به نام خدا

دانشکده مهندسی برق و کامپیوتر
دانشگاه تهران

مینی پروژه اول درس مکاترونیک

دکتر طالع ماسوله

دانشجو : فاطمه نائینیان

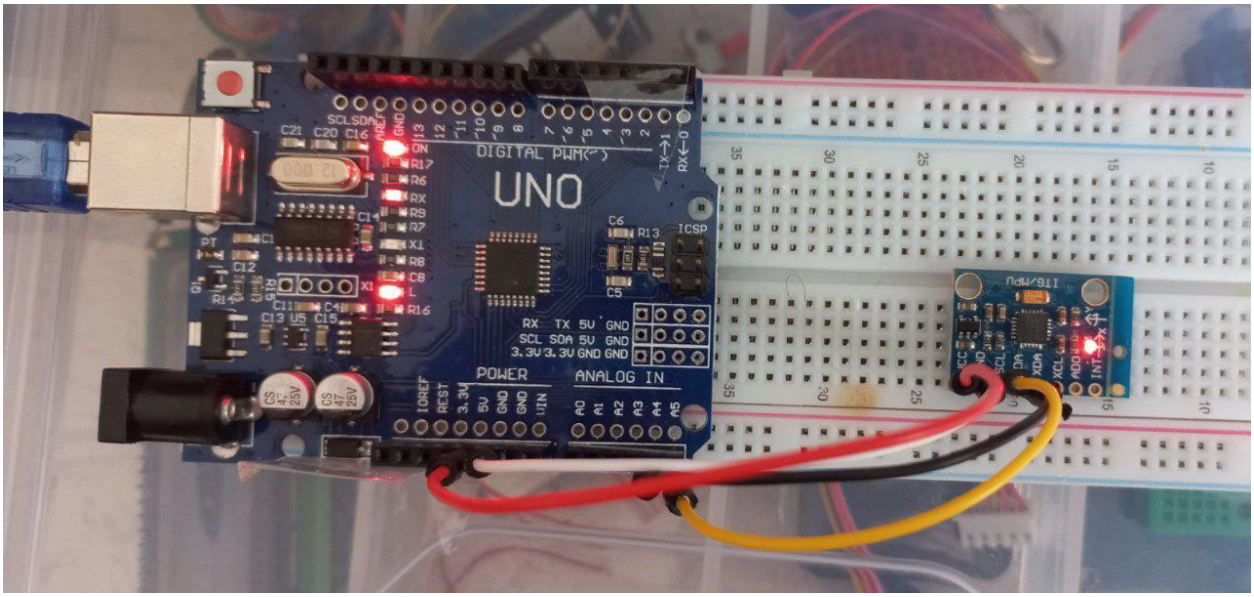
شماره دانشجویی : 810198479

بهار 1401

فهرست

بخش اول

ابتدا یک مدار به شکل زیر میبندیم و سپس به لپ تاپ وصل میکنیم و از طریق IDE آردوینو کد ها را بر روی ان اپلود کرده و طریق serial monitor میتوانیم سیگنال هایی که ژیروسکوپ میفرستد را مشاهده کنیم.



نمایش دادهها روی نمودار

در این بخش داده ها را با کمک نرم افزار serial plot مشاهده میکنیم.

- با کمک serial plot می توانیم میزان roll,pitch,yaw را مقایسه کنیم. حال فرض کنید هیچ اطلاعاتی از جهت ها نداریم. میتوانیم برد بورد را فیکس کنیم و در یکی از جهت ها دوران کنیم ، حال با توجه به سیگنالی که در serial plot میبینیم میتوانیم تشخیص دهیم چه محوری بوده است. Roll محور x و pitch محور y و yaw محور z را نشان میدهد. حال اگر x را ثابت نگه داریم، میتوانیم انقدر دوران را ادامه دهیم و ازمون و خطا کنیم تا در نهایت دو سیگنال صفر شده و یک سیگنال باقی بماند حال با توجه به اینکه سیگنال غیر صفر کدام سیگنال است ، میتوان محور را شناسایی کرد.

به دست آوردن ماتریس دوران

میدانیم roll زاویه گردش حول x و pitch زاویه گردش حول y و yaw زاویه گردش حول z است. بنابر این با هرکدام از این زاویه ها میتوانیم ماتریس ان دوران را پیدا کنیم. ماتریس دوران حول هر محور به شکل زیر است.

