaire :

$$\omega_{roue} = \frac{2 \times v}{D} = 198\, tr.min^{-1}$$

Dimensionnement d'un motoréducteur

Couple : Système : {Robot de masse m}
Référentiel terrestre supposé galiléen
Les forces agissant sur le véhicule sont la gravité, la réaction du support et la propulsion.
Or le sol est plat donc seule la propulsion agit sur le système.
D'après le principe fondamental de la dynamique :

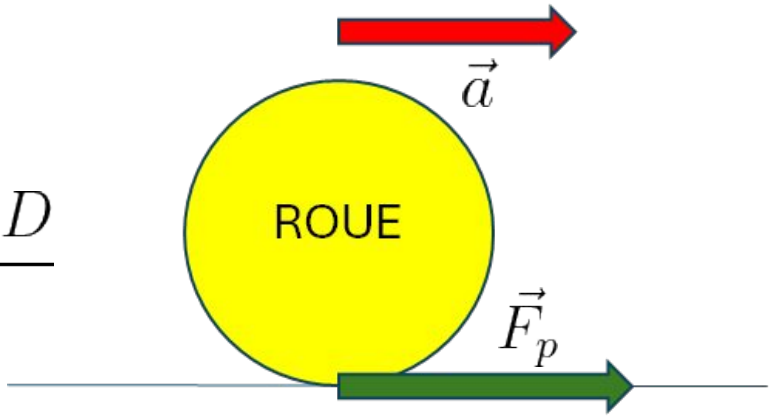
$$\vec{F_p} = \frac{m}{4} \times \vec{a}$$

Or la propulsion et l'accélération ont le même sens et la même direction :

$$F_p = \frac{m}{4} \times a$$

De plus :

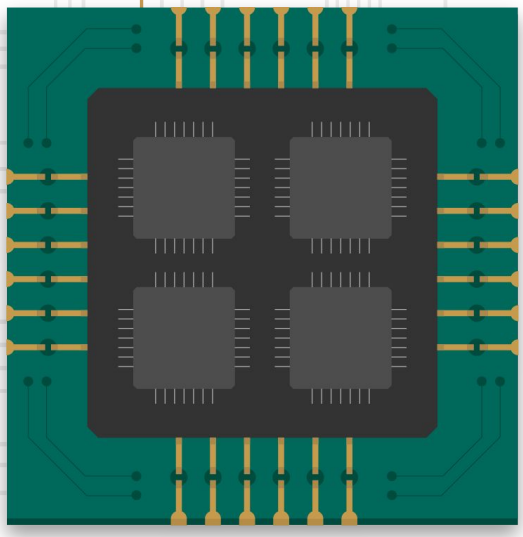
$$C_{roue} = F_p \times \frac{D}{2} = \frac{m \times a \times D}{8}$$



Comme :

m = 1,5 kg D = 67 mm a = 1,67 m.s^{-2}

Le couple sur l'axe de la roue est : $C_{roue} = 0,021\ N.m << 0,12\ N.m$



03 Présentation du système

- Fonctionnement avec une caméra
- Présentation des composants
- Connexion des composants

Fonctionnement avec une caméra

Composants principaux

- Utilisation d'une caméra pixy 2.
- Ajout d'un pan tilt
- Qualité de l'image 480 pixels
- Présence d'une raspberry pi 4

