```python
# importar la llibreria serial que ens  permet realitzar comunicacions serie via els ports usb
import serial
# importar el document HandTrackingModule en el qual hi ha tot el codi necessari per a fer servir la càmera
# com a càmera intel·ligent amb el nom htm ja que és més curt
import HandTrackingModule as htm
# importar la llibreria cv2 que es fa servir per a realitzar captures de la càmera i modificar-ne les imatges
from HandTrackingModule import cv2
# importar la llibreria time la qual ens permet fer coses com parar momentaniament el procés o saber quant temps fa que s'ha engegat
from HandTrackingModule import time

#inicialitzem la comunicació sèrie entre l'ordinador i el robot a través del port COM 5 i amb una velocitat de 9600 bouds
robot = serial.Serial('COM3', 9600)
#Indiquem al robot que ha començat el programa de l'ordinador
robot.write(b's')
#Definim un tipus nou de classe que conté només una matriu de 3 per 3. Aquesta ens permetrà guardar el valor
#de les peces jugades per cadascun dels jugadors fins a aquell moment
class jugador():
    #Establim el mètode d'inici el qual genera la matriu i la assigna a la variable graella de la classe.
    def __init__(self):
        graella = [
        [False, False, False],
        [False, False, False],
        [False, False, False]
        ]
        self.graella = graella

#Establim les variables glovals de diferents paràmetres que poden trobar-se dins de diferents funcions
#o que han de guardar el seu valor entre crida i cirda de la mateixa funció

#Primer trobem els valors que defineixen la posició en la qual es dibuixa la graella sobre la imatge de la càmera
#limit_dalt representa el desplaçament entre el lateral superior de la graella i el lateral superior de la imatge
limit_dalt = 200
#limit_esq representa el desplaçament entre el lateral esquerra i el mateix lateral de la imatge
limit_esq = 150
#yBarGap és la distància entre les diferents barres horitzontals que hi ha al gràfica
yBarGap = 50
#xBarGap és la distància entre les diferents barres verticals que hi ha a la gràfica
xBarGap = 50
#Les barres estan separades segons direcció per si els caselles no fossin quadrades completament i
#fossin rectangulars a causa de la posició de la càmera

#Aqui hi ha els valors que han de guardar el valor entre diferents funcions.
#tPosar és un valor on es guarda l'ultim temps en el que s'ha posat el dit sobre una cuadrícula vàlida diferent de la que era fins el moooment
tPosar = 0
#Novapos indica que la posició del dit ha canviat i per tant el valor de tPosar també
Novapos = True
#posx i posy indica en les dues cordenades respectives a quina casella de la matriu es troba el dit. Es podia passar com a valor de la funció posdiit
#però d'aquesta manera és més senzilla d'utilitzar i a més ocupa menys variables temporals futils.
```

```
posx = 5
posy = 5

#Aquesta funció es fa servir per a ajustar la mida de la graella, és a dir les variables que hi ha a sobre, respecte la imatge.
def ajust():
    """
    La funció ajust és utilitzada per ajustar la posició de la graella que es dibuixa respecte la graella real en cas que s'hagi mogut la càmera
    """
    #es criden les variables que hem definit anteriorment per a fer-les servir en aquesta funció
    global limit_dalt
    global limit_esq
    global yBarGap
    global xBarGap
    #Inicialitzem l'objecte de la càmera
    cap = cv2.VideoCapture(0)
    #Fem sortir per la terminal les instruccions de com funciona el bloc
    print("""
Introdueix un darrere l'altre els píxels en els quals vols que es trobin els seguents valors:
  1 - distància entre la part superior de la pantalla i la part superior de la graella
  2 - distància entre la part esquerra de la pantalla i la part esquerra de la graella
  3 - distància entre fila i fila de la graella
  4 - distància entre columna i columna de la graella.

