**import** RPi**.**GPIO **as** GPIO

GPIO**.**setmode**(**GPIO**.**BOARD**)**

GPIO**.**setup**(**37**,** GPIO**.**OUT**)**

GPIO**.**setup**(**35**,** GPIO**.**OUT**)** # GPIO INITIALIZATION

GPIO**.**setup**(**33**,** GPIO**.**OUT**)**

GPIO**.**setup**(**29**,** GPIO**.**OUT**)**

GPIO**.**setup**(**31**,** GPIO**.**OUT**)**

GPIO**.**setup**(**38**,** GPIO**.**OUT**)**

GPIO**.**setup**(**40**,** GPIO**.**OUT**)**

GPIO**.**setup**(**7**,** GPIO**.**IN**)**

**import** math

center\_read**=**325

center\_read\_green**=**320

# import the necessary packages

**from** picamera**.**array **import** PiRGBArray # CAMERA INITIALIZATION

**from** picamera **import** PiCamera

**import** time

**import** cv2

**import** numpy **as** np

# initialize the camera and grab a reference to the raw camera capture

camera **=** PiCamera**()**

camera**.**resolution **=** **(**640**,** 480**)**

camera**.**framerate **=** 50

camera**.**hflip **=** **True**

rawCapture **=** PiRGBArray**(**camera**,** size**=(**640**,** 480**))**

# allow the camera to warmup

time**.**sleep**(**0.1**)**

**def** find\_red\_center**():**

# capture frames from the camera

**for** frame **in** camera**.**capture\_continuous**(**rawCapture**,** format**=**"bgr"**,** use\_video\_port**=True):**

# grab the raw NumPy array representing the image, then initialize the timestamp

# and occupied/unoccupied text

image **=** frame**.**array

blur **=** cv2**.**blur**(**image**,** **(**3**,**3**))**

#hsv to complicate things, or stick with BGR

hsv **=** cv2**.**cvtColor**(**blur**,**cv2**.**COLOR\_BGR2HSV**)**

#thresh = cv2.inRange(hsv,np.array((0, 200, 200)), np.array((20, 255, 255)))

lower **=** np**.**array**([**167**,**184**,**86**],**dtype**=**"uint8"**)**

#upper = np.array([225,88,50], dtype="uint8")

upper **=** np**.**array**([**179**,**255**,**255**],** dtype**=**"uint8"**)**

thresh **=** cv2**.**inRange**(**hsv**,** lower**,** upper**)**

thresh2 **=** thresh**.**copy**()**

# find contours in the threshold image

contours**,**hierarchy **=** cv2**.**findContours**(**thresh**,**cv2**.**RETR\_LIST**,**cv2**.**CHAIN\_APPROX\_SIMPLE**)**

# finding contour with maximum area and store it as best\_cnt

max\_area **=** int**(**100**)**

best\_cnt **=** 1

cx\_most\_left **=** int**(**0**)**

new\_area **=** 1

cx\_search**,**cy\_search **=** **[**0**,**0**]**

**for** cnt **in** contours**:**

area **=** int**(**cv2**.**contourArea**(**cnt**))**

M\_search **=** cv2**.**moments**(**cnt**)**

**if** **not(**M\_search**[**'m00'**]==** 0**):**

cx\_search**,**cy\_search **=** int**(**M\_search**[**'m10'**]/**M\_search**[**'m00'**]),** int**(**M\_search**[**'m01'**]/**M\_search**[**'m00'**])**

**if** **((**area **>** max\_area**)** **&** **(**cx\_search **>** cx\_most\_left**)):**

best\_cnt **=** cnt

new\_area **=** area

cx\_most\_left **=** cx\_search

# finding centroids of best\_cnt and draw a circle there

M **=** cv2**.**moments**(**best\_cnt**)**

#new\_area = cv2.contourArea(best\_cnt)

r **=** math**.**sqrt**(**new\_area**/**3.1415**)**

r**=**int**(**math**.**ceil**(**r**))**

cx**,**cy **=** int**(**M**[**'m10'**]/**M**[**'m00'**]),** int**(**M**[**'m01'**]/**M**[**'m00'**])**

#if best\_cnt>1:

cv2**.**circle**(**blur**,(**cx**,**cy**),**10**,(**0**,**0**,**255**),-**1**)**

cv2**.**circle**(**blur**,(**cx**,**cy**),**r**,(**0**,**255**,**0**),**5**)**

# show the frame

cv2**.**imshow**(**"Frame"**,** blur**)**

#cv2.imshow('thresh',thresh2)

key **=** cv2**.**waitKey**(**1**)** **&** 0xFF

# clear the stream in preparation for the next frame

rawCapture**.**truncate**(**0**)**

#print the center coordinates

**print(**cx**,**cy**)**

# if the `q` key was pressed, break from the loop

**if** key **==** ord**(**"q"**):**

**break**

**return** cx

**def** find\_green\_center**():**

# capture frames from the camera

**for** frame **in** camera**.**capture\_continuous**(**rawCapture**,** format**=**"bgr"**,** use\_video\_port**=True):**