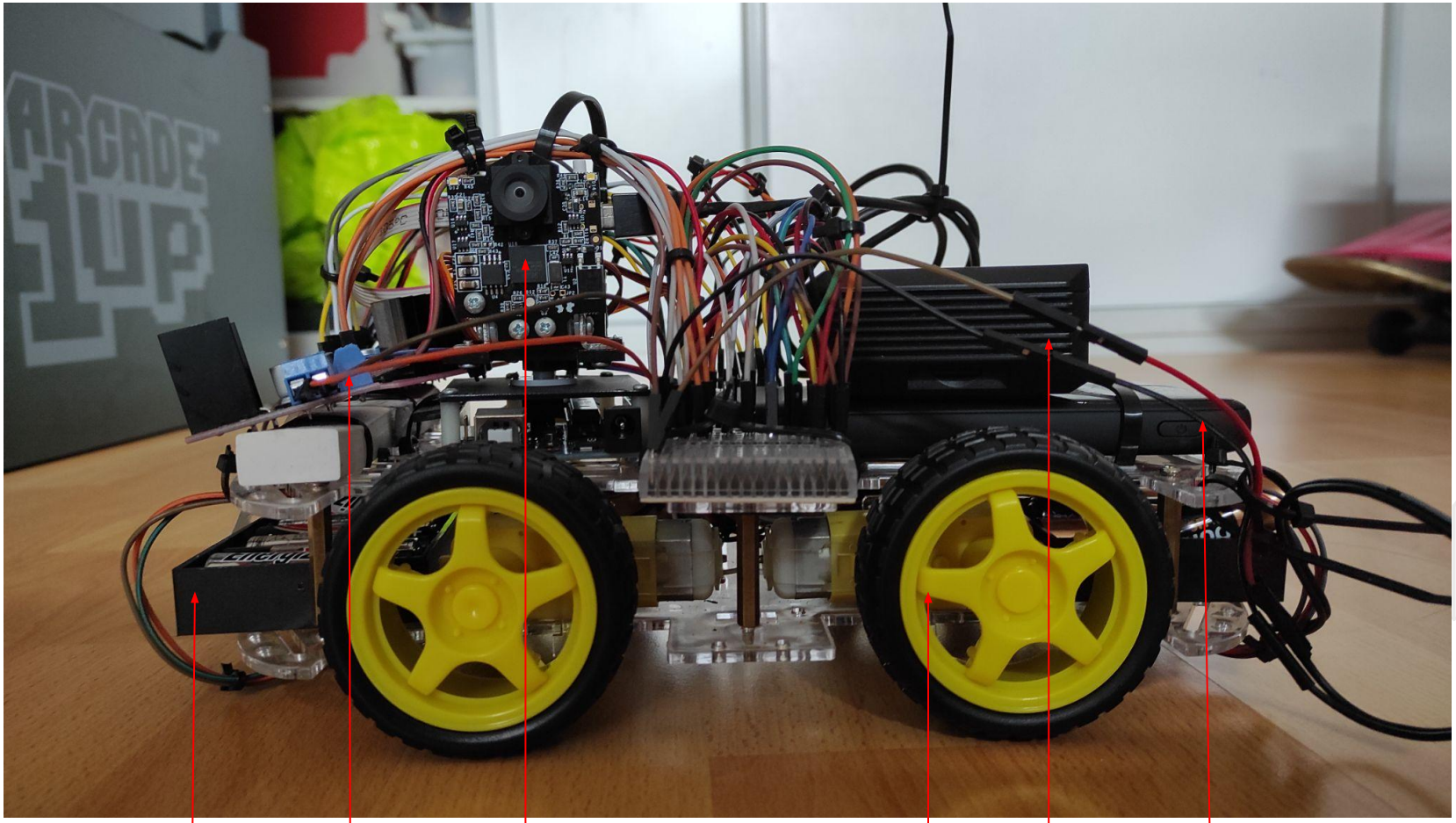
https://docs.pixycam.com/

https://pixycam.com/pixy2-pan-tilt-kit/

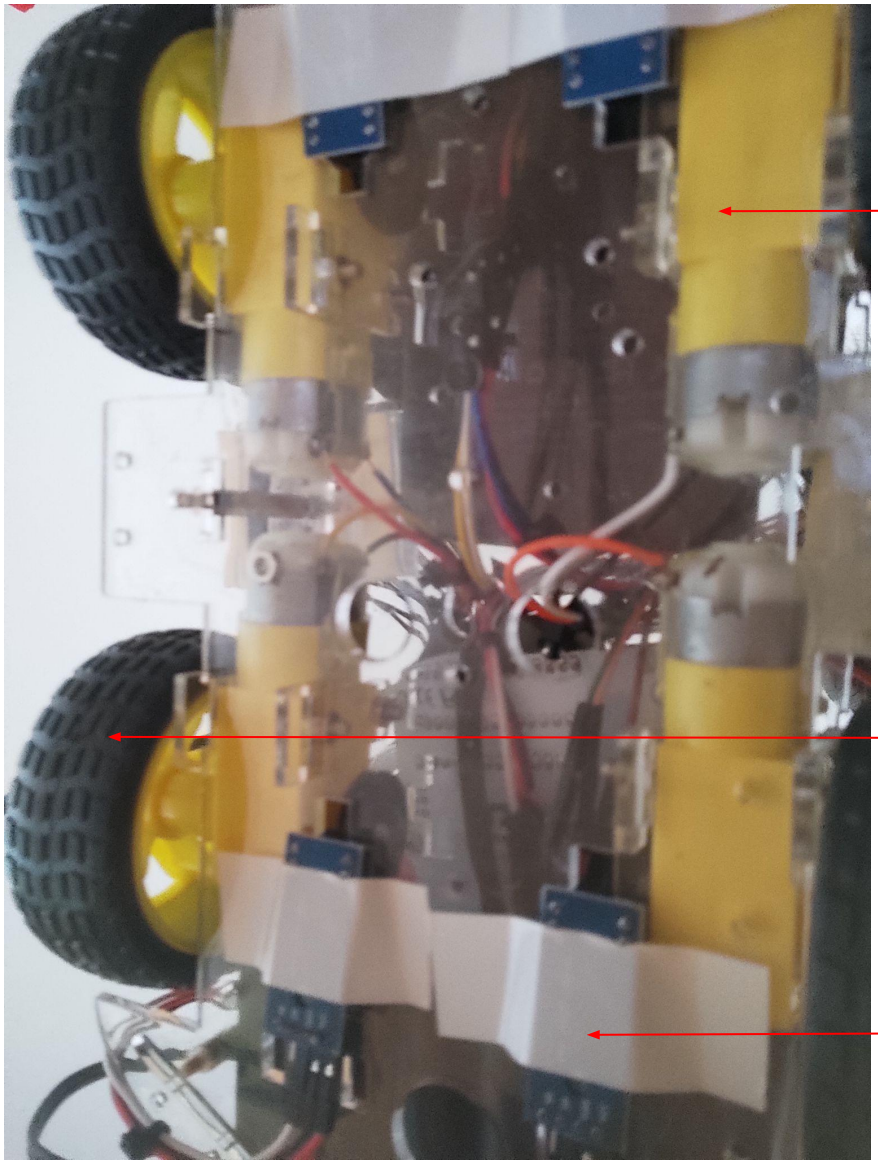
Principe

- On repère la position de l'athlète sur l'image de la caméra
- On calcule la distance entre l'athlète et le centre de la caméra
- On agit sur les moteurs pour rapprocher le robot de l'athlète