  Per exemple si volem una desplaçament de 200 pixels en la superior, 150 en l'esquerra i una mida de les caselles de 50 per 50 introduïm

  200
  150
  50
  50

""")
    #Realitzem un bucle perquè es pugui modificar vàries vegades els valors de les variàbles aabans de que s'acabi la funció
    while True:
        #capturem una imatge de la càmera i la guardem en la variable img
        success, img = cap.read()
        #Dibuixem la quadricula en la imatge capturada anteriorment
        dibuixaCuadricula(img, limit_esq, limit_dalt, xBarGap, yBarGap)
        #fem la imatge més gran perquè es vegi millor
        img = cv2.resize(img, (int(img.shape[1]*2), int(img.shape[0] *2)), interpolation = cv2.INTER_AREA)
        #Posem la imatge en pantalla
        cv2.imshow("Image",img)
        #ens esperem un milisegon com a mínim
        cv2.waitKey(1)
        #Fem un try except, que és una funció que et permet intentar executar un codi i en cas que hi hagi algun error s'executa la part de except
        #Això ens permet que no es trenqui tot el programa sinó que encapsules tots els possibles errors. Com que aquest tros és molt susceptible a
        #generar errors hem utilitzat aquest recurs.
        try:
            #introduim els valors per consola i els guardem en forma d'enters
            limit_dalt = int(input())
            limit_esq = int(input())
            yBarGap = int(input())
            xBarGap = int(input())
            #En cas que no hagi saltat cap error informem de que els valors han sigut canviats correctament
```

```
            print("Valors canviats correctament")
            #Repetim el procés d'imprimir la cuadricula per comprovar-ne els resultats
            success, img = cap.read()
            dibuixaCuadricula(img, limit_esq, limit_dalt, xBarGap, yBarGap)
            img = cv2.resize(img, (int(img.shape[1]*2), int(img.shape[0] *2)), interpolation = cv2.INTER_AREA)
            cv2.imshow("Image",img)
            cv2.waitKey(1)
            #En cas que la cuadricula estigui bé es surt en cas que no es trona a començar el bucle
            if input("Ok? 1- si 2 - no \n") == "1":
                break
        #En cas que hi hagi algun error
        except:
            #Informem de que hi ha algun error i que s'han de tornar a introduir les dades
            print("Hi ha hagut algun error, assegura't d'introduir només nombres sense espais ni lletres. Des del principi \n")
#En aquesta funció es comprova una graella per saber si les peces que hi ha posades estan posades de tal menera que el jugador al qual
#li pertany ha guanyat
#El parametre d'entrada "jugador" és la graella del jugador en questió, és a dir que ha de ser una matriu de 3 per 3
def haguanyat(jugador):
    """
    En aquesta funció es busca en una graella de 3 per 3 si les posicions de les variables són guanyadores d'un 3 en ratlla
    """
    #Declarem quatre variables que suposen les 4 possibles condicions de victòria.
    #La primera condició és que hi hagi una fila plena, i per tant que en una fila hi hagi 3 fitxes
    consecutivesx = 0
    #La segona condició és que hi hagi una columna plena, és a dir que en una columna hi hagi 3 fitxes
    consecutivesy = 0
    #Les altres dues són les diagonals, és a dir que en una d'elles hi hagi 3 peces.
    diag1 = 0
    diag2 = 0
    #Creem un cicle for per a correr un dels eixos de la matriu
    for x in range(len(jugador)):
        #a cada eix hem de reiniciar el procés de contatge de peces que hi ha en aquella fila o columna.
        consecutivesy = 0
        consecutivesx = 0
        #Dins el primer for en creem un altre per recorrer cadascuna de les posicions dins de cadascun dels eixos de la matriu.
        for y in range(len(jugador[x])):
            #En aquest cas l'eix és horitzontal i es va corrent per cadascuna de les columnes
            if jugador[x][y]:
                #Si a casella hi ha una fitxa es suma 1 al contador de les files horitzontals
                consecutivesx += 1
            #En aquest cas l'eix és vertical i es va comporovant les files
            if jugador[y][x]:
                #En cas que la casella sigui plena es suma 1 al contador de les files verticals
                consecutivesy += 1
            #Només en el cas que la fila i columna de la casella coincideixin (i per tant es trobi dins la primera diagonal) es comprova
            #si hi ha peça o no
            if x == y and jugador[x][y]:
                #En cas que hi hagi una peça es suma 1 al contador de la diagonal 1
                diag1 += 1
            #En cas que la suma de la fila i la columna de la casella resulti 2, vol dir
```

```
que la casella forma part de la segona diagonal (ja que
152             # per fer dos només pot ser 2 + 0, 1 + 1 o 0 + 2, les tres coordenades de les
caselles de la segona diagonal)
153             if x + y == 2 and jugador[x][y]:
154                 #En tal cas si la casella és plena es suma 1 a la variable de la segona
diagonal
155                 diag2 += 1
156             #En cas que una de les variables arribi a 3, vol dir que s'ha complert una de
les condicions de victoria
157             if consecutivesx == 3 or consecutivesy == 3 or diag1 == 3 or diag2 == 3:
158                 #A l'haver-se complert una de les condicions la funcio retorna un valor
True
159                 return True
160     #En cas que no s'hagi complert cap de les condicions i per tant no s'hagi retornat True
es retorna False un cop s'ha recorregut tota la matriu
161     return False
162 #La funció posdit busca les coordenades del dit en una graella a partr de les coordenades
del mateix
163 def posdit(xma, yma):
164     """
165     En la funció posdit es busca de quina de les diferents caselles se n'ha d'enviar les
coordenades al robot
166     La gracia d'aquesta és que no ho fa de manera instantània ja que sinó no podries posar
mai res a la casella del mig (1,1) ja que per
167     arribar-hi s'ha de passar per una altra. Així que per a decidir a quina posició es posa
la peça s'espera una estona i comprova que el dit estigui
168     a la mateixa posició durant una estona
169     """
170     #S'importen les dades glovals ja que les fem servir en altres funcions com és el main
171     global posx
172     global posy
173     global Novapos
174     global tPosar
175     #Calculem el temps real en el moment en el que es crida la funció
176     cTime = time.time()
177     #En el cas que la ma estigui dins el perímetre de la graella
178     if xma > limit_esq and yma > limit_dalt and xma - limit_esq < xBarGap * 3 and yma -
limit_dalt < yBarGap * 3:
179         #recorrem per les diferents files de la graella
180         for i in range(0, 3):
181             #En cas que la ma estigui per sota de la barra a comprovar en aquell moment
s'enten que la mà està en la posició de la barra
182             #ja que per darrere de la graella no hi pot ser ja que ho hem comprovat
anteriorment
183             if xma - limit_esq < xBarGap * (i + 1):
184                 #En cas que la posició sigui diferent a la última que s'ha detectat
185                 if i != posx:
186                     #Es s'iguala el temps en el que s'ha posat el dit en una posició nova a
la posició actual
187                     tPosar = cTime
188                     #S'identifica que s'ha fet un canvi de posició
189                     Novapos = True
190                 #Es posa com a ultima posició trobada la actual
191                 posx = i
192                 #es surt del bucle
193                 break
194         #Es repeteix el procés anterior amb l'eix de les y
195         for i in range(0, 3):
196             if yma - limit_dalt < yBarGap * (i + 1):
197                 if i != posy:
198                     tPosar = cTime
199                     Novapos = True
200                 posy = i
201                 break
```

```
202         #En cas que s'hagi posat una posició en cada eix i per tant no siguin les inicials i hagin passat 3 segons des que s'ha canviat el dit de
203         #casella per últim cop es considera que la posició és bona
204         if posx != 5 and posy != 5 and (cTime - tPosar > 3) and Novapos:
205             #de tal manera que es torna un True per fer saber que s'ha trobat una posició bona
206             #i es torna la variable Novapos a Fals per si es cridés la funció de manera seguida que no detectés directament dues posicions de cop
207             Novapos = False
208             return True
209     #En cas que no es compleixi alguna de les condicions necessàries es torna un False ja que no hi ha cap posició detectada
210     return False
211 #Aquesta funció és només per a endreçar, és a dir que no és necessària estrictament però ajuda a entendre el codi
212 #El que fa és dibuixar una cuadricula de 3 per 3 en una imatge donada amb les cordenades de l'extrem superior esquerre i la distància
213 #entre les diferents barres
214 def dibuixaCuadricula(img, limesq, limdalt, xbargap, ybargap):
215     """
216     Dibuixa en la imatge img, una cuadricula de 3 x 3 amb el vertex superior esquerre en el punt (limesq, limdalt) com a x i y i amb una separació de
217     columnes de xbargap i de files de ybargap.
218 
219     """
220     #Linea inferior del recuadre interior
221     cv2.line(img,(limesq, limdalt + ybargap), (limesq + (3*xbargap), limdalt + ybargap), (255,255,255), 5)
222     #Linea superior del recuadre interior
223     cv2.line(img, (limesq, limdalt + (2 *ybargap)), (limesq + (3*xbargap), limdalt + (2 *ybargap)), (255,255,255), 5)
224     #Linea esquerra del recuadre interior
225     cv2.line(img, (limesq + xbargap, limdalt), (limesq + xbargap, limdalt + ybargap * 3), (255,255,255), 5)
226     #Linea dreta del recuadre interior
227     cv2.line(img, (limesq + xbargap*2, limdalt), (limesq + xbargap*2, limdalt + ybargap * 3), (255,255,255), 5)
228     #Recuadre exterior. Aquest es dibuixa al final perquè les línies que es superposin quedin dibuixades com a les línies de fora
229     cv2.rectangle(img, (limesq, limdalt), (limesq + xbargap * 3, limdalt + ybargap * 3), (255,255,0), 5 )
230 #Aquest és el funcionament principal de al màquina
231 def main():
232     #Primer iniciem l'objecte que ens permet processar la imatge i trobar les poscions de cadascuna de les parts de la mà o mans que hi apareguin
233     detector = htm.handDetector()
234     #Seguidament iniciem l'objecte per al qual ens referirem a la càmera
235     cap = cv2.VideoCapture(0)
236     #Importem les variables globals que s'han d'utilitzar
237     global limit_dalt
238     global limit_esq
239     global xBarGap
240     global yBarGap
241     #Iniciem una variable per guardar el color del cercle que es pintarà a la pantalla
242     colorCercle = (0,0,255)
243     #Una alatra variable per saber si s'ha canviat la mà que es veia a la pantalla
244     novama = True
245     #una variable per l'estat de la màquina d'estats
246     estat = 0
247     #Un contador pel nombre de peces que ha posat el robot
248     peces = 0
249     #Una variable per guardar els bytes que es llegeixin del robot
250     info = b's'
251     #una variable per a la graella de cada jugador
```

```
    j1 = jugador()
    j2 = jugador()
    #S'inicia un bucle en el que hi haurà la màquina d'estats
    while True:
        print(estat)
        #capturem una imatge de la càmera i la guardem en la variable img
        success, img = cap.read()
        #Es processa la imatge per trobar les mans que hi surtin
        img = detector.findHands(img, False, False)
        #Es guarden les posicions de les diferents parts de les mans en una array
        lmList = detector.findPosition(img, 0, True)
        #A l'inici de cada cicle es busca si hi ha hagut alguna comunicació sèrie i si hi ha sigut es guarda en la variable info
        if robot.inWaiting():
            info = robot.read()
            #En el cas que l'ultim valor enviat sigui una a, significaria que s'ha premut el polsador d'atur i es para tot el programa.
            if info == b'a':
                #Es borra la informació que hi havia en la variable
                info = b's'
                #Es passa a l'estat 5 per informar de que hi ha hagut algun error
                estat = 5

        #En cas que s'estigui a l'estat 0
        if estat == 0:
            #Es posa el text iindicat a sota en la part superior de la imatge
            cv2.putText(img, "Torn de jugar el jugador 1 (amb les creus)", (10,20), cv2.FONT_HERSHEY_PLAIN, 1, (25,0,255), 2)
            #Sempre i quant la llista no sigui buida, és a dir que hi hagi alguna mà a la pantalla
            if len(lmList) != 0:
                #es guarda les coordenades del dit index en les variables xma i yma( líndex és la posició 8 de la array que guarda les posicions
                # de la ma, i dins de la lmList[8], la [1] És la x i la [2] és la y. El [0] seria el valor de la màrca però és 8 ja que coincideix
                # el valor de la posició en la array)
                xma = lmList[8][1]
                yma = lmList[8][2]
                #En cas que la mà sigui una de diferent que per defecte és cert ja que al principi no hi ha cap ma
                if novama:
                    #Es busca la posició de la graella en la que es troba el dit
                    if posdit(xma, yma):
                        #Un cop es sap, es mira aviam si aquella mateixa posició ja estava ocupada
                        if j1.graella[posy][posx] == False and j2.graella[posy][posx] == False:
                            #En cas que no estigués ocupada s'ocupa
                            j1.graella[posy][posx] = True
                            #Es canvia de color el cercle
                            colorCercle = (0,255,0)
                            #Es passa al seguent estat
                            estat = 1
                        else:
                            #En cas que la posició estés plena s'informa
                            print("posició plena, torna-ho a provar")
                        #Es considera que la ma ja ha estat feta servir i s'ha de treure i tornar a posar-ne una altra
                        novama = False
                #Es pinta el cercle del color que faci falta
                cv2.circle(img, (10, 70), 10, colorCercle, -1)
            #En cas que la llista sigui vuida i per tant no hi hagi cap ma a la pantalla
            else:
```

```
305                 #Es torna a possar el color inicial
306                 colorCercle = (0,0,255)
307                 #Es considera que s'ha canviat la ma
308                 novama = True
309         elif estat == 1:
310             #Estat per a esperar una resposta del robot, ja que no podem saber si està a
    punt per rebre informació o no
311             time.sleep(0.5)
312             if info == b'm':
313                 #Un cop s'ha rebut la senyal de marrxa
314                 info = ''
315                 #S'envia la informació de la posició a la que ha d'anar el robot
316                 robot.write(b'f')
317                 #Els nombres enters s'han de codificar com una string amb format UTF8
318                 robot.write(str(posy).encode('UTF-8'))
319                 robot.write(b'c')
320                 robot.write(str(posx).encode('UTF-8'))
321                 robot.write(str(peces).encode('UTF-8'))
322                 #Es suma 1 al contador de peçes posades
323                 peces +=1
324                 #Es comprova si ha guanyat el jugador 1 després d'haver posat la peça
325                 if haguanyat(j1.graella):
326                     #En cas que  hagi guanyat es trenca el bucle
327                     estat = 4
328                 else:
329                     #Si no ha guanyat es passa al següent estat
330                     estat = 2
331         elif estat == 2:
332             cv2.putText(img, "Torn de jugar el jugador 2 (amb els cercles)", (10,20),
    cv2.FONT_HERSHEY_PLAIN, 1, (255,255,0), 2)
333             if len(lmList) != 0:
334                 xma = lmList[8][1]
335                 yma = lmList[8][2]
336                 if novama:
337                     if posdit(xma, yma):
338                         if j1.graella[posy][posx] == False and j2.graella[posy][posx] ==
    False:
339                             j2.graella[posy][posx] = True
340                             colorCercle = (0,255,0)
341 
342                             estat = 3
343                         else:
344                             print("posició plena, torna-ho a provar")