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

حال میتوانیم از این قانون استفاده کنیم که در دستگاه مختصات متحرک به ترتیب از ابتدا به انتها در هم ضرب می شوند.

$$Qxyz = R_x * R_y * R_z$$

در نهایت ماتریس دوران کلی به دست خواهد امد.

$$\begin{aligned}
&R_X R_Y R_Z \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos A & -\sin A \\ 0 & \sin A & \cos A \end{pmatrix} \begin{pmatrix} \cos B & 0 & \sin B \\ 0 & 1 & 0 \\ -\sin B & 0 & \cos B \end{pmatrix} \begin{pmatrix} \cos C & -\sin C & 0 \\ \sin C & \cos C & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} \cos B & 0 & \sin B \\ \sin A \sin B & \cos A & -\sin A \cos B \\ -\cos A \sin B & \sin A & \cos A \cos B \end{pmatrix} \begin{pmatrix} \cos C & -\sin C & 0 \\ \sin C & \cos C & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} \cos B \cos C & -\cos B \sin C & \sin B \\ \sin A \sin B \cos C + \cos A \sin C & -\sin A \sin B \sin C + \cos A \cos C & -\sin A \cos B \\ -\cos A \sin B \cos C + \sin A \sin C & \cos A \sin B \sin C + \sin A \cos C & \cos A \cos B \end{pmatrix}
\end{aligned}$$

- بنابراین کد این بخش به شکل رو به رو خواهد شد.

```python
import time
import serial
import numpy as np
from vpython import *

arduinoData = serial.Serial('com7', 115200)
time.sleep(100)

toRad = np.pi/180.0
toDeg = 1/toRad

while True:
    while arduinoData.inWaiting() == 0:
        pass
    dataPacket = arduinoData.readline()
    try:
        dataPacket = str(dataPacket, 'utf-8')
        splitPacket = dataPacket.split(",")
        roll = float(splitPacket[0])*toRad
        pitch = float(splitPacket[1])*toRad
        yaw = float(splitPacket[2])*toRad
        print('roll,pitch,yaw')
        print(roll*toDeg, pitch*toDeg, yaw*toDeg)

        Qx = np.array( [[1,0,0 ],[0,np.cos(roll),-np.sin(roll)],[0,np.sin(roll),np.cos(roll)]])
        Qy = np.array( [[np.cos(pitch),0,np.sin(pitch)],[0,1,0],[-np.sin(pitch),0,np.cos(pitch)]])
        Qz = np.array( [[np.cos(yaw),-np.sin(yaw),0],[np.sin(yaw),np.cos(yaw),0],[0,0,1]])

        print('rotation matrix')
        Qxyz = np.matmul(np.matmul(Qx,Qy),Qz)
        print(Qxyz)
    except:
        pass
```

حال میخواهیم با کمک Quaternions ماتریس دوران را پیدا کنیم. ماتریس quaternions دارای 4 سطر است که انها را q0,q1,q2,q3 نام گذاری میکنیم. حال رابطه زیر برای پیدا کردن ماتریس دوران برقرار است.

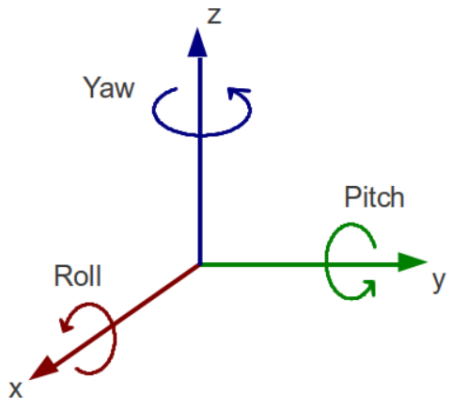
$$R(Q) = \begin{bmatrix} 2(q_0^2 + q_1^2) - 1 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & 2(q_0^2 + q_2^2) - 1 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & 2(q_0^2 + q_3^2) - 1 \end{bmatrix}$$

- کد این بخش به شکل زیر می شود:

