ESCUELA UNIVERSITARIA POLITÉCNICA

DE LA ALMUNIA DE DOÑA GODINA (ZARAGOZA)

ANEXOS

Análisis y puesta en marcha de una plataforma triaxial como demostrador tecnológico

[424.19.38]

Autor: Raúl López Martín

Director: Juan Diego Jaria Gazol / José Manuel Rodriguez Fortún

Fecha: 27/11/2019

INDICE DE CONTENIDO

ANEXO 1. (CÓDIGO)

1.1. PYTHON

A través de este código se realiza la detección y el control de la pelota.

```
import cv2
import numpy as np
import time
import imutils
import tkinter as tk
import tkinter.messagebox
from PIL import Image, ImageTk
import serial
import serial.tools.list_ports
from math import *
x_a = np.zeros((12000, 1))
y_a = np.zeros((12000, 1))
Ix_a = np.zeros((12000, 1))
Iy_a = np.zeros((12000, 1))
consigneX_a = np.zeros((12000, 1))
consigneY_a = np.zeros((12000, 1))
alpha_a = np.zeros((12000, 1))
beta_m_a = np.zeros((12000, 1))
errorX_a = np.zeros((12000, 1))
errorY_a = np.zeros((12000, 1))
preX_a = np.zeros((12000, 1))
preY_a = np.zeros((12000, 1))
time_a = np.zeros((12000, 1))
```

```
f = 0

c = 0

write_index=0

lines = open("/Users/rlopez/Desktop/Codigo/data.txt").read().splitlines()

lines = lines[:-11]     #enlève les 11 dernières lignes du fichier

lines = lines[1:]       #enlève la première ligne du fichier

dataDict = {}

camHeight =480

camWidth = 640

cam = cv2.VideoCapture(0)

cam.set(3,camWidth)

cam.set(4,camHeight)

getPixelColor = True #False

H,S,V = 16,131,229

mouseX,mouseY = 0,0

for i in range(0,len(lines)):

        key, value = lines[i].split("#")

        alpha, beta = key.split("|")

        angleA, angleB, angleC = value.split("|")

        dataDict[(float(alpha),float(beta))] = (float(angleA), float(angleB), float(angleC))

controllerWindow = tk.Tk()

controllerWindow.title("fenêtre de contrôle")

controllerWindow.geometry("820x500")

controllerWindow["bg"]="white"

controllerWindow.resizable(0, 0)

videoWindow = tk.Toplevel(controllerWindow)

videoWindow.title("retour caméra")

videoWindow.resizable(0, 0)  #empeche de modifier les dimensions de la fenêtre
```

```
lmain = tk.Label(videoWindow)

lmain.pack()

videoWindow.withdraw()

graphWindow = tk.Toplevel(controllerWindow)

graphWindow.title("Position en fonction du temps")

graphWindow.resizable(0, 0)

graphCanvas = tk.Canvas(graphWindow,width=camHeight+210,height=camHeight)

graphCanvas.pack()

graphWindow.withdraw()

pointsListCircle = []

def createPointsListCircle(rayon):  global pointsListCircle

            for angle in range(0,360):

                angle=angle-90

pointsListCircle.append([rayon*cos(radians(angle))+240,rayon*sin(radians(angle))+2
40])

createPointsListCircle(50)

pointsListEight = []

def createPointsListEight(rayon):

 global pointsListEight

 for angle in range(270,270+360):


pointsListEight.append([rayon*cos(radians(angle))+240,rayon*sin(radians(angle))+2
40+rayon])

 for angle in range(360,0,-1):

 angle=angle+90

pointsListEight.append([rayon*cos(radians(angle))+240,rayon*sin(radians(angle))+2
40-rayon])

createPointsListEight(80)

drawCircleBool = False
```

```
def startDrawCircle():
 global drawCircleBool, drawEightBool, consigneX, consigneY
  if drawCircleBool == False:
        drawCircleBool = True
        BballDrawCircle["text"] = "Centrer la bille"
    else:
        drawCircleBool = False
        consigneX, consigneY = 240, 240
        sliderCoefP.set(sliderCoefPDefault)
        BballDrawCircle["text"] = "Faire tourner la bille en cercle"

drawEightBool = False
def startDrawEight():
    global drawEightBool, drawCircleBool, consigneX, consigneY
    if drawEightBool == False:
        drawEightBool = True
        BballDrawEight["text"] = "Centrer la bille"
    else:
        drawEightBool = False
        consigneX, consigneY = 240, 240
        sliderCoefP.set(sliderCoefPDefault)
        BballDrawEight["text"] = "Faire tourner la bille en huit"

pointCounter = 0
def drawWithBall():
    global pointCounter, consigneX, consigneY
    if drawCircleBool == True:
        #sliderCoefP.set(15)
```

```
                if pointCounter >= len(pointsListCircle):
                    pointCounter = 0
                point = pointsListCircle[pointCounter]
                consigneX, consigneY = point[0], point[1]
                pointCounter += 7
            if drawEightBool == True:
                #sliderCoefP.set(15)
                if pointCounter >= len(pointsListEight):
                    pointCounter = 0
                point = pointsListEight[pointCounter]
                consigneX, consigneY = point[0], point[1]
                pointCounter += 7
        def setConsigneWithMouse(mousePosition):
            global consigneX, consigneY
            if mousePosition.y > 10:
                refreshGraph()
                consigneX,consigneY = mousePosition.x,mousePosition.y


def getMouseClickPosition(mousePosition):
            global mouseX,mouseY
            global getPixelColor
            mouseX,mouseY = mousePosition.x,mousePosition.y
            getPixelColor = True
           showVideoWindow = False
           def showCameraFrameWindow():
            global showVideoWindow, showGraph
            global BRetourVideoTxt
            if showVideoWindow == False:
                if showGraph == True:
```

```
          graphWindow.withdraw()
          showGraph = False
          BafficherGraph["text"] = "Afficher graphique"
        videoWindow.deiconify()
        showVideoWindow = True
        BRetourVideo["text"] = "Cacher le retour vidéo "
      else:
        videoWindow.withdraw()
        showVideoWindow = False
        BRetourVideo["text"] = "Afficher le retour vidéo"
showCalqueCalibrationBool = False
def showCalqueCalibration():
global showCalqueCalibrationBool
 showCalqueCalibrationBool = not showCalqueCalibrationBool
    showGraph = False
    def showGraphWindow():
      global showGraph, showVideoWindow
      global BafficherGraphif showGraph == False:
        if showVideoWindow == True:
          videoWindow.withdraw()
          showVideoWindow = False
          BRetourVideo["text"] = "Afficher le retour vidéo"
        showGraph = True
        BafficherGraph["text"] = "Cacher graphique "
      else:
        showGraph = False
        BafficherGraph["text"] = "Afficher graphique"
```

```
t = 480
    consigneY = 240
    consigneX = 240
    x=0
    y=0
    def paintGraph():
        global t,consigneY,x,y,prevX,prevY,alpha,prevAlpha
        global showGraphPositionX,showGraphPositionY, showGraphAlpha, showGraphBeta_m
        if showGraph == True:
            graphWindow.deiconify()

            if showGraphPositionX.get() == 1:
                graphCanvas.create_line(t-3,prevX,t,x, fill="#b20000", width=2)
            if showGraphPositionY.get() == 1:
                graphCanvas.create_line(t-3,prevY,t,y, fill="#0069b5", width=2)
            if showGraphAlpha.get() == 1:
                graphCanvas.create_line(t-3,240-prevAlpha*3,t,240-alpha*3, fill="#8f0caf", width=2)
            if showGraphBeta_m.get() == 1:
                graphCanvas.create_line(t-3,240-prevBeta,t,240-beta_m, fill="#476042", width=2)
            if t >= 480:
                t = 0
                graphCanvas.delete("all")
                graphCanvas.create_line(3,3,480,3,fill="black", width=3)
                graphCanvas.create_line(3,480,480,480,fill="black", width=3)
                graphCanvas.create_line(3,3,3,480,fill="black", width=3)
```

```
                    graphCanvas.create_line(480,3,480,480,fill="black", width=3)

                    graphCanvas.create_line(550,32,740,32,fill="#b20000", width=5)

                    graphCanvas.create_line(550,53,740,53,fill="#0069b5", width=5)

                    graphCanvas.create_line(550,73,740,73,fill="#8f0caf", width=5)

                    graphCanvas.create_line(550,94,740,94,fill="#476042", width=5)


                    if showGraphPositionX.get() == 1:

                        graphCanvas.create_line(3,consigneX,480,consigneX,fill="#ff7777",
width=2)

                    if showGraphPositionY.get() == 1:

                        graphCanvas.create_line(3,consigneY,480,consigneY,fill="#6f91f7",
width=2)

                t += 3

            else:

                graphWindow.withdraw()



        def refreshGraph():

            global t

            t=480


        def endProgam():

            global x_a, y_a, consigneX_a, consigneY_a, Ix_a, Iy_a, alpha_a, beta_m_a,
errorX_a, errorY_a, preX_a, preY_a, time_a

            fic = open("archivoX.txt","w")

            fic.writelines("%f\n" % f for f in x_a)

            fic.close()

            fic1 = open("archivoY.txt","w")

            fic1.writelines("%f\n" % f for f in y_a)
```

```
    fic1.close()
    fic2 = open("archivoCX.txt","w")
    fic2.writelines("%f\n" % f for f in consigneX_a)
    fic2.close()
    fic3 = open("archivoCY.txt","w")
    fic3.writelines("%f\n" % f for f in consigneY_a)
    fic3.close()
    fic4 = open("archivoIx.txt","w")
    fic4.writelines("%f\n" % f for f in Ix_a)
    fic4.close()
    fic5 = open("archivoIy.txt","w")
    fic5.writelines("%f\n" % f for f in Iy_a)
    fic5.close()
    fic6 = open("archivoalpha.txt","w")
    fic6.writelines("%f\n" % f for f in alpha_a)
    fic6.close()
    fic7 = open("archivobeta_m.txt","w")
    fic7.writelines("%f\n" % f for f in beta_m_a)
    fic7.close()
    fic8 = open("Error_x.txt","w")
    fic8.writelines("%f\n" % f for f in errorX_a)
    fic8.close()
    fic9 = open("Error_y.txt","w")
    fic9.writelines("%f\n" % f for f in errorY_a)
    fic9.close()
    fic10 = open("Prev_x.txt","w")
    fic10.writelines("%f\n" % f for f in preX_a)
    fic10.close()
    fic11 = open("Prev_y.txt","w")
```

```
            fic11.writelines("%f\n" % f for f in preY_a)

            fic11.close()

            fic12 = open("time.txt","w")

            fic12.writelines("%f\n" % f for f in time_a)

            fic12.close()

            controllerWindow.destroy()


        sliderHDefault = 50

        sliderSDefault = 50

        sliderVDefault = 50

        sliderCoefPDefault = 0

        sliderCoefIDefault = 0

        sliderCoefDDefault = 0


        def resetSlider():

            sliderH.set(sliderHDefault)

            sliderS.set(sliderSDefault)

            sliderV.set(sliderVDefault)

            sliderCoefP.set(sliderCoefPDefault)

            sliderCoefI.set(sliderCoefIDefault)

            sliderCoefD.set(sliderCoefDDefault)

            sommeErreurX=0

            sommeErreurY=0

def donothing():

            pass

def rangerPlateau():

            if arduinoIsConnected == True:

                if tkinter.messagebox.askokcancel("Avertissement", "Pensez à retirer le plateau."):
```

```
            print("abaissement des bras")
            ser.write(("descendreBras\n").encode())
    else:
        if tkinter.messagebox.askokcancel("Avertissement","L'Arduino n'est pas connecté"):
            donothing()
def eleverPlateau():
    global alpha
    if arduinoIsConnected == True:
        if tkinter.messagebox.askokcancel("Avertissement", "Pensez à retirer le plateau."):
            print("Elevation des bras")
            ser.write((str(dataDict[(0,0)])+"\n").encode())
            alpha = 0
    else:
        if tkinter.messagebox.askokcancel("Avertissement","L'Arduino n'est pas connecté"):
            donothing()
def servosTest():
    if arduinoIsConnected == True:
        if tkinter.messagebox.askokcancel("Avertissement", "Le plateau doit être en place."):
            for i in range(2):
                beta = 0
                alpha = 35
                while beta < 360:
                    ser.write((str(dataDict[(alpha,beta)])+"\n").encode())
                    ser.flush()
                    time.sleep(0.002)
                    beta = round(beta+0.2,2)
                print(alpha,beta)
            time.sleep(1)
            ser.write((str(dataDict[(0,0)])+"\n").encode())
    else:
        if tkinter.messagebox.askokcancel("Avertissement","L'Arduino n'est pas connecté"):
            donothing()

arduinoIsConnected = False
def connectArduino():
    global ser
    global label
    global arduinoIsConnected
    ports = list(serial.tools.list_ports.comports())
    for p in ports:
        if "Arduino" in p.description:
            ser = serial.Serial(p[0], 19200, timeout=1)
            time.sleep(1) #give the connection a second to settle
            label.configure(text="Arduino connecté", fg="#36db8b")
            arduinoIsConnected = True

startBalanceBall = False
def startBalance():
    global startBalanceBall
    if arduinoIsConnected == True:
        if startBalanceBall == False:
            startBalanceBall = True
            BStartBalance["text"] = "Arrêter"
        else:
            startBalanceBall = False
            BStartBalance["text"] = "Commencer"
    else:
        if tkinter.messagebox.askokcancel("Avertissement","L'Arduino n'est pas connecté"):
            donothing()

FrameVideoControl = tk.LabelFrame(controllerWindow, text="Vidéo contrôle")
FrameVideoControl.place(x=20,y=20,width=380)
BRetourVideo = tk.Button(FrameVideoControl, text="Afficher le retour vidéo", command=showCameraFrameWindow)
BRetourVideo.pack()
BPositionCalibration = tk.Button(FrameVideoControl, text="Calque", command=showCalqueCalibration)
BPositionCalibration.place(x=290,y=0)

sliderH = tk.Scale(FrameVideoControl, from_=0, to=100, orient="horizontal", label="Sensibilité H", length=350, tickinterval = 10)
sliderH.set(sliderHDefault)
sliderH.pack()
sliderS = tk.Scale(FrameVideoControl, from_=0, to=100, orient="horizontal", label="Sensibilité S", length=350, tickinterval = 10)
sliderS.set(sliderSDefault)
sliderS.pack()
sliderV = tk.Scale(FrameVideoControl, from_=0, to=100, orient="horizontal", label="Sensibilité V", length=350, tickinterval = 10)
sliderV.set(sliderVDefault)
sliderV.pack()

FrameServosControl = tk.LabelFrame(controllerWindow, text="Servos contrôle")
```

```
FrameServosControl.place(x=20,y=315,width=380)
BAbaissementPlateau = tk.Button(FrameServosControl, text="Ranger les bras", command=rangerPlateau)
BAbaissementPlateau.pack()
BElevationBras = tk.Button(FrameServosControl, text="Mettre en place le plateau", command=eleverPlateau)
BElevationBras.pack()
BTesterServos = tk.Button(FrameServosControl, text="Tester les servomoteurs", command=servosTest)
BTesterServos.pack()
BStartBalance = tk.Button(FrameServosControl, text="Démarrer", command=startBalance, highlightbackground = "#36db8b")
BStartBalance.pack()

FramePIDCoef = tk.LabelFrame(controllerWindow, text="PID coefficients")
FramePIDCoef.place(x=420,y=20,width=380)
BafficherGraph = tk.Button(FramePIDCoef, text="Afficher graphique", command=showGraphWindow)
BafficherGraph.pack()
sliderCoefP = tk.Scale(FramePIDCoef, from_=0, to=15, orient="horizontal", label="P", length=350, tickinterval = 3, resolution=0.01)
sliderCoefP.set(sliderCoefPDefault)
sliderCoefP.pack()
sliderCoefI = tk.Scale(FramePIDCoef, from_=0, to=1, orient="horizontal", label="I", length=350, tickinterval = 0.2, resolution=0.001)
sliderCoefI.set(sliderCoefIDefault)
sliderCoefI.pack()
sliderCoefD = tk.Scale(FramePIDCoef, from_=0, to=10, orient="horizontal", label="D", length=350, tickinterval = 2, resolution=0.01)
sliderCoefD.set(sliderCoefDDefault)
sliderCoefD.pack()
```

```
FrameBallControl = tk.LabelFrame(controllerWindow, text="Bille contrôle")
FrameBallControl.place(x=420,y=315,width=380, height= 132)
BballDrawCircle = tk.Button(FrameBallControl, text="Faire tourner la bille en cercle", command=startDrawCircle)
BballDrawCircle.pack()
BballDrawEight = tk.Button(FrameBallControl, text="Faire tourner la bille en huit", command=startDrawEight)
BballDrawEight.pack()

label = tk.Label(controllerWindow, text="Arduino déconnecté   ", fg="red", anchor="ne")
label.pack(fill="both")
BReset = tk.Button(controllerWindow, text = "Reset", command = resetSlider)
BReset.place(x=20, y=460)
BConnect = tk.Button(controllerWindow, text = "Connexion", command = connectArduino, background="black")
BConnect.place(x=100, y=460)
BQuit = tk.Button(controllerWindow, text = "Quitter", command = endProgam)
BQuit.place(x=730, y=460)

showGraphPositionX = tk.IntVar()
showGraphPositionX.set(1)
CheckbuttonPositionX = tk.Checkbutton(graphWindow, text="Position en X", variable=showGraphPositionX, command=refreshGraph)
CheckbuttonPositionX.place(x=500,y=20)
showGraphPositionY = tk.IntVar()
showGraphPositionY.set(1)
CheckbuttonPositionY = tk.Checkbutton(graphWindow, text="Position en Y", variable=showGraphPositionY, command=refreshGraph)
```

```
        CheckbuttonPositionY.place(x=500,y=40)

        showGraphAlpha = tk.IntVar()

        CheckbuttonAlpha = tk.Checkbutton(graphWindow, text="Inclinaison du plateau", variable=showGraphAlpha, command=refreshGraph)

        CheckbuttonAlpha.place(x=500,y=60)

        showGraphBeta_m = tk.IntVar()

        CheckbuttonBeta_m = tk.Checkbutton(graphWindow, text="Inclinacion plataforma Beta", variable=showGraphBeta_m, command=refreshGraph)

        CheckbuttonBeta_m.place(x=500,y=80)


        videoWindow.protocol("WM_DELETE_WINDOW",donothing)

        videoWindow.bind("<Button-2>",getMouseClickPosition)

        videoWindow.bind("<Button-1>",setConsigneWithMouse)

        sommeErreurX = 1

        sommeErreurY = 1

        timeInterval = 1

        alpha, beta, prevAlpha, prevBeta = 0,0,0,0

        omega = 0.2

        indice_prueba = 1


        def PIDcontrol(ballPosX, ballPosY, prevBallPosX, prevBallPosY, consigneX, consigneY):

            global omega

            global sommeErreurX, sommeErreurY

            global alpha, beta, prevAlpha, prevBeta, gama, beta_m

            global startBalanceBall, arduinoIsConnected

            global x_a, write_index, y_a, consigneX_a, consigneY_a, Ix, Iy, alpha_a, beta_m_a, prevX, prevY, x, y, errorX_a, errorY_a, preX_a, preY_a


            gama = 0
```

```
    Kp = sliderCoefP.get()

    Ki = sliderCoefI.get()

    Kd = sliderCoefD.get()


    Ix = Kp*(consigneX-ballPosX) + Ki*sommeErreurX + Kd*((prevBallPosX-ballPosX)/0.0333)

    Iy = Kp*(consigneY-ballPosY) + Ki*sommeErreurY + Kd*((prevBallPosY-ballPosY)/0.0333)


    Ix = round(Ix/10000, 4)

    Iy = round(Iy/10000, 4)


    if Ix == 0 and Iy == 0:

        alpha = 0

        beta = 0

        gama = 0


   elif Ix != 0 and sqrt(Ix**2 + Iy**2) < 1:

        gama = atan(Iy/Ix)

        alpha = asin(sqrt(Ix**2 + Iy**2))

        gama = degrees(gama)

        alpha = degrees(alpha)


        if Ix < 0 and Iy >= 0:

            beta = abs(gama)

        elif Ix > 0 and Iy >= 0:

            beta = 180-abs(gama)

        elif Ix > 0 and Iy <= 0:

            beta = 180+abs(gama)
        elif Ix < 0 and Iy <= 0:
            beta = 360-abs(gama)

    elif Ix == 0 and sqrt(Ix**2 + Iy**2) < 1:
        if Iy > 0:
            beta = 90
            alpha = asin(sqrt(Ix**2 + Iy**2))
        elif Iy < 0:
            beta = 270
            alpha = asin(sqrt(Ix**2 + Iy**2))
        alpha = degrees(alpha)

    elif Ix != 0 and sqrt(Ix**2 + Iy**2) > 1:
        beta = degrees(atan(Iy/Ix))
        alpha = 30
        if Ix < 0 and Iy >= 0:
            beta = abs(gama)
        elif Ix > 0 and Iy >= 0:
            beta = 180-abs(gama)
        elif Ix > 0 and Iy <= 0:
            beta = 180+abs(gama)
        elif Ix < 0 and Iy <= 0:
            beta = 360-abs(gama)

    elif Ix == 0 and sqrt(Ix**2 + Iy**2) > 1:
        alpha = 30
        if Iy > 0:
            beta = 90
        elif Iy < 0:
```

```
                    beta = 270
  if alpha > 30:
                alpha = 30
         alpha = round(round(alpha / 0.2) * 0.2, -int(floor(log10(0.2))))   ## permet d'arrondire avec 0.2 de précision
            beta = round(round(beta / 0.2) * 0.2, -int(floor(log10(0.2))))
            beta_m = beta - 180
            if beta_m >= 0:
                beta_m = beta_m
                beta_m = round(round(beta_m / 0.2) * 0.2, -int(floor(log10(0.2))))
            else:
                beta_m = 360 - abs(beta_m)
                beta_m= round(round(beta_m / 0.2) * 0.2, -int(floor(log10(0.2))))

            if alpha <= 30 and beta <= 360 and arduinoIsConnected == True and startBalanceBall == True:
                ser.write((str(dataDict[(alpha,beta_m)])+"\n").encode())

            alpha = prevAlpha * omega + (1-omega) * alpha
            beta_m = prevBeta * omega + (1-omega) * beta_m

            write_index=write_index+1
            if write_index<12000:
                    x_a[write_index] = ballPosX

            if write_index<12000:
                    y_a[write_index] = ballPosY

            if write_index<12000:
```

```
            consigneX_a[write_index] = consigneX

        if write_index<12000:
            consigneY_a[write_index] = consigneY

        if write_index<12000:
            Ix_a[write_index] = Ix

        if write_index<12000:
            Iy_a[write_index] = Iy

        if write_index<12000:
            alpha_a[write_index] = alpha

        if write_index<12000:
            beta_m_a[write_index] = beta_m

        if write_index<12000:
            errorX_a[write_index] = sommeErreurX

        if write_index<12000:
            errorY_a[write_index] = sommeErreurY

        if write_index<12000:
            preX_a[write_index] = prevX

        if write_index<12000:
            preY_a[write_index] = prevY
```

```
    if startBalanceBall == True:
        sommeErreurX += (consigneX-ballPosX)
        sommeErreurY += (consigneY-ballPosY)

prevX,prevY = 0,0
prevConsigneX, prevConsigneY = 0,0
start_time = 0
Time_INIT=time.time()

def main():
    start_timeFPS = time.time()
    global H,S,V
    global getPixelColor
    global x,y, alpha, beta, beta_m
    global prevX, prevY, prevAlpha, prevBeta, prevConsigneX, prevConsigneY
    global consigneX, consigneY, sommeErreurX, sommeErreurY
    global camWidth,camHeight
    global timeInterval, start_time, time_interval, Time_INIT
    global showVideoWindow
    global img
    global indice_prueba
    global x_a, write_index, y_a, consigneX_a, consigneY_a, Ix, Iy, alpha_a, beta_m_a, time_a

    _, img=cam.read()
    img = img[0:int(camHeight),int((camWidth-camHeight)/2):int(camWidth-((camWidth-camHeight)/2))] #[Y1:Y2,X1:X2]
    imgCircle = np.zeros(img.shape, dtype=np.uint8)
    cv2.circle(imgCircle, (240,240), 190, (255, 255, 255), -1, 8, 0)
```

```
        img = img & imgCircle
        imgHSV = cv2.cvtColor(img,cv2.COLOR_BGR2HSV)

        if getPixelColor == True and mouseY > 0 and mouseY < 480 and mouseX <
480:
            pixelColorOnClick = img[mouseY,mouseX]
            pixelColorOnClick = np.uint8([[pixelColorOnClick]])
            pixelColorOnClick = cv2.cvtColor(pixelColorOnClick,cv2.COLOR_BGR2HSV)
            H = pixelColorOnClick[0,0,0]
            S = pixelColorOnClick[0,0,1]
            V = pixelColorOnClick[0,0,2]
            getPixelColor = False

        lowerBound=np.array([10,100,20])
        upperBound=np.array([25,255,255])

        mask=cv2.inRange(imgHSV,lowerBound,upperBound)
        mask = cv2.blur(mask,(7,7))                       # ajoute du flou à l'image
        mask = cv2.erode(mask, None, iterations=2)        # retire les parasites
        mask = cv2.dilate(mask, None, iterations=2)       # retire les parasites



        cnts,hierarchy=cv2.findContours(mask.copy(),
cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
          cv2.circle(img, (int(consigneX), int(consigneY)), int(4),(255, 0, 0), 2)
         if showCalqueCalibrationBool == True:
            cv2.circle(img, (240,240), 220,(255, 0, 0), 2)
            cv2.circle(img, (240,240), 160,(255, 0, 0), 2)
            cv2.line(img, (240, 240), (240, 240+160), (255,0,0), 2)
        cv2.line(img, (240, 240), (240+138, 240-80), (255,0,0), 2)
        cv2.line(img, (240, 240), (240-138, 240-80), (255,0,0), 2)

    if len(cnts) > 0:
        cnts = cnts[0]
        center = None
        c = cnts
        timeInterval = time.time() - start_time
        (x, y), radius = cv2.minEnclosingCircle(c)
        if radius > 10:
            cv2.putText(img,str(int(x)) + ";" + str(int(y)).format(0, 0),(int(x)-50, int(y)-50), cv2.FONT_HERSHEY_SIMPLEX,1, (255, 255, 255), 2)
            cv2.circle(img, (int(x), int(y)), int(radius),(0, 255, 255), 2)
            PIDcontrol(int(x),int(y),prevX,prevY,consigneX,consigneY)
            start_time = time.time()
        else:
            sommeErreurX, sommeErreurY = 0,0
            x, y = prevX, prevY

    if showVideoWindow == True:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img = Image.fromarray(img)
        imgtk = ImageTk.PhotoImage(image=img)
        lmain.imgtk = imgtk
        lmain.configure(image=imgtk)

    lmain.after(10, main)
  #drawWithBall()
    if prevConsigneX != consigneX or prevConsigneY != consigneY:
           sommeErreurX, sommeErreurY = 0,0


        paintGraph()

        prevX,prevY = int(x), int(y)

        prevConsigneX, prevConsigneY = consigneX, consigneY

        prevAlpha = alpha

        prevBeta = beta_m

        time_interval = time.time()- Time_INIT


        if (time.time() - start_timeFPS) > 0:

           print("FPS: ", 1.0 / (time.time() - start_timeFPS))


        if write_index<12000:

             time_a[write_index] = time_interval

main()

tk.mainloop()
```

1.2. ARDUINO

El programa que se muestra a continuación es el que comanda los motores.

```
#include <Servo.h>


    Servo servoA;

    Servo servoB;

    Servo servoC;


    int ledTemoin = 8;

    float angleA = 45;

    float angleB = 45;

    float angleC = 45;

    int ledRojo = 8;

    int ledR = LOW;

    unsigned long previousMillis = 0;

    void setup() {

      Serial.begin(19200);

      //pinMode(ledTemoin, OUTPUT); // led temoin

      pinMode(ledRojo, OUTPUT);

      //digitalWrite(ledTemoin, LOW);

      servoC.attach(9);     //servo A

      servoB.attach(11);     //servo B  // se han cambiado el motor A por el C para ajustar el control

      servoA.attach(10);    //servo C

      delay(1000);
```

```
  servoA.writeMicroseconds((-2380 + 1450)*float(angleA) / 90 + 2380);
  //servoB.writeMicroseconds((-2370 + 1480)*float(angleB) / 90 + 2370);
  servoB.writeMicroseconds((-2330 + 1480)*float(angleB) / 90 + 2330);
  servoC.writeMicroseconds((-2270 + 1430)*float(angleC) / 90 + 2270);

}

int count = 0;

void loop() {
  //digitalWrite(ledTemoin , millis() / 500 % 2 ); // led temoin clignotement

  if (Serial.available()) {
    unsigned long currentMillis = millis();
    if (currentMillis - previousMillis >= 500) {
      previousMillis = currentMillis;
      if (ledR == LOW) {
        ledR = HIGH;
      } else {
        ledR = LOW;
      }
      digitalWrite(ledRojo, ledR);
    }



 String a = Serial.readStringUntil('\n');
    if (a == "descendreBras") {
      angleA = 45;
```

```
       angleB = 45;

       angleC = 45;


      } else {

       a.remove(0, 1);

       a.remove(a.length() - 1, 1);

       angleA = getValue(a, ',', 0).toFloat();

       angleB = getValue(a, ',', 1).toFloat();

       angleC = getValue(a, ',', 2).toFloat();

      }

      servoA.writeMicroseconds((-2380 + 1450)*float(angleA) / 90 + 2380);

      servoB.writeMicroseconds((-2370 + 1480)*float(angleB) / 90 + 2370);

      servoC.writeMicroseconds((-2300 + 1430)*float(angleC) / 90 + 2300);

    }

}

String getValue(String data, char separator, int index) {

     int found = 0;

     int strIndex[] = { 0, -1 };

     int maxIndex = data.length() - 1;

for (int i = 0; i <= maxIndex && found <= index; i++) {

      if (data.charAt(i) == separator || i == maxIndex) {

       found++;

       strIndex[0] = strIndex[1] + 1;

       strIndex[1] = (i == maxIndex) ? i + 1 : i;

      }

     }

     return found > index ? data.substring(strIndex[0], strIndex[1]) : "";

    }
```

A continuación se encuentra el programa que se realizo para comprobar el funcionamiento de los motores.

```
#include <Servo.h>

#include <math.h>

int ledRojo = 8;

int ledR = LOW;

int i=0;

unsigned long previousMillis = 0;

// Declaramos la variable para controlar el servo

Servo servoA;

Servo servoB;

Servo servoC;

int posicion, posicionA, posicionB, posicionC;

void setup() {

  // Iniciamos el monitor serie para mostrar el resultado

  Serial.begin(9600);

  pinMode(ledRojo, OUTPUT);

  // Iniciamos el servo para que empiece a trabajar con el pin 9

  servoA.attach(9);

  servoB.attach(10);

  servoC.attach(11);

}


void loop() {

  posicionA = 130;

  posicionB = 120;

  posicionC = 115;
```

```
 servoA.write(posicionA+5*sin(2*3.14*i*1));
 servoB.write(posicionA+5*sin(2*3.14*i*1+120/10));
 servoC.write(posicionA+5*sin(2*3.14*i*1-120/10));
i++;
Serial.print(i);
 unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= 1000) {
    previousMillis = currentMillis;
    if (ledR == LOW) {
      ledR = HIGH;
    } else {
      ledR = LOW;
    }
    digitalWrite(ledRojo, ledR);
  }
}
```

1.3. OCTAVE

El código empieza con la tabla de datos (data) donde aparece la relación entre los ángulos de giro de la plataforma y los ángulos de giro de los motores, la cual no puede mostrarse debido a su gran tamaño ya que ocupa 316.976 filas en el editor del programa.

En esta primera parte de código se definen la relación entre los angulos de giro de la plataforma, theta_x y theta_y con el desplazamiento y con los angulos de los motores theta1, theta2 y theta 3.

```
data_converted=[data(:,1).*cos(data(:,2)*pi/180),data(:,1).*sin(data(:,2)*pi/180),d
ata(:,3:5)];

    a=15;

    r1=70;

    r2=85;

    l1=51.55;

    l3=103.1;

    w1=89.287;

    w2=89.287;

    theta=90*pi/180*[-1:0.1:1];

    theta_x=30*pi/180*[-1:0.1:1];

    theta_y=30*pi/180*[-1:0.1:1];

    for x=1:length(theta)

d_m1(x)=sqrt(-a^2+2*a*r1*cos(theta(x))+r1^2*(sin(theta(x)))^2-
r1^2+r2^2)+r1*sin(theta(x))-(sqrt(-
a^2+2*a*r1*cos(52.97*pi/180)+r1^2*(sin(52.97*pi/180))^2-
r1^2+r2^2)+(r1*sin(52.97*pi/180)));

    endfor

    plot(theta*180/pi,d)

    for i=1:length(theta_x)

      for j=1:length(theta_y)
```

```
          d1(i,j)=-((l1+l3)*tan(theta_x(i)))/3+((w1+w2)*tan(theta_y(j)))/2;

          theta1(i,j)=interp1(d_m1,theta,d1(i,j));

theta1_ref(i,j)=griddata(data_converted(:,1),data_converted(:,2),data_converted(:,3),theta_x(i)*180/pi,theta_y(j)*180/pi);

          d2(i,j)=2*((l1+l3)*tan(theta_x(i)))/3;

          theta2(i,j)=interp1(d_m1,theta,d2(i,j));

theta2_ref(i,j)=griddata(data_converted(:,1),data_converted(:,2),data_converted(:,4),theta_x(i)*180/pi,theta_y(j)*180/pi);

          d3(i,j)=-((l1+l3)*tan(theta_x(i)))/3-((w1+w2)*tan(theta_y(j)))/2;

          theta3(i,j)=interp1(d_m1,theta,d3(i,j));

theta3_ref(i,j)=griddata(data_converted(:,1),data_converted(:,2),data_converted(:,5),theta_x(i)*180/pi,theta_y(j)*180/pi);

          endfor

endfor
```

En la siguiente parte que se muestra, corresponde a la que genera el plano con los puntos obtenidos en RecurDyn.

```
X_data=[19.46*pi/180,19.45*pi/180,17*pi/180,0,0,0,-19.46*pi/180,-19.45*pi/180,-17*pi/180];

Y_data=[19.28*pi/180,0.21*pi/180,-18.39*pi/180,19.35*pi/180,0.047*pi/180,-19.64*pi/180,19.28*pi/180,0.21*pi/180,-18.39*pi/180];

Z_data1=[16.743,32.960,48.042,43.140,52.970,65.530,65.179,73.670,90.702];

Z_data2=[83.907,54.850,29.932,77.060,52.970,28.770,83.907,54.850,29.932];

Z_data3=[65.179,73.670,90.702,43.140,52.970,65.530,16.743,32.960,48.042];

x1=[-0.3:0.1:0.3];

y1=[-0.3:0.1:0.3];

for i=1:length(x1)

  for j=1:length(y1)

z1(i,j)=griddata(X_data,Y_data,Z_data1,x1(i),y1(j));

z2(i,j)=griddata(X_data,Y_data,Z_data2,x1(i),y1(j));

z3(i,j)=griddata(X_data,Y_data,Z_data3,x1(i),y1(j));
```

```
endfor

endfor
```

Y para terminar con la parte que compara los modelos cinemáticos, aparece el código mediante el cual se han dibujado los planos expuestos en la memoria.

```
figure(1);hold on;mesh(theta_x,theta_y,theta3'*180/pi,"facecolor","c"),

hold on;

mesh(-theta_x,-theta_y,theta1_ref,"facecolor","b")

hold off;

figure(2);mesh(theta_x,theta_y,theta2'*180/pi,"facecolor","c"),

hold on;

mesh(-theta_x,-theta_y,theta2_ref,"facecolor","b"),hold off;

figure(3);mesh(theta_x,theta_y,theta1'*180/pi,"facecolor","c"),

hold on;

mesh(-theta_x,-theta_y,theta3_ref,"facecolor","b"),hold off;

figure(1);hold on;mesh(x1,y1,z1,"facecolor","g");hold off;xlabel('Tx');ylabel('Ty');zlabel('T1');

figure(2);hold on;mesh(x1,y1,z2,"facecolor","g");hold off;xlabel('Tx');ylabel('Ty');zlabel('T2');

figure(3);hold on;mesh(x1,y1,z3,"facecolor","g");hold off;xlabel('Tx');ylabel('Ty');zlabel('T3');
```

A continuación se muestra el código utilizado para representar la grafica mediante la cual hemos obtenido el valor de la masa en función del ángulo alpha.

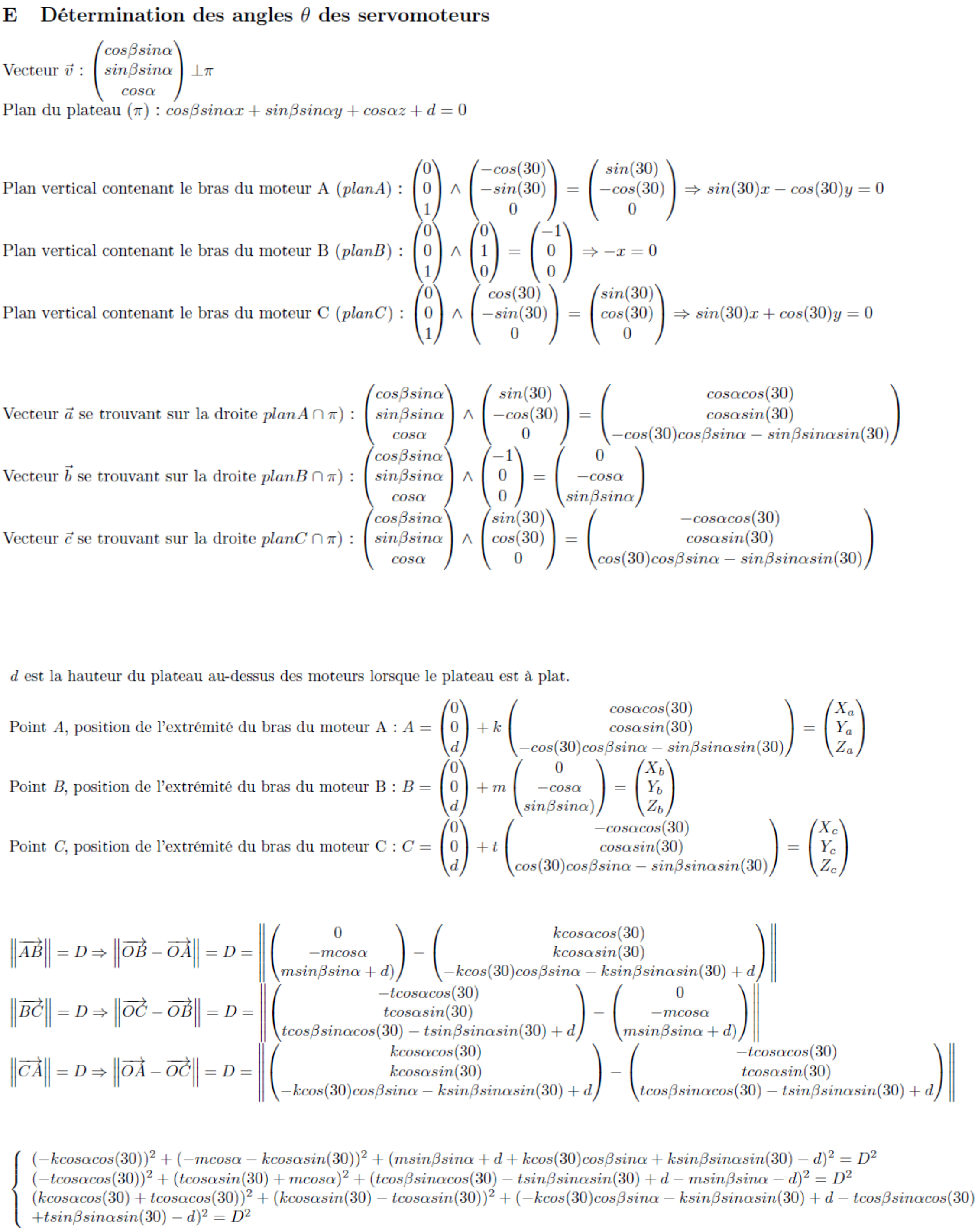
```
alpha_m=[0:1:90]*pi/180;

for x=1:length(alpha_m)

    beta_m(x)=acos(((-0.015+0.07*cos(alpha_m(x))/0.085)));

end

for i=1:length(alpha_m)
```

```
m(i)=0.32/(9.8*cos(alpha_m(i))*0.07+(9.8/tan(beta_m(i)))*sin(alpha_m(i))*0.07-9.8*cos(beta_m(i))*0.085+(9.8/tan(beta_m(i)))*sin(beta_m(i))*0.085);
end
plot(alpha_m*180/pi,m);
```

Para finalizar con el código generado en Octave, se muestran las pruebas que se realizaron aplicando el método de las raíces para comprobar que tipo de control favorecía al sistema.

```
s = tf('s');
Kd=0;
Ki=0;
Kp=0;
%g = tf([9.8],[1,0,0]);
g=(9.8*Kp/s^2)+(9.8*Kd/s)+(9.8*Ki/s^3);
%g=(Kp*9.8/s^2)+(9.8*Kd/s);
%g=(Kp*9.8/s^2)+(9.8*Ki/s^3);
%g=Kp*9.8/s^2;
%g=9.8*Kd/s;
%g=9.8*Ki/s^3;
rlocus(g);
```

ANEXO 2. (MODELO CINEMÁTICO ORIGINAL)



E Détermination des angles θ des servomoteurs

Vecteur $\vec{v}$: $\begin{pmatrix} cos\beta sin\alpha \\ sin\beta sin\alpha \\ cos\alpha \end{pmatrix} \perp\pi$

Plan du plateau (π) : $cos\beta sin\alpha x + sin\beta sin\alpha y + cos\alpha z + d = 0$

Plan vertical contenant le bras du moteur A $(planA)$: $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \wedge \begin{pmatrix} -cos(30) \\ -sin(30) \\ 0 \end{pmatrix} = \begin{pmatrix} sin(30) \\ -cos(30) \\ 0 \end{pmatrix} \Rightarrow sin(30)x - cos(30)y = 0$

Plan vertical contenant le bras du moteur B $(planB)$: $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \wedge \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} \Rightarrow -x = 0$

Plan vertical contenant le bras du moteur C $(planC)$: $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \wedge \begin{pmatrix} cos(30) \\ -sin(30) \\ 0 \end{pmatrix} = \begin{pmatrix} sin(30) \\ cos(30) \\ 0 \end{pmatrix} \Rightarrow sin(30)x + cos(30)y = 0$

Vecteur $\vec{a}$ se trouvant sur la droite $planA \cap \pi)$: $\begin{pmatrix} cos\beta sin\alpha \\ sin\beta sin\alpha \\ cos\alpha \end{pmatrix} \wedge \begin{pmatrix} sin(30) \\ -cos(30) \\ 0 \end{pmatrix} = \begin{pmatrix} cos\alpha cos(30) \\ cos\alpha sin(30) \\ -cos(30)cos\beta sin\alpha - sin\beta sin\alpha sin(30) \end{pmatrix}$

Vecteur $\vec{b}$ se trouvant sur la droite $planB \cap \pi)$: $\begin{pmatrix} cos\beta sin\alpha \\ sin\beta sin\alpha \\ cos\alpha \end{pmatrix} \wedge \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ -cos\alpha \\ sin\beta sin\alpha \end{pmatrix}$

Vecteur $\vec{c}$ se trouvant sur la droite $planC \cap \pi)$: $\begin{pmatrix} cos\beta sin\alpha \\ sin\beta sin\alpha \\ cos\alpha \end{pmatrix} \wedge \begin{pmatrix} sin(30) \\ cos(30) \\ 0 \end{pmatrix} = \begin{pmatrix} -cos\alpha cos(30) \\ cos\alpha sin(30) \\ cos(30)cos\beta sin\alpha - sin\beta sin\alpha sin(30) \end{pmatrix}$

d est la hauteur du plateau au-dessus des moteurs lorsque le plateau est à plat.

Point A, position de l'extrémité du bras du moteur A : $A = \begin{pmatrix} 0 \\ 0 \\ d \end{pmatrix} + k \begin{pmatrix} cos\alpha cos(30) \\ cos\alpha sin(30) \\ -cos(30)cos\beta sin\alpha - sin\beta sin\alpha sin(30) \end{pmatrix} = \begin{pmatrix} X_a \\ Y_a \\ Z_a \end{pmatrix}$

Point B, position de l'extrémité du bras du moteur B : $B = \begin{pmatrix} 0 \\ 0 \\ d \end{pmatrix} + m \begin{pmatrix} 0 \\ -cos\alpha \\ sin\beta sin\alpha) \end{pmatrix} = \begin{pmatrix} X_b \\ Y_b \\ Z_b \end{pmatrix}$

Point C, position de l'extrémité du bras du moteur C : $C = \begin{pmatrix} 0 \\ 0 \\ d \end{pmatrix} + t \begin{pmatrix} -cos\alpha cos(30) \\ cos\alpha sin(30) \\ cos(30)cos\beta sin\alpha - sin\beta sin\alpha sin(30) \end{pmatrix} = \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix}$

$$\left\|\overrightarrow{AB}\right\| = D \Rightarrow \left\|\overrightarrow{OB} - \overrightarrow{OA}\right\| = D = \left\| \begin{pmatrix} 0 \\ -mcos\alpha \\ msin\beta sin\alpha + d) \end{pmatrix} - \begin{pmatrix} kcos\alpha cos(30) \\ kcos\alpha sin(30) \\ -kcos(30)cos\beta sin\alpha - ksin\beta sin\alpha sin(30) + d \end{pmatrix} \right\|$$

$$\left\|\overrightarrow{BC}\right\| = D \Rightarrow \left\|\overrightarrow{OC} - \overrightarrow{OB}\right\| = D = \left\| \begin{pmatrix} -tcos\alpha cos(30) \\ tcos\alpha sin(30) \\ tcos\beta sin\alpha cos(30) - tsin\beta sin\alpha sin(30) + d \end{pmatrix} - \begin{pmatrix} 0 \\ -mcos\alpha \\ msin\beta sin\alpha + d) \end{pmatrix} \right\|$$

$$\left\|\overrightarrow{CA}\right\| = D \Rightarrow \left\|\overrightarrow{OA} - \overrightarrow{OC}\right\| = D = \left\| \begin{pmatrix} kcos\alpha cos(30) \\ kcos\alpha sin(30) \\ -kcos(30)cos\beta sin\alpha - ksin\beta sin\alpha sin(30) + d \end{pmatrix} - \begin{pmatrix} -tcos\alpha cos(30) \\ tcos\alpha sin(30) \\ tcos\beta sin\alpha cos(30) - tsin\beta sin\alpha sin(30) + d \end{pmatrix} \right\|$$

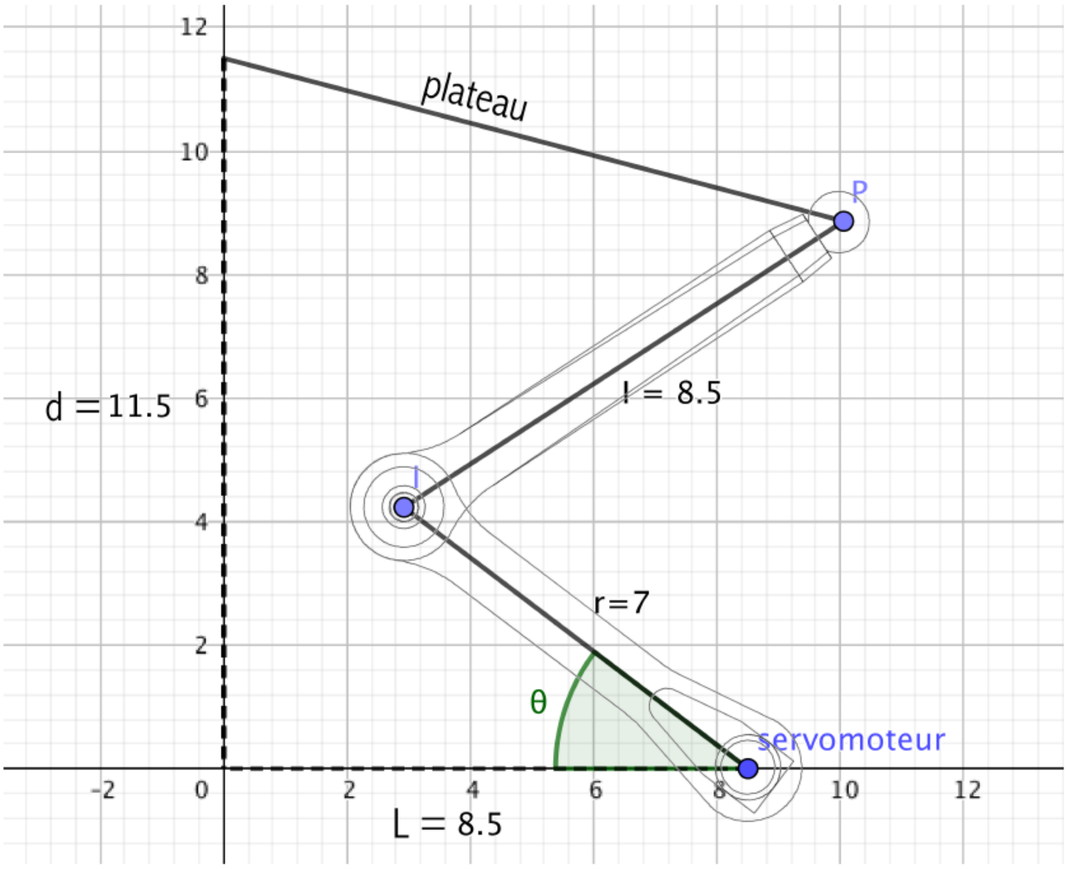
$$\begin{cases} (-kcos\alpha cos(30))^2 + (-mcos\alpha - kcos\alpha sin(30))^2 + (msin\beta sin\alpha + d + kcos(30)cos\beta sin\alpha + ksin\beta sin\alpha sin(30) - d)^2 = D^2 \\ (-tcos\alpha cos(30))^2 + (tcos\alpha sin(30) + mcos\alpha)^2 + (tcos\beta sin\alpha cos(30) - tsin\beta sin\alpha sin(30) + d - msin\beta sin\alpha - d)^2 = D^2 \\ (kcos\alpha cos(30) + tcos\alpha cos(30))^2 + (kcos\alpha sin(30) - tcos\alpha sin(30))^2 + (-kcos(30)cos\beta sin\alpha - ksin\beta sin\alpha sin(30) + d - tcos\beta sin\alpha cos(30) \\ +tsin\beta sin\alpha sin(30) - d)^2 = D^2 \end{cases}$$

Point I, emplacement de l'articulation du bras d'un moteur : $I(L - rcos\theta; rsin\theta)$
Point P, emplacement de l'extrémité du bras d'un moteur : $P(\sqrt{X_p^2 + Y_p^2}; Z_p)$

$$\left\|\overrightarrow{PI}\right\| = l$$

$$\overrightarrow{PI} = \begin{pmatrix} L - rcos\theta \\ rsin\theta \end{pmatrix} - \begin{pmatrix} \sqrt{X_p^2 + Y_p^2} \\ Z_p \end{pmatrix}$$

$$(L - rcos\theta - \sqrt{X_p^2 + Y_p^2})^2 + (rsin\theta - Z_p)^2 = l^2$$



ANEXO 3. (PRESUPUESTO)

Piezas impresora 3D	Cantidad	Material	Precio
Parte inferior de la camara	1	PLA	7,96 €
Parte superior de la camara	1	PLA	5,71 €
Parte inferior del soporte camara	1	PLA	7,20 €
Parte superior del soporte camara	1	PLA	3,19 €
Soporte PCB	1	PLA	2,71 €
Contorno base 1	1	PLA	13,62 €
Contorno base 2	1	PLA	13,27 €
Contorno base 3	1	PLA	13,03 €
Contorno base 4	1	PLA	14,62 €
Contorno base 5	1	PLA	8,62 €
Parte interior base 1	1	PLA	12,34 €
Parte interior base 2	1	PLA	13,79 €
Parte interior base 3	1	PLA	13,40 €
Sujeccion conector joystck	1	PLA	1,23 €
Parte inferior articulacion	3	PLA	7,38 €
Parte superior articulacion	3	PLA	7,38 €
			145,45 €

Posibles piezas impresora 3D (depende medidas impresion)	Cantidad	Material	Precio
Parte superior plataforma	1	Poliamida	142,80 €
Parte inferior plataforma	1	Poliamida	98,28 €
Tapa de la base	1	Poliamida	112,56 €
			353,64 €

Componentes comprados	Cantidad	Precio
Servomotor Futaba S3003	3	48,63 €
Camara	1	36,64 €
Kit de varillas y bolas	1	13,35 €
		98,62 €

El coste total del proyecto asciende a 597,71€.

Relación de documentos

(_) Memoria ....NN   páginas

(X) Anexos ....NN   páginas

La Almunia, a 27 de 11 de 2019

Firmado: Raúl López Martín