345                         novama = False
346                 cv2.circle(img, (10, 70), 10, colorCercle, -1)
347             else:
348                 colorCercle = (0,0,255)
349                 novama = True
350         elif estat == 3:
351             #Estat per a esperar una resposta del robot, ja que no podem saber sinó si ha
    deixat o no la peça
352             time.sleep(0.5)
353             #Un cop el rboot està en posició de tornar-se a moure
354             if info == b'm':
355                 #Un cop s'ha rebut la senyal de marrxa
356                 info = ''
357                 #S'envia la informació de la posició a la que ha d'anar el robot
358                 robot.write(b'f')
359                 #Els nombres enters s'han de codificar com una string amb format UTF8
360                 robot.write(str(posy).encode('UTF-8'))
361                 robot.write(b'c')
362                 robot.write(str(posx).encode('UTF-8'))
```

```
363                 robot.write(str(peces).encode('UTF-8'))
364                 #Es suma 1 al contador de peçes posades
365                 peces +=1
366                 #Es comprova si ha guanyat el jugador 2 després d'haver posat la seva peça
367                 if haguanyat(j2.graella):
368                     #En cas que  hagi guanyat es trenca el bucle
369                     estat = 4
370                 else:
371                     #Si s'han posat totes les peces possibles però encara no hi ha cap
guanyador es considera empat
372                     if peces >= 7:
373                         #Si hi ha empat es passa a un estat diferent
374                         estat = 6
375                     else:
376                         #En cas que no hi hagi el nombre màxim de peces posat i tampoc hagi
guanyat ningú es segueix el joc amb normalitat
377                         estat = 0
378         #En el cas que un dels jugadors guanyi es passa a aquest estat, en el que es
notifica i es surt del bucle
379         elif estat == 4:
380             print("Felicitats, has guanyat")
381             break
382         #Si hi ha un empat, també es notifica i es surt del bucle. Els dos estats estan
separats per si es volgués implementar funcions
383         #diferents per a cadascun
384         elif estat == 6:
385             print("Empat!")
386             break
387         #En cas que hi hagi algun error o que es premi el polsador d'atur des del robot es
passa en aquest estat en el que s'ignora tot el que
388         #hi hagi en el buffer d'entrada i es surt del bucle
389         elif estat == 5:
390             robot.read_all()
391             robot.flushInput()
392             estat = 0
393             print("hi ha hagut algun error")
394             break
395 
396         #Un cop s'acaba cadascuna de les voltes del bucle es dibuixa la graella a la
pantalla per quan s'ensenyi
397         dibuixaCuadricula(img, limit_esq, limit_dalt, xBarGap, yBarGap)
398 
399         #es fa la imatge més gran ja que la resolució per defecte és molt petita
400         img = cv2.resize(img, (int(img.shape[1]*2), int(img.shape[0] *2)), interpolation =
cv2.INTER_AREA)
401 
402         #Finalment s'ensenya la imatge per pantalla
403         cv2.imshow("Image",img)
404         #Es para el programa com a mínim un milisegon
405         cv2.waitKey(1)
406 
407 #En el cas que s'executi aquest programa com a principal i no com a llibreria, que és la
intenció principal, però és el protocol correcte en aquests casos
408 if __name__ == "__main__":
409     #Es mostra el menú inicial
410     print(
411         """Introdueix un nombre segons el que vulguis fer:
412         1 - Funcionament normal
413         2 - ajustar valors graella
414         3 - sortir
415         """)
416     #S'agafa la entrada com a nombre enter
417     estat = int(input())
```

```
418     #Si és 2, indica que es vol moure la graella que hi ha per defecte i s'executa la
    funció corresponent
419     if estat == 2:
420         ajust()
421         #Es torna a demanar que es vol fer, qualsevol resposta diferent de 1 farà acabar el
    programa
422         estat = int(input("vols començar a jugar? \n 1 - si \n 2 - sortir\n"))
423     #En cas que es vulgui jugar s'entra a aquest estat
424     while estat == 1:
425         #S'executa una vegada el joc
426         main()
427         #Si es vol tornar a jugar com que l'estat torna a ser 1, no trenca el bucle i es
    torna a llençar el joc.
428         estat = int(input("vols tornar a jugar? \n 1 - si \n 2 - sortir\n"))
```