# grab the raw NumPy array representing the image, then initialize the timestamp

# and occupied/unoccupied text

image **=** frame**.**array

blur **=** cv2**.**blur**(**image**,** **(**3**,**3**))**

#hsv to complicate things, or stick with BGR

hsv **=** cv2**.**cvtColor**(**blur**,**cv2**.**COLOR\_BGR2HSV**)**

#thresh = cv2.inRange(hsv,np.array((0, 200, 200)), np.array((20, 255, 255)))

lower **=** np**.**array**([**35**,**91**,**10**],**dtype**=**"uint8"**)**

#upper = np.array([225,88,50], dtype="uint8")

upper **=** np**.**array**([**80**,**255**,**255**],** dtype**=**"uint8"**)**

thresh **=** cv2**.**inRange**(**hsv**,** lower**,** upper**)**

thresh2 **=** thresh**.**copy**()**

# find contours in the threshold image

contours**,**hierarchy **=** cv2**.**findContours**(**thresh**,**cv2**.**RETR\_LIST**,**cv2**.**CHAIN\_APPROX\_SIMPLE**)**

# finding contour with maximum area and store it as best\_cnt

max\_area **=** 100

best\_cnt **=** 1

**for** cnt **in** contours**:**

area **=** cv2**.**contourArea**(**cnt**)**

**if** area **>** max\_area**:**

best\_cnt **=** cnt

max\_area **=** area

# finding centroids of best\_cnt and draw a circle there

M **=** cv2**.**moments**(**best\_cnt**)**

r **=** math**.**sqrt**(**max\_area**/**3.1415**)**

r**=**int**(**math**.**ceil**(**r**))**

cx**,**cy **=** int**(**M**[**'m10'**]/**M**[**'m00'**]),** int**(**M**[**'m01'**]/**M**[**'m00'**])**

#if best\_cnt>1:

cv2**.**circle**(**blur**,(**cx**,**cy**),**10**,(**0**,**0**,**255**),-**1**)**

cv2**.**circle**(**blur**,(**cx**,**cy**),**r**,(**0**,**255**,**0**),**5**)**

# show the frame

cv2**.**imshow**(**"Frame"**,** blur**)**

#cv2.imshow('thresh',thresh2)