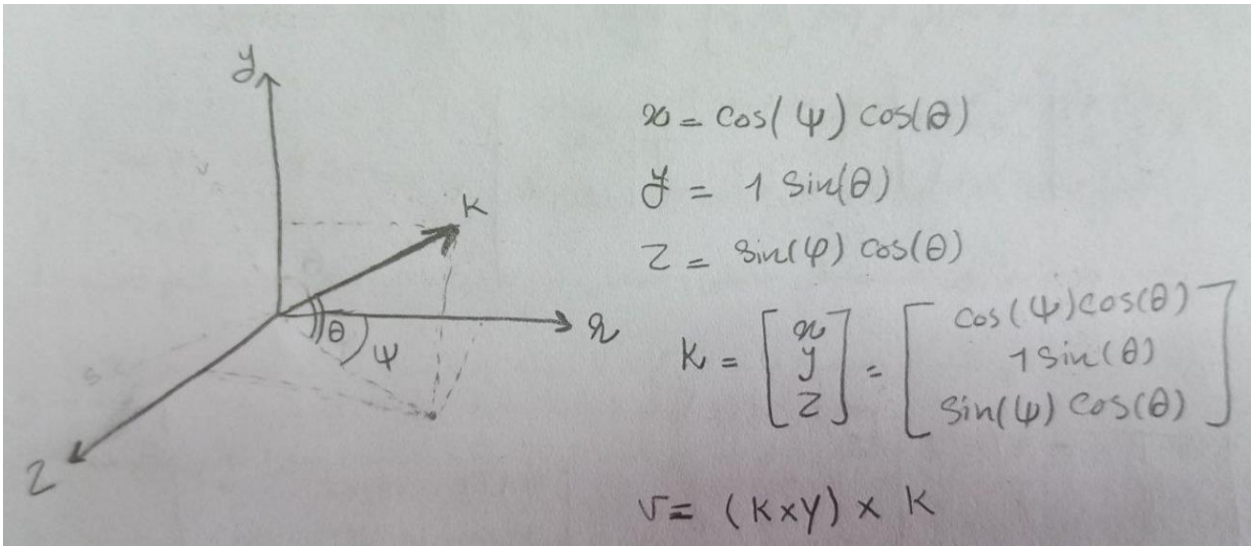
```
while True:
    while arduinoData.inWaiting() == 0:
        pass
    dataPacket = arduinoData.readline()
    try:
        dataPacket = str(dataPacket, 'utf-8')
        splitPacket = dataPacket.split(",")

        q0 = float(splitPacket[0])
        q1 = float(splitPacket[1])
        q2 = float(splitPacket[2])
        q3 = float(splitPacket[3])

        print('Quaternions')
        print(q0,q1,q2,q3)

        rm11 = 2 * (q0 * q0 + q1 * q1) - 1
        rm12 = 2 * (q1 * q2 - q0 * q3)
        rm13 = 2 * (q1 * q3 + q0 * q2)
        rm21 = 2 * (q1 * q2 + q0 * q3)
        rm22 = 2 * (q0 * q0 + q2 * q2) - 1
        rm23 = 2 * (q2 * q3 - q0 * q1)
        rm31 = 2 * (q1 * q3 - q0 * q2)
        rm32 = 2 * (q2 * q3 + q0 * q1)
        rm33 = 2 * (q0 * q0 + q3 * q3) - 1

        rotation_matrix = np.array([[rm11, rm12, rm13],[rm21, rm22, rm23],[rm31, rm32, rm33]])
        print('rotation matrix')
        print(rotation_matrix)
    except:
        pass
```

کتابخانه VPython

در این بخش با کمک roll , pitch , yaw میخواهیم دوران جسم را شبیه سازی کنیم. همانند قبل با کمک کتابخانه serial اطلاعات را از ژیروسکوپ میگیریم. سپس به صورت زیر عمل میکنیم.



برای مثال بردار زیر را در نظر میگیریم. میخواهیم تصویر بردار k را بر هر محور پیدا کنیم. اگر فرض کنیم زاویه بردار k با محور X و y را داریم. روابط به صورت زیر خلاصه می شوند.



بردار k نشان دهنده محور بردار است و میخواهیم بردار های عمود بر k را پیدا کنیم تا بتوانیم ان را نمایش دهیم. Vrot بردار عمود بر k است که به روش زیر به دست می اید.

$$\mathbf{v}_{\text{rot}} = \mathbf{v}\cos\theta + (\mathbf{k}\times\mathbf{v})\sin\theta + \mathbf{k}\,(\mathbf{k}\cdot\mathbf{v})(1-\cos\theta)$$

از ضرب خارجی بردار k و vrot میتوانیم بردار عمود بر این دو را پیدا کنیم که همان بردار جانبی است.

$$side\ Arrow = V_{rot} \times k$$

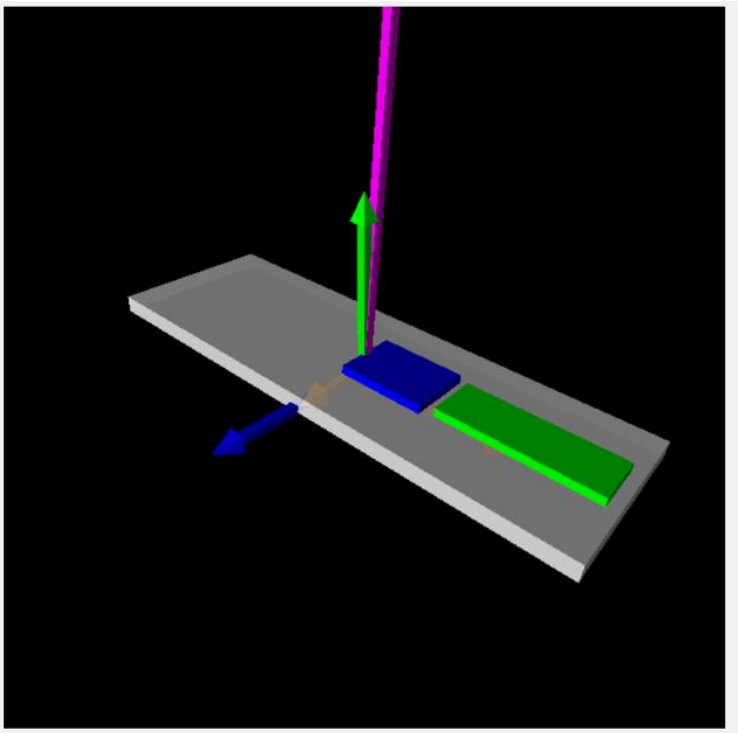
حال میتوانیم در کد پایتون برداری که با vpython نمایش میدهیم را اپدیت کنیم.

```python
import time
import serial
import numpy as np
from vpython import *

arduinoData = serial.Serial('com7', 115200)
time.sleep(100)

toRad = np.pi/180.0
toDeg = 1/toRad

scene.range = 4
scene.forward = vector(-1,-1,-1)
scene.width = 600
scene.height = 600
xarr = arrow(length=2 , shaftwidth=0.1 ,color=color.red, axis=vector(1,0,0))
yarr = arrow(length=2 , shaftwidth=0.1 ,color=color.green, axis=vector(0,1,0))
zarr = arrow(length=2 , shaftwidth=0.1 ,color=color.blue, axis=vector(0,0,1))

frontarr = arrow(length=4 ,shaftwidth=0.1 ,color=color.purple,axis=vector(1,0,0))
uparr = arrow(length=1,shaftwidth=0.1 ,color=color.magenta,axis=vector(0,1,0))
sidearr = arrow(length=1,shaftwidth=0.1 ,color=color.orange,axis=vector(0,0,1))
bBoard = box(length=6,width=2,height=0.2,opacity=0.8,pos=vector(0,0,0))
sensor = box(length=1,width=0.75,height=0.1,color=color.blue,pos=vector(0.5,0.15,0))
arduino = box(length=1.75,width=0.6,height=0.1,color=color.green,pos=vector(2,0.15,0))
obj = compound([bBoard,sensor,arduino])

while True:
    while arduinoData.inWaiting() == 0:
        pass
    dataPacket = arduinoData.readline()
    try:
        dataPacket = str(dataPacket, 'utf-8')
        splitPacket = dataPacket.split(",")
        roll = float(splitPacket[0])*toRad
        pitch = float(splitPacket[1])*toRad
        yaw = float(splitPacket[2])*toRad
        print('roll,pitch,yaw')
        print(roll*toDeg, pitch*toDeg, yaw*toDeg)

        k = vector(cos(yaw)*cos(pitch),sin(pitch),sin(yaw)*cos(pitch))
        y = vector(0,1,0)
        s = cross(k,y)
        v = cross(s,k)
        vrot = v*cos(roll)+cross(k,v)*sin(roll)
        frontarr.axis = k
        uparr.axis = vrot
        sidearr.axis = cross(k,vrot)
        obj.axis = k
        obj.up = vrot
        frontarr.length = 2
        uparr.length = 4
        sidearr.length = 1

        Qx = np.array( [[1,0,0 ],[0,cos(roll),-sin(roll)],[0,sin(roll),cos(roll)]])
        Qy = np.array( [[cos(pitch),0,sin(pitch)],[0,1,0],[-sin(pitch),0,cos(pitch)]])
        Qz = np.array( [[cos(yaw),-sin(yaw),0],[sin(yaw),cos(yaw),0],[0,0,1]])

        print('rotation matrix')
        Qxyz = np.matmul(np.matmul(Qx,Qy),Qz)
        print(Qxyz)
    except:
        pass
```

```
5
6   arduinoData = serial.Serial('com7', 115200)
7   time.sleep(100)
8
9   toRad = np.pi/180.0
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

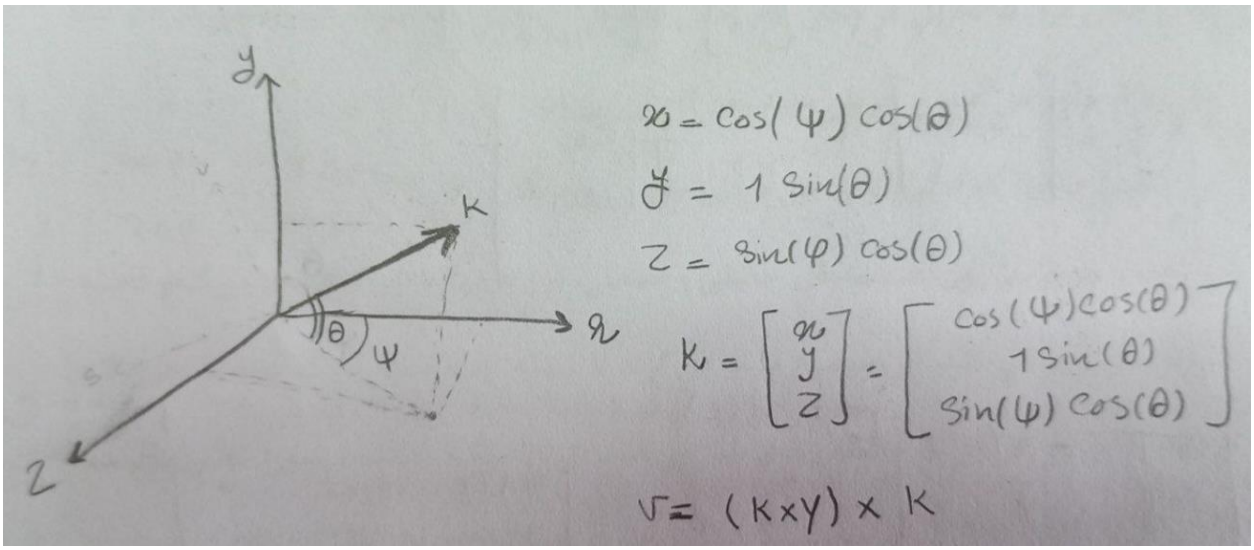
```
[[ 0.99614582  0.08382379 -0.025828  ]
 [-0.08192349  0.9943522   0.06747038]
 [ 0.03133775 -0.06509442  0.99738692]]
roll,pitch,yaw
-3.87 -1.48 -4.81
rotation matrix
[[ 0.99614582  0.08382379 -0.025828  ]
 [-0.08192349  0.9943522   0.06747038]
 [ 0.03133775 -0.06509442  0.99738692]]
roll,pitch,yaw
-3.87 -1.48 -4.81
rotation matrix
[[ 0.99614582  0.08382379 -0.025828  ]
 [-0.08192349  0.9943522   0.06747038]
 [ 0.03133775 -0.06509442  0.99738692]]
roll,pitch,yaw
-3.86 -1.48 -4.81
rotation matrix
[[ 0.99614582  0.08382379 -0.025828  ]
 [-0.08192896  0.99436354  0.0672963 ]
 [ 0.03132345 -0.06492087  0.99739868]]
roll,pitch,yaw
-3.87 -1.48 -4.81
rotation matrix
[[ 0.99614582  0.08382379 -0.025828  ]
 [-0.08192349  0.9943522   0.06747038]
 [ 0.03133775 -0.06509442  0.99738692]]
roll,pitch,yaw
-3.87 -1.48 -4.81
rotation matrix
[[ 0.99614582  0.08382379 -0.025828  ]
 [-0.08192349  0.9943522   0.06747038]
[[ 0.03133775 -0.06509442  0.99738692]]
```



- یکی از مشکلات روش roll , pitch , yaw این است که در تعقیب حرکت جسم هنگامی که زاویه از 90 درجه بیشتر شود یا از 90- کمتر شود ، دچار مشکل می شود و نمیتواند ان حرکت را انجام دهد.دلیل این موضوع این است در یک دوران برای رسیدن به ان دوران صرفا یک مسیر وجود ندارد و بنابراین حسگر دچار خطا می شود و مسیر کوتاه تر را شناسایی میکند. بنابراین به سراغ استفاده از quaternions میرویم که این مشکل را ندارد .

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan \frac{2(q_0 q_1 + q_2 q_3)}{1-2(q_1^2+q_2^2)} \\ \arcsin(2(q_0 q_2 - q_3 q_1)) \\ \arctan \frac{2(q_0 q_3 + q_1 q_2)}{1-2(q_2^2+q_3^2)} \end{bmatrix}$$

با داشتن quaternions از روابط بالا میتوان roll , pitch , yaw را پیدا کرد. از تابع atan2 استفاده کرد که سینوس و کسینوس را میگیرