```
#|---------------------------------------------------------------------------|#
#| Program and run 'PID for vis track.cydsn' on PSoC board to use this code. |#
#|                                                                           |#
#| This code tracks red objects with a webcam.                               |#
#| Based off of camera data, stepper motor speed/direction is calculated and |#
#| transmitted to the PSoC board over UART                                   |#
#|---------------------------------------------------------------------------|#


###########################################################
## setup
###########################################################

import serial           #import libraries
import numpy as np      #...
import time
import cv2

ser=serial.Serial()     #create serial object
ser.baudrate=9600       #set baud rate
ser.port='COM4'         #port PSoC board is plugged into
ser.open()              #open port

count=0                 #counter
d_error=0               #derivative error
last_error=0            #used to calculate previous cam frame error
sum_error=0             #for integral error
cap=cv2.VideoCapture(1) #camera object


###########################################################
## infinite while loop, press ESC button to stop loop
###########################################################

while(1):


    ###########################################################
    ## read camera, subtract all color but red, show video
    ###########################################################

    _,frame=cap.read()                                      #frame captured from camera

    red=np.matrix(frame[:,:,2])                             #red matrix
    green=np.matrix(frame[:,:,1])                           #green matrix
    blue=np.matrix(frame[:,:,0])                            #blue matrix

    red_only=np.int16(red)-np.int16(green)-np.int16(blue)   #subtract green & blue

    red_only[red_only<0]=0                                  #make all negative numbers 0
    red_only[red_only>255]=255                              #make anything >255 = 255

    red_only[red_only<50]=0                                 #set threshold
    red_only[red_only>=50]=255

    red_only=np.uint8(red_only)                             #put red only as uint8

    cv2.imshow('rgb',frame)                                 #show before subtraction
    cv2.imshow('red_only',red_only)                         #show subtracted frame

    ###########################################################
    ## center of brightness: 5 step calculation for X column
    ###########################################################

    red_only[red_only>0]=1                              #all values 0 or 1

    column_sums=np.matrix(np.sum(red_only,0))           #step 1:sum columns
```

```python
    column_numbers=np.matrix(np.arange(640))            #step 2: 1st [1,2,....,639,640]
    column_mult=np.multiply(column_sums,column_numbers) #step 2: 2nd multiply matrices
    total=np.sum(column_mult)                           #step 3: sum multiplied matrix
    total_total=np.sum(np.sum(red_only))                #step 4: sum of original matrix
    if total_total>0:                                   #eliminates division by 0
            column_location=total/total_total           #step 5: divide
    else:
        column_location=320.0                           #set as the center
    print ('Column location= ',column_location)         #print result in shell

    #################################################################
    ## check for nan camera output and set column location tolerance
    #################################################################

    nanCheck=np.isnan(column_location)                      #check for nan column_location
    if nanCheck == True:
        column_location=320                                 #column location is center
    if column_location<330.0 and column_location>310.0:     #tolerance
        count=count+1                                       #in tolerance
    else:
        count=0                                             #out of tolerance
    if count>5:                                             #in tolerance for 5 frames
        column_location=320.0                               #set location as center

    #################################################################
    ## calculate speed and direction of stepper motor
    #################################################################

    error=column_location-320                   #distancefrom the center
    kp=4.0                                      #proportional gain
    kd=0.0                                      #derivative gain
    ki=0.0                                      #integral gain
    speed=kp*error+ki*sum_error+kd*d_error      #PID, only P

    if (speed>1000):                            #1000 steps/s is max speed
        speed=1000
    if (speed<=-1000):
        speed=-1000

    direction=1                                 #clockwise
    if (speed<0):                               #if speed is negative
        speed=-speed                            #set speed as positive
        direction=2                             #set direction to ccw

    #################################################################
    ## send calculated speed and direction to PSoC board over UART
    #################################################################

    u=[0,0,0]                   #motor data
    u[2]=int(speed/256)         #speed byte
    u[1]=int(speed-(256*u[2]))  #speed byte
    u[0]=int(direction)         #direction byte

    j=0                         #counter
    while (j<3):                #send 3 bytes of information
        i=bytearray([u[j]])     #make integer a byte
        ser.write(i)            #transmit byte
        j=j+1                   #increment counter
        time.sleep(0.006)       #delay

    #################################################################
    ## for ID gain calculations for next frame - not used in this
    ## code - stepper ran best using only proportional gain
    #################################################################

    d_error=(error-last_error)/0.118
    last_error=error
    sum_error=sum_error+(error*0.118)

    #################################################################
    ## stop the code
    #################################################################

    k=cv2.waitKey(50)               #delay
    if k==27:                       #escape key is pressed
        u=[1,0,0]                   #set speed 0
        j=0                         #reset counter
        while (j<3):                #send 3 bytes
            i=bytearray([u[j]])     #make integer a byte
            ser.write(i)            #transmit byte
            j=j+1                   #increment counter
            time.sleep(0.006)       #delay
        j=0                         #reset counter
        while (j<3):                #send it again
            i=bytearray([u[j]])
            ser.write(i)
            j=j+1
            time.sleep(0.006)
        ser.close()                 #close serial port
        break                       #break the infinite while loop

cv2.destroyAllWindows()             #close camera windows
```