key **=** cv2**.**waitKey**(**1**)** **&** 0xFF

# clear the stream in preparation for the next frame

rawCapture**.**truncate**(**0**)**

#print the center coordinates

**print(**cx**,**cy**)**

# if the `q` key was pressed, break from the loop

**if** key **==** ord**(**"q"**):**

**break**

**return** cx

**def** asagi\_in**():**

GPIO**.**output**(**29**,** 1**)** # topu bırak

GPIO**.**output**(**31**,** 1**)** # topu al

GPIO**.**output**(**38**,** 0**)** # aşağı

GPIO**.**output**(**40**,** 1**)** # yukarı

GPIO**.**output**(**29**,** 1**)** # topu bırak

GPIO**.**output**(**31**,** 1**)** # topu al

GPIO**.**output**(**38**,** 1**)** # aşağı

GPIO**.**output**(**40**,** 1**)** # yukarı

**def** yukari\_cik**():**

GPIO**.**output**(**29**,** 1**)** # topu bırak

GPIO**.**output**(**31**,** 1**)** # topu al

GPIO**.**output**(**38**,** 1**)** # aşağı

GPIO**.**output**(**40**,** 0**)** # yukarı

GPIO**.**output**(**29**,** 1**)** # topu bırak

GPIO**.**output**(**31**,** 1**)** # topu al

GPIO**.**output**(**38**,** 1**)** # aşağı

GPIO**.**output**(**40**,** 1**)** # yukarı

**def** topu\_birak**():**

GPIO**.**output**(**29**,** 0**)** # topu bırak

GPIO**.**output**(**31**,** 1**)** # topu al

GPIO**.**output**(**38**,** 1**)** # aşağı

GPIO**.**output**(**40**,** 1**)** # yukarı

GPIO**.**output**(**29**,** 1**)** # topu bırak

GPIO**.**output**(**31**,** 1**)** # topu al

GPIO**.**output**(**38**,** 1**)** # aşağı

GPIO**.**output**(**40**,** 1**)** # yukarı

**def** topu\_al**():**

GPIO**.**output**(**29**,** 1**)** # topu bırak

GPIO**.**output**(**31**,** 0**)** # topu al

GPIO**.**output**(**38**,** 1**)** # aşağı

GPIO**.**output**(**40**,** 1**)** # yukarı

GPIO**.**output**(**29**,** 1**)** # topu bırak

GPIO**.**output**(**31**,** 1**)** # topu al

GPIO**.**output**(**38**,** 1**)** # aşağı

GPIO**.**output**(**40**,** 1**)** # yukarı

**def** stop**():**

GPIO**.**output**(**37**,** 0**)**

GPIO**.**output**(**35**,** 0**)**

GPIO**.**output**(**33**,** 0**)**

**def** forward**():**

GPIO**.**output**(**37**,** 1**)**

GPIO**.**output**(**35**,** 1**)**

GPIO**.**output**(**33**,** 1**)**

**def** right**():**

GPIO**.**output**(**37**,** 0**)**

GPIO**.**output**(**35**,** 0**)**

GPIO**.**output**(**33**,** 1**)**

**def** left**():**

GPIO**.**output**(**37**,** 1**)**

GPIO**.**output**(**35**,** 0**)**

GPIO**.**output**(**33**,** 0**)**

**def** backward**():**

GPIO**.**output**(**37**,** 0**)**

GPIO**.**output**(**35**,** 1**)**

GPIO**.**output**(**33**,** 1**)**

**def** cwsearch**():**

GPIO**.**output**(**37**,** 1**)**

GPIO**.**output**(**35**,** 1**)**

GPIO**.**output**(**33**,** 0**)**

top\_bende **=** 0

flag**=**0 #initial search flagi

flag\_basket**=**0 #initial search flagi basket

**while(**1**):**

**if** top\_bende **==** 0**:**

center\_x\_ball **=** find\_red\_center**()**

left\_margin**=**center\_read**-**1

right\_margin**=**center\_read**+**1

**if** flag**==**0**:**

cwsearch**()**

time**.**sleep**(**2.2**)**

stop**()**

**while(not(**center\_read**-**100**<**center\_x\_ball **<**center\_read**+**100**)):** #initial search

cwsearch**()**

time**.**sleep**(**0.7**)**

stop**()**

center\_x\_ball **=** find\_red\_center**()**

**if** center\_read**-**100**<**center\_x\_ball **<**center\_read**+**100**:**

flag**=**1

**break**

**if** left\_margin**<** center\_x\_ball **<** right\_margin**:**

forward**()**

**elif** center\_x\_ball **==** 0 **:**

cwsearch**()**

**print(**"turning around"**)**

**elif** center\_x\_ball **<** left\_margin**:**

**print(**"going right"**)**

right**()**

time**.**sleep**(**0.05**)**

**elif** right\_margin **<** center\_x\_ball**:**

**print(**"going left"**)**

left**()**

time**.**sleep**(**0.05**)**

**else** **:**

stop**()**

**if** **(**GPIO**.**input**(**7**)==**0**):** #top gördüm mü

stop**()**

time**.**sleep**(**0.2**)**

**if(**GPIO**.**input**(**7**)==**0**):** #kesin gördüm

top\_bende **=** 1

asagi\_in**()**

backward**()**

time**.**sleep**(**1**)**

stop**()**

time**.**sleep**(**16**)**

forward**()**

time**.**sleep**(**2**)**

stop**()**

yukari\_cik**()** #topu aldımmmmm

time**.**sleep**(**5**)**

backward**()**

time**.**sleep**(**3**)**

stop**()**

**print** "top bende"

cwsearch**()**

time**.**sleep**(**2.2**)**

stop**()**

**if** top\_bende **==** 1**:**

**print** "simdi basketi ariyorum"

center\_basket **=** find\_green\_center**()**

left\_margin\_green**=**center\_read\_green**-**1

right\_margin\_green**=**center\_read\_green**+**1

**if** flag\_basket**==**0**:**

**while(not(**center\_read\_green**-**100**<**center\_basket**<** center\_read\_green**+**100**)):** #initial search

cwsearch**()**

time**.**sleep**(**0.7**)**

stop**()**

center\_basket **=** find\_green\_center**()**

**if** center\_read\_green**-**150**<**center\_basket**<** center\_read\_green**+**150**:**

flag\_basket**=**1

**break**

**if** left\_margin\_green**<** center\_basket **<** right\_margin\_green**:**

forward**()**

**elif** center\_basket **==** 0 **:**

cwsearch**()**

**print(**"turning around basketi ariyorum"**)**

**elif** center\_basket **<** left\_margin\_green**:**

**print(**"going right"**)**

right**()**

time**.**sleep**(**0.05**)**

**elif** right\_margin\_green **<** center\_basket**:**

**print(**"going left"**)**

left**()**

time**.**sleep**(**0.05**)**

**else** **:**

stop**()**

**if** **(**GPIO**.**input**(**7**)==**0**):** #gordum mu

stop**()**

time**.**sleep**(**0.2**)**

**if(**GPIO**.**input**(**7**)==**0**):** #kesin gordum

**print(**"basketi gördüm"**)**

stop**()**

forward**()**

time**.**sleep**(**0.4**)**

stop**()**

topu\_birak**()**

time**.**sleep**(**1**)**

backward**()**

time**.**sleep**(**2.5**)**

stop**()**

time**.**sleep**(**1**)**

topu\_al**()**

time**.**sleep**(**2**)**

**print** "topu biraktim"

flag\_basket**=**0 #initialize all flags

top\_bende**=**0

flag**=**0