|  |  |  |
| --- | --- | --- |
| Task 1 | | |
| task\_1a\_part1.py | | Global:shapes={}  Shapes(dictionary) getContours(imgColor)  Mask(np.array) colorMask(imgHsv)  cX,cY(number) calibrateCentroid(cX,cY)  Obj(String) getShape4(cnt)  Bool isEqual(v1,v2)  Bool isPerpendicular(v1,v2)  Bool isParallel(v1,v2)  shapes = scan\_image(img\_file\_path) |
| Task\_1a\_part2.py | | Global:frame\_details={}  Frame\_details process\_video(vid\_file\_path, frame\_list) |
| Task\_1b.py | | Rect orderedPolyDp(corners)  imgMorph getBorderCoordinates(imgMorph)  warped\_img = applyPerspectiveTransform(input\_img)  maze\_array = detectMaze(warped\_img)  warped\_img = writeToCsv('test\_cases/maze00.csv', maze\_array) |
| Task 2 | | |
| Task\_2a.py | | Global :  client\_id = -1  calibrate=[]  flag=False  vision\_sensor\_handle=-1\  Shapes getShape()  client\_id = init\_remote\_api\_server()  return\_code = start\_simulation()  vision\_sensor\_image, image\_resolution, return\_code = get\_vision\_sensor\_image()  transformed\_image = transform\_vision\_sensor\_image(vision\_sensor\_image, image\_resolution)  return\_code = stop\_simulation()  exit\_remote\_api\_server() |
| Task\_2b.py | | Global :  client\_id = -1  client\_id = init\_remote\_api\_server()  vision\_sensor\_image, image\_resolution, return\_code = get\_vision\_sensor\_image()  transformed\_image = transform\_vision\_sensor\_image(vision\_sensor\_image, image\_resolution)  send\_data(maze\_array)  exit\_remote\_api\_server() |
| Task\_3 | | |
| Task\_3.py | | Global:  client\_id = -1  setpoint = [1024,256]  vision\_sensor\_handle = 0  revolute\_handle=[-1,-1,-1,-1,-1,-1,-1,-1]  outMax=60  outMin=-60  kp=np.array([0.02,0.02],dtype='float64')  ki=np.array([0.001,0.001],dtype='float64')#ki=ki\*SampleTime  kd=np.array([0.135,0.135],dtype='float64')#kd=kd/SampleTime  lastTime=0  error = np.array([0,0],dtype='float64')  summation = np.array([0,0],dtype='float64')  ITerm=np.array([0,0],dtype='float64')  lastInput=np.array([0,0],dtype='float64')  Input=np.array([0,0],dtype='float64')  SampleTime = 0.01 #0.01 sec  Output=np.array([0,0],dtype="float64")  lastOutput=np.array([0,0],dtype="float64")  setAngles(Output)  SetTunings(Kp,Ki,Kd)  SetSampleTime(NewSampleTime)  SetOutputLimits(  Min,Max)  Initialize()  init\_setup()  np.array(transformed,dtype='float64') coordinateTransform(xy)  control\_logic(center\_x,center\_y) |
| Task\_4 | | |
| Task\_4a.py | | distance\_mat make\_step(k, distance\_mat, maze\_array)  path\_copy[min\_indx] getMinPath(path)  final\_path getPath(start\_coord,end\_coord,distance\_mat,maze\_array)  path = find\_path(maze\_array, start\_coord, end\_coord)  start, end = read\_start\_end\_coordinates("start\_end\_coordinates.json", "maze00") |
| Task4b.py | | Global:  client\_id = -1  setpoint = [0, 0]  start\_coord = (0,4)  end\_coord = (9,5)  center\_x,center\_y getBallCoordinates()  maze\_array, path = calculate\_path\_from\_maze\_image(img\_file\_path) send\_data\_to\_draw\_path(rec\_client\_id,path)  pixel\_path = convert\_path\_to\_pixels(path)  traverse\_path(pixel\_path) |
| Task\_4b\_script.lua | Global:  maze\_array  setWallLocation(pos,ori,name,parent)  deleteWall(name)  sim.getObjectPosition(handle,-1) getObjectCoordinates(name,base)  dimensions getObjectSize(name)  z getWallZValue()  deleteExit(number)  wallObjectHandle = createWall()  receiveData(inInts,inFloats,inStrings,inBuffer)  generateHorizontalWalls()  generateVerticalWalls()  deleteWalls()  createMaze()  groupWalls()  addToCollection()  drawPath(inInts,path,inStrings,inBuffer)  sysCall\_init()  sysCall\_beforeSimulation()  sysCall\_afterSimulation()  sysCall\_cleanup() | |
|  | | |
|  | | |
|  | | |