Présentation des composants



Présentation des composants

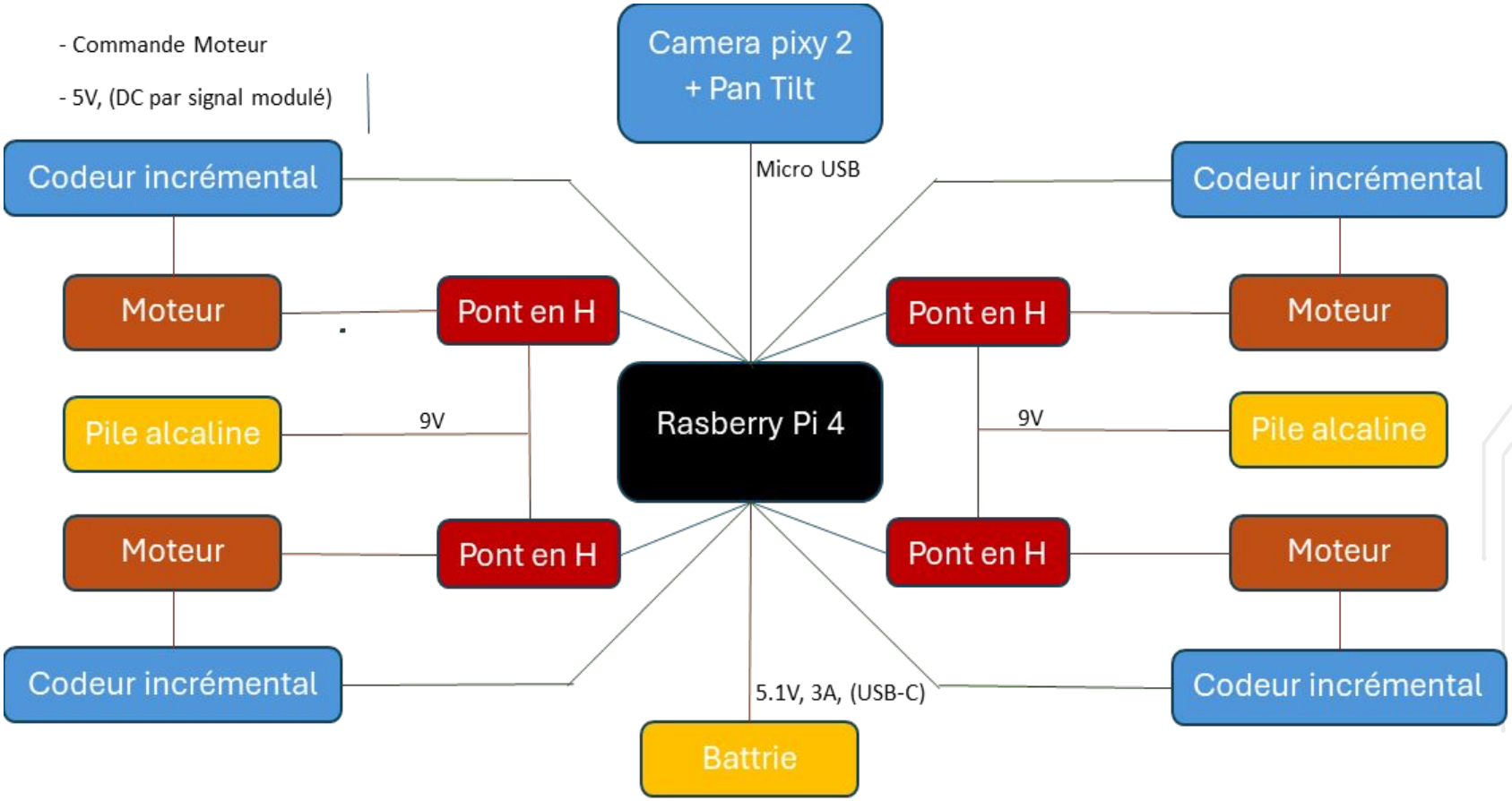


Motoréducteur
X4

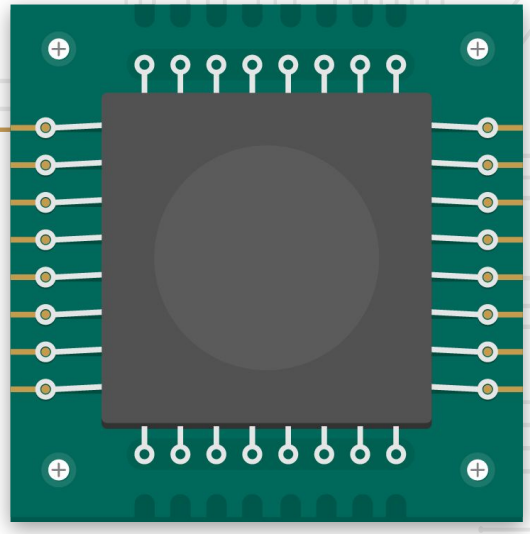
Roue
X4

LM393
Codeur incrémental
X4

Connexion des composants



04 Présentation de l'algorithme



- Présentation de l'erreur
- Choix des constantes
- Principe du code

Présentation de l'erreur

Erreur : 49 pixels

Choix de Constantes

PID : Correcteur caméra

- Kp = 0,7
- Ki = 0,2
- Kd = 0,1

PID : Correcteur moteurs

- Kp_ GD= 0,35
- Ki_GD = 0,1
- Kd_GD = 0,1

Test des moteurs

Par expérience, le robot n'avance pas lorsque le rapport cyclique est < 18%

L'asservissement en vitesse des quatres roues est identique car il est réalisé à partir d'un unique correcteur PID

Choix de Constantes

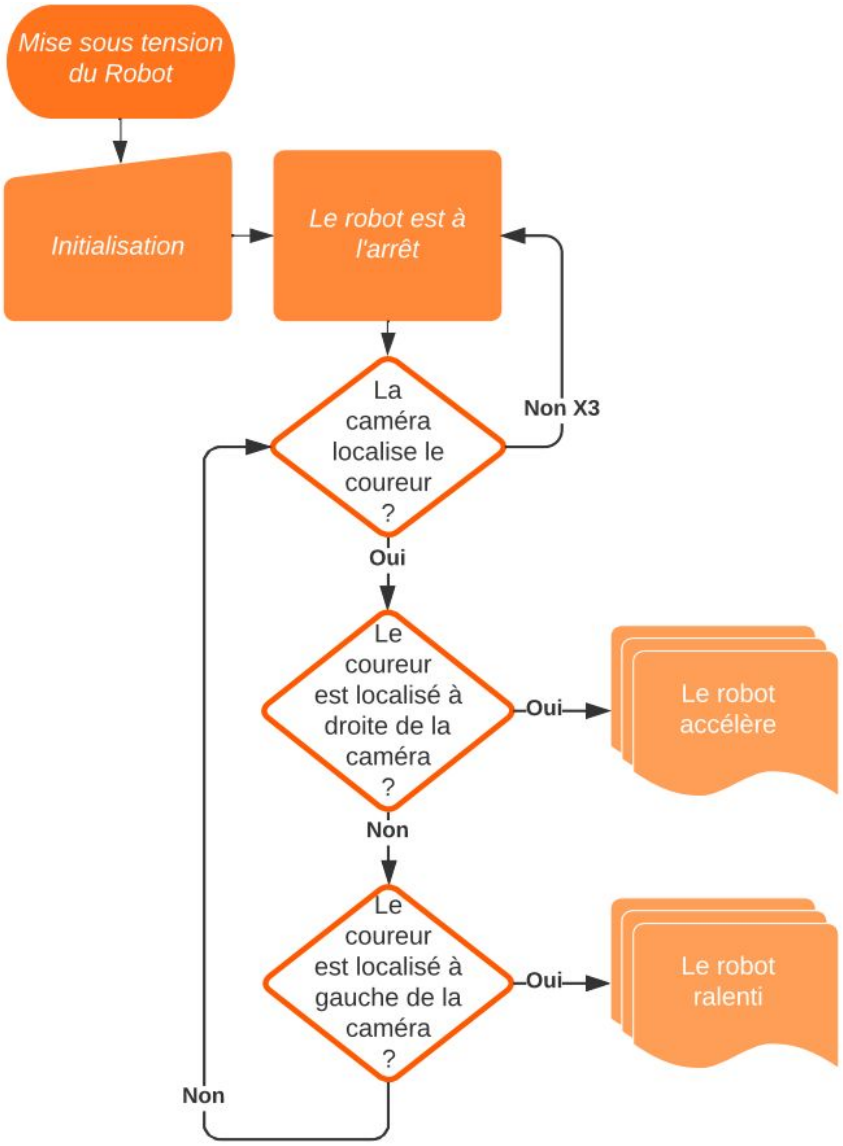
Méthode de réglage des paramètres PID :

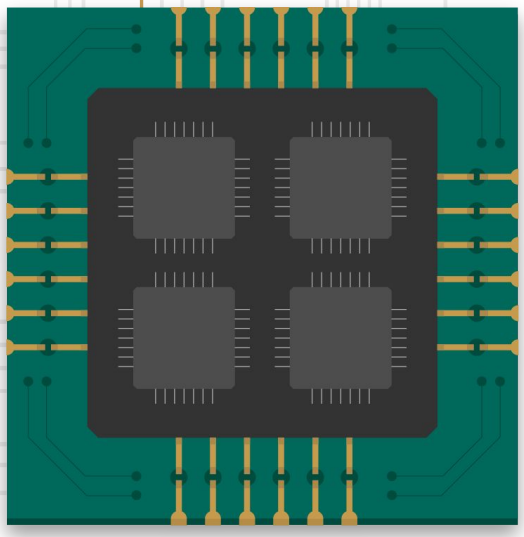
Méthode manuelle :

1. Si le système doit rester en marche, une méthode de réglage consiste à mettre les valeurs Intégrale (I) et Dérivée (D) à zéro.
2. Augmenter ensuite le gain Proportionnel (P) jusqu'à ce que la sortie oscille.
3. Puis, augmenter le gain de l'Intégrale et du Dérivée jusqu'à ce que cesse l'oscillation.

Rapport cyclique = Rapport cyclique calculé + Rapport cyclique minimal
Rapport cyclique minimal = 18%
Rapport cyclique calculé = Kp x erreur + Ki x somme erreur + Kd x delta erreur

Principe du code





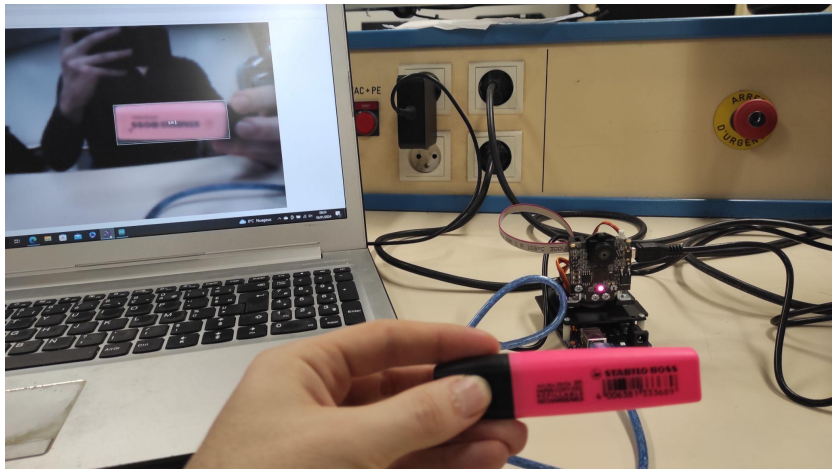
05
Expérience

- Principe de l'expérience
- Résultats
- Conclusion

Principe de l'expérience

Préparation

Faire apprendre à la caméra une couleur à l'aide du logiciel Piximon



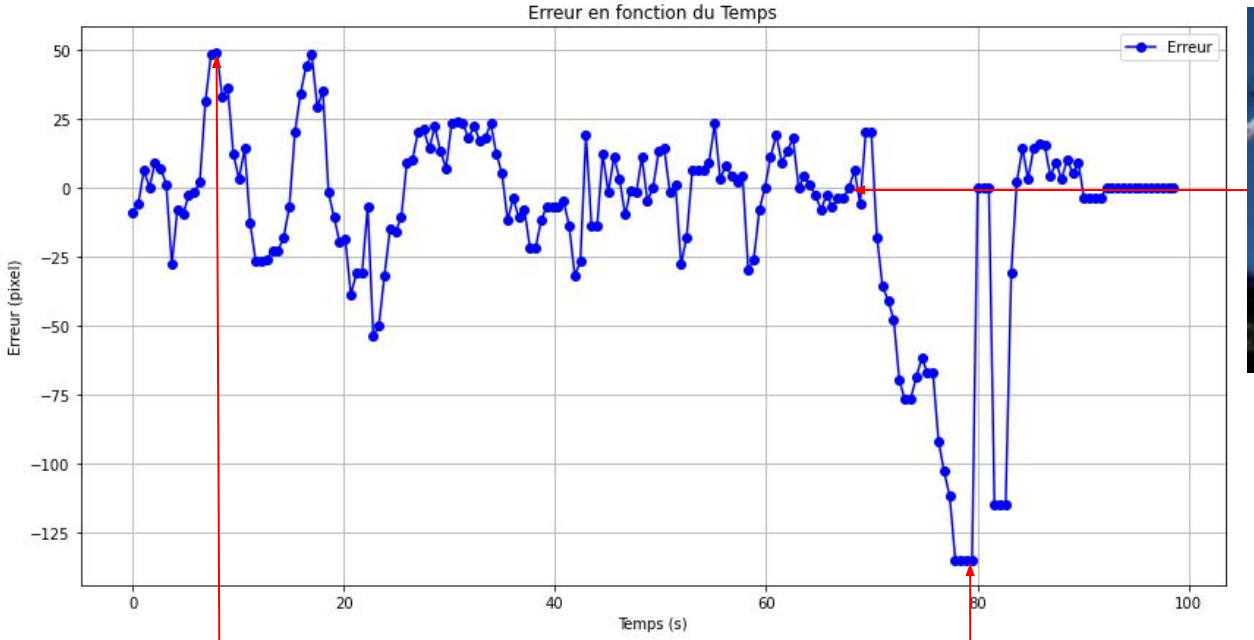
Lancement de l'expérience

Mettre le robot sous tension.
Connecter le robot à l'ordinateur.
Lancer le programme.

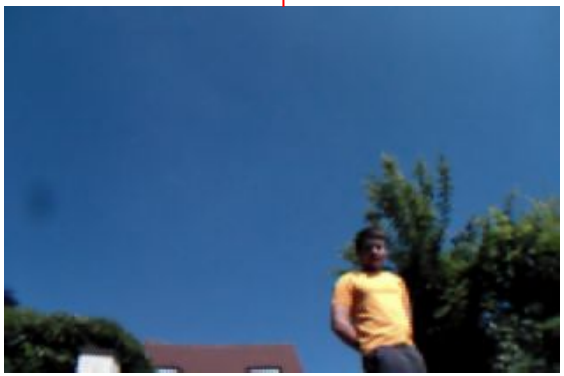
Récupération des résultats

Sortir de la vision de la caméra
Récupération des données à l'aide d'un ordinateur

Résultats



Erreur : 0 pixels

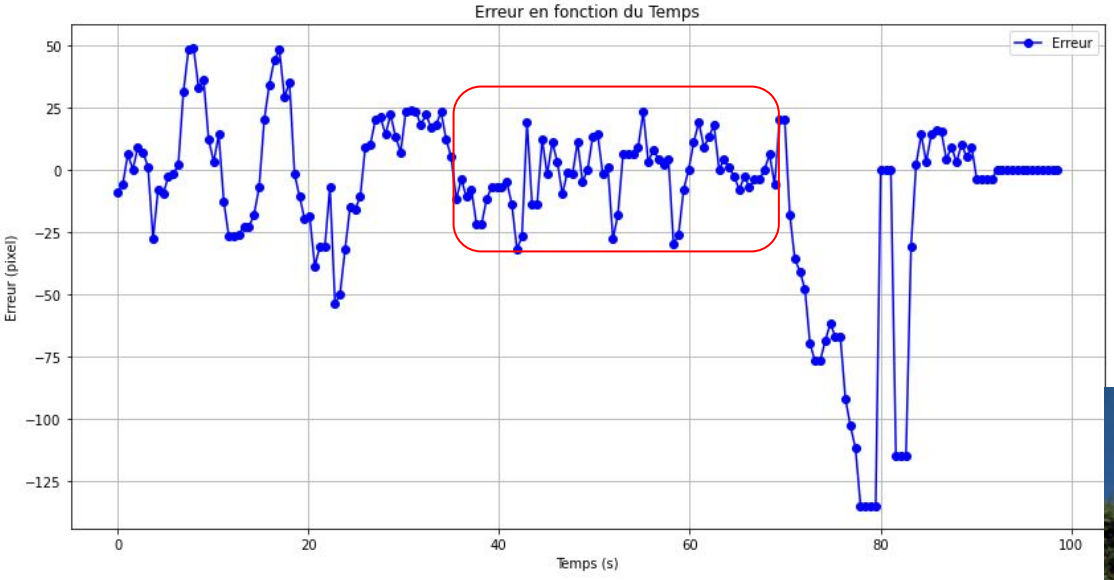


Erreur : 49 pixels

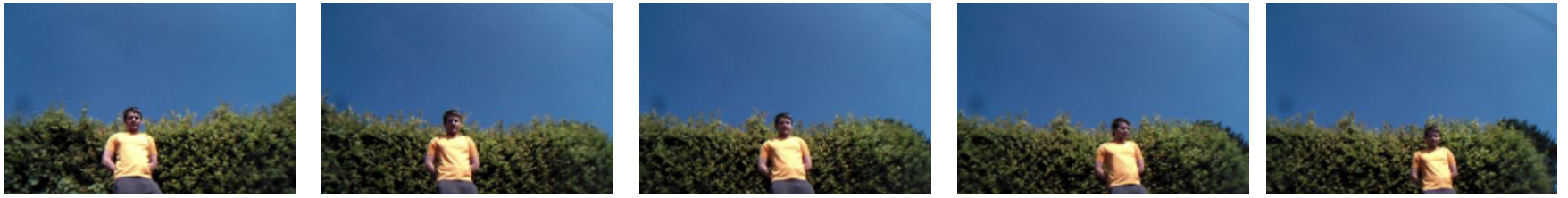
Erreur : -135 pixels

Déplacement en crabe

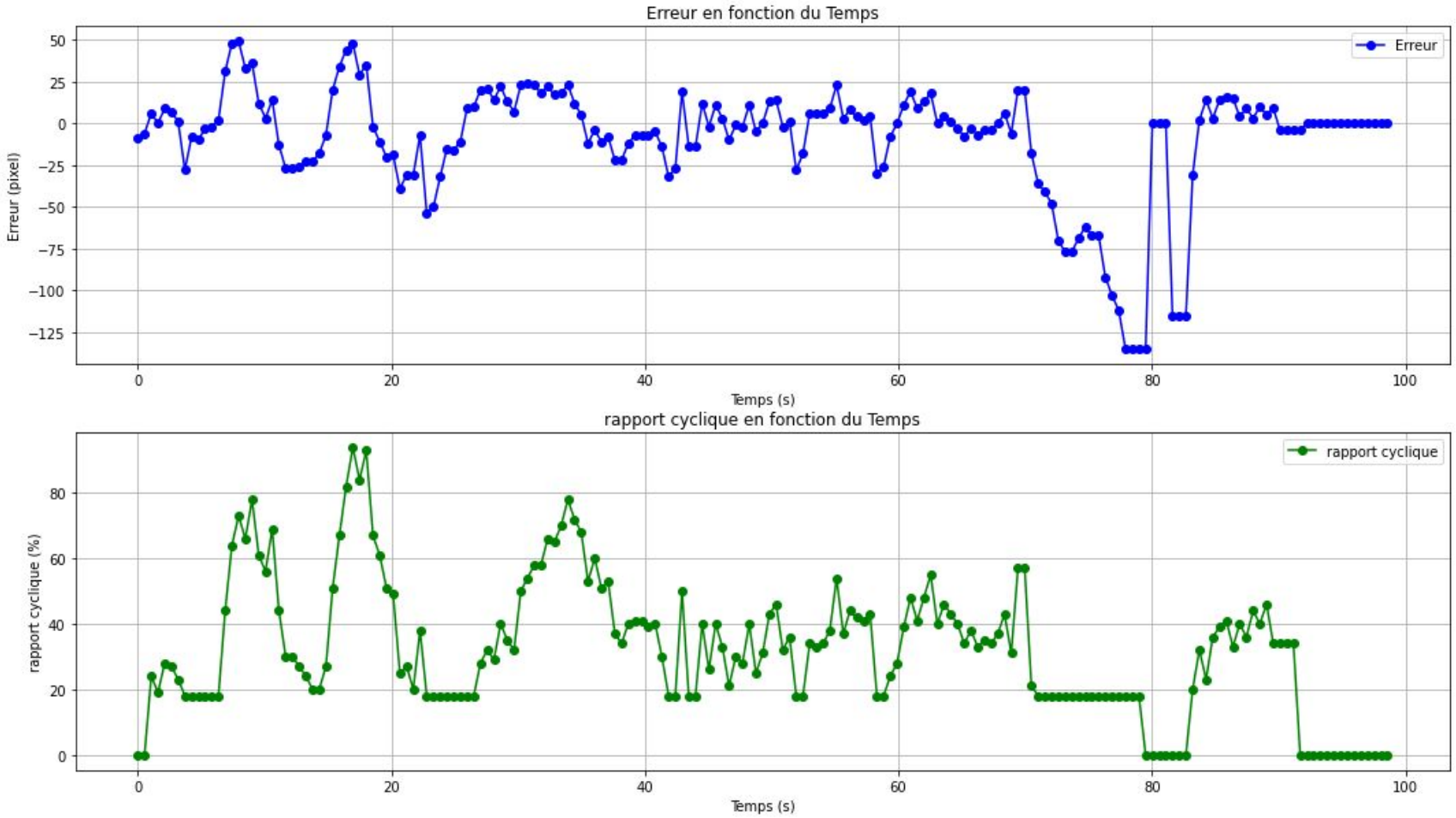
Résultats



Déplacement en crabe



Résultats



Le robot atteint sa vitesse maximale en 2,5 s
Temps moyen pour que le coureur soit centré : 0,75 s
Lors d'un suivie à vitesse constante l'erreur se situe entre 25 et -25 pixels
Lors de grand changement de vitesse le coureur est centré en : 3,5 s

Conclusion

Cahier des charges fonctionnels

1. Le robot me suit en me centrant sur l'image
2. Le temps moyen pour que le coureur soit centré est de 0,75 s.
3. Lors de grand changement de vitesse le coureur est centré en 3,5 s.
4. Le robot atteint sa vitesse maximale en 2,5 s.

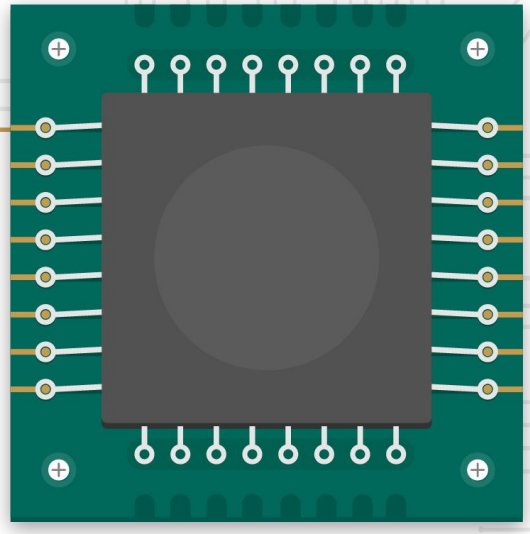
Limitation du système

- Sur de très longue distance, le code ne permet pas de maintenir le robot en ligne droite.
- La caméra ne fonctionne que lors d'un temps claire
- Les frottements ralentissent le Robot

Évolution du système

Utilisation de rails et d'un unique moteur.
Améliore le mouvement en ligne droite.
Augmente l'accélération et la vitesse du robot.
Cela se rapproche de la caméra de poursuite Speedcam.

06
Annexe



- Codes

Codes

```python
from __future__ import print_function

import os
import time
import RPi.GPIO as GPIO


import pixy
from ctypes import *
from pixy import *

# on efface les anciens fichier
if os.path.exists("*.ppm"):
  os.remove("*.ppm")

# variables globales
SV_Gauche_Av_cpt = 0
SV_Gauche_Ar_cpt = 0
SV_Droit_Av_cpt = 0
SV_Droit_Ar_cpt = 0

# Fonction incrementation des compteurs des capteurs vitesse
def SV_Gauche_Av_inc_cpt(self):
    global SV_Gauche_Av_cpt
    SV_Gauche_Av_cpt = SV_Gauche_Av_cpt + 1

def SV_Gauche_Ar_inc_cpt(self):
    global SV_Gauche_Ar_cpt
    SV_Gauche_Ar_cpt = SV_Gauche_Ar_cpt + 1

def SV_Droit_Av_inc_cpt(self):
    global SV_Droit_Av_cpt
    SV_Droit_Av_cpt = SV_Droit_Av_cpt + 1

def SV_Droit_Ar_inc_cpt(self):
    global SV_Droit_Ar_cpt
    SV_Droit_Ar_cpt = SV_Droit_Ar_cpt + 1


# Definition des pins GPIO utilisées
Moteur_Gauche_En = 13
Moteur_Droit_En = 12

Moteur_Vcc = 21

SV_Gauche_Av_D0 = 27
SV_Gauche_Ar_D0 = 22
SV_Droit_Av_D0 = 23
SV_Droit_Ar_D0 = 24

# Setup GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

GPIO.setup(Moteur_Gauche_En, GPIO.OUT)
GPIO.setup(Moteur_Droit_En, GPIO.OUT)

GPIO.setup(Moteur_Vcc, GPIO.OUT)

GPIO.setup(SV_Gauche_Av_D0, GPIO.IN)
GPIO.setup(SV_Gauche_Ar_D0, GPIO.IN)
GPIO.setup(SV_Droit_Av_D0, GPIO.IN)
GPIO.setup(SV_Droit_Ar_D0, GPIO.IN)

GPIO.output(Moteur_Vcc, GPIO.HIGH)

# Configuration des pins pour détection de la rotation des roues
# des capteurs vitesse
GPIO.add_event_detect(SV_Gauche_Av_D0, GPIO.FALLING, SV_Gauche_Av_inc_cpt)
GPIO.add_event_detect(SV_Gauche_Ar_D0, GPIO.FALLING, SV_Gauche_Ar_inc_cpt)
GPIO.add_event_detect(SV_Droit_Av_D0, GPIO.FALLING, SV_Droit_Av_inc_cpt)
GPIO.add_event_detect(SV_Droit_Ar_D0, GPIO.FALLING, SV_Droit_Ar_inc_cpt)


# Configuration PMW des pins pour commande moteur
Moteur_Gauche_Vitesse = GPIO.PWM(Moteur_Gauche_En, 100)
Moteur_Droit_Vitesse = GPIO.PWM(Moteur_Droit_En, 100)

# Configuration Pixy2
pixy.init()
pixy.change_prog("color_connected_components")
pixy.set_servos (500, 0)

# Definition detection bloc de la camera
class Blocks (Structure):
  _fields_ = [ ("m_signature", c_uint),
    ("m_x", c_uint),
    ("m_y", c_uint),
    ("m_width", c_uint),
    ("m_height", c_uint),
    ("m_angle", c_uint),
    ("m_index", c_uint),
    ("m_age", c_uint) ]

max_blocks = 10
blocks = BlockArray(max_blocks)
Increment = 0
nb_x = 316
time_stamp = 0
time_stamp_old = 0
cmd_min = 18
rayon_roue_robot = 0.03 #3 cm
nb_trous_capteur_vit = 20.0
pi = 3.14159

# Fonction de calcul de la vitesse à partir
# des compteurs des capteurs vitesse
def cpt2speed(cpt, deltaT):

    Vitesse_tr_min = (cpt/nb_trous_capteur_vit)*(60.0/deltaT)
    Vitesse_m_s = (Vitesse_tr_min*2*pi*rayon_roue_robot)/60
    Vitesse_km_h = Vitesse_m_s*3.6

    return Vitesse_km_h

# initialisations
debut_course = False

# PID
kp = 0.7
ki = 0.2
kd = 0.1

kp_GD = 0.35
ki_GD = 0.1
kd_GD = 0.1

erreur = 0
somme_erreur = 0
delta_erreur = 0
erreur_old = 0

erreur_GD = 0
somme_erreur_GD = 0
delta_erreur_GD = 0
erreur_old_GD = 0

cmd_calc = 0
cmd = 0
cmd_calc_GD = 0
cmd_G = 0
cmd_D = 0
old_pos = 0
nb_non_detect = 0
```

Codes

```
# Fichier de trace
ftrace = open("TraceTIPE.csv", "w")
ftrace.write("Kp ; " + str(kp) + "\n")
ftrace.write("Ki ; " + str(ki) + "\n")
ftrace.write("Kd ; " + str(kd) + "\n")
ftrace.write("Kp_GD ; " + str(kp_GD) + "\n")
ftrace.write("Ki_GD ; " + str(ki_GD) + "\n")
ftrace.write("Kd_GD ; " + str(kd_GD) + "\n")
ftrace.write("Increment;deltaT;debut_course;Frame_count;erreur;somme_erreur;delta_erreur;cmd_calc;cmd;erreur_GD;somme_erreur_GD;delta_erreur_GD;cmd_calc_GD;cmd_G;cmd_D;SV_Gauche_Av_cpt;SV_Gauche_Ar_cpt;SV_Droit_Av_cpt;SV_Droit_Ar_cpt;Vitesse_km_h_G;Vitesse_km_h_D\n")

print("Increment;deltaT;debut_course;Frame_count;erreur;somme_erreur;delta_erreur;cmd_calc;cmd;erreur_GD;somme_erreur_GD;delta_erreur_GD;cmd_calc_GD;cmd_G;cmd_D;SV_Gauche_Av_cpt;SV_Gauche_Ar_cpt;SV_Droit_Av_cpt;SV_Droit_Ar_cpt;Vitesse_km_h_G;Vitesse_km_h_D\n")

while True:

  Increment = Increment + 1

  # Attente prise en compte de la commande moteur précédente
  time.sleep(0.2)

  # Reset capteur vitesse
  SV_Gauche_Av_cpt = 0
  SV_Droit_Av_cpt = 0
  SV_Droit_Ar_cpt = 0
  SV_Gauche_Ar_cpt = 0

  # Temps de mesure des capteurs vitesse
  time_stamp_old = time_stamp
  time.sleep(0.3)
  time_stamp = time.perf_counter()
  deltaT = time_stamp - time_stamp_old # Verefication temps réellement écoulé


  # Acquisition des mesures des capteurs vitesse
  cptGAv = SV_Gauche_Av_cpt
  cptDAv = SV_Droit_Av_cpt
  cptDAr = SV_Droit_Ar_cpt
  cptGAr = SV_Gauche_Ar_cpt

  cptG = (cptGAv+cptGAr)
  cptD = (cptDAv+cptDAr)

  # Erreur PID GD
  erreur_GD =  cptG - cptD

  Vitesse_km_h_G = cpt2speed(cptG, deltaT)
  Vitesse_km_h_D = cpt2speed(cptD, deltaT)

  # Capteur position
  count = pixy.ccc_get_blocks (max_blocks, blocks)

  # Sauvegarde image capteur position
  # 1 fois sur 3 cause problème temps reel sinon
  if (Increment-1) % 3 == 0:
    pixy.sav_raw_frame('out_test_' + str(Increment))

  print("count : " + str(count))

  # Si au moins un bloc détecté par la caméra
  if count > 0:

    nb_non_detect = 0

    # Erreur PID
    erreur = blocks[0].m_x - nb_x/2

    # Détection début de course
    if (erreur > 0) & (debut_course == False):

        debut_course = True


    if debut_course == True:

        # PID moteur
        somme_erreur = somme_erreur + erreur
        delta_erreur = erreur - erreur_old

        cmd_calc = kp*erreur + ki*somme_erreur + kd*delta_erreur

        if cmd_calc < 0:
          cmd_calc = 0

        # Recalage
        cmd = cmd_calc + cmd_min

        if cmd > 100:
          cmd = 100

        erreur_old = erreur

  else:

    nb_non_detect = nb_non_detect + 1

     # Reset si la camera ne detecte aucun bloc plus de 3 fois de suite
    if nb_non_detect > 3:

        #Coureur perdu : Reset
        erreur = 0
        somme_erreur = 0
        delta_erreur = 0
        erreur_old = 0
        erreur_GD = 0
        somme_erreur_GD = 0
        delta_erreur_GD = 0
        erreur_old_GD = 0
        cmd_calc = 0
        cmd = 0
        cmd_calc_GD = 0
        cmd_G = 0
        cmd_D = 0
        debut_course = False


  # PID GD
  if debut_course == True:

    somme_erreur_GD = somme_erreur_GD + erreur_GD
    delta_erreur_GD = erreur_GD - erreur_old_GD

    cmd_calc_GD = kp_GD*erreur_GD + ki_GD*somme_erreur_GD + kd_GD*delta_erreur_GD

    erreur_old_GD = erreur_GD

    # Commande moteur
    cmd_G = round(cmd)
    cmd_D = round(cmd + cmd_calc_GD)

    if cmd_D < cmd_min:
        cmd_D = cmd_min

    if cmd_D > 100:
        cmd_G = cmd_G - (cmd_D - 100)
        cmd_D = 100

  Moteur_Gauche_Vitesse.start(cmd_G)
  Moteur_Droit_Vitesse.start(cmd_D)
```

Codes

```
283     # Sortie des donnees
284     print(str(Increment) + ";" + str(deltaT) + ";" + str(debut_course) + ";" +
        str(count) + ";" + str(erreur) + ";" + str(somme_erreur) + ";" + str(delta_erreur)
        + ";" + str(cmd_calc) + ";" + str(cmd) + ";"  + str(erreur_GD) + ";"  +
        str(somme_erreur_GD) + ";"  + str(delta_erreur_GD) + ";"  + str(cmd_calc_GD) +
        ";"  + str(cmd_G) + ";"  + str(cmd_D) + ";" + str(SV_Gauche_Av_cpt) + ";"  +
        str(SV_Gauche_Ar_cpt) + ";" + str(SV_Droit_Av_cpt) + ";" + str(SV_Droit_Ar_cpt) +
        ";" + str(Vitesse_km_h_G) + ";" + str(Vitesse_km_h_D) + "\n")
285     ftrace.write(str(Increment) + ";" + str(deltaT) + ";" + str(debut_course) + ";" +
        str(count) + ";" + str(erreur) + ";" + str(somme_erreur) + ";" + str(delta_erreur)
        + ";" + str(cmd_calc) + ";" + str(cmd) + ";" + str(erreur_GD) + ";"  +
        str(somme_erreur_GD) + ";"  + str(delta_erreur_GD) + ";"  + str(cmd_calc_GD) +
        ";"  + str(cmd_G) + ";"  + str(cmd_D) + ";"+ str(SV_Gauche_Av_cpt) + ";"  +
        str(SV_Gauche_Ar_cpt) + ";" + str(SV_Droit_Av_cpt) + ";" + str(SV_Droit_Ar_cpt) +
        ";" + str(Vitesse_km_h_G) + ";" + str(Vitesse_km_h_D) + "\n")
286     ftrace.flush()
287
288
289
290
291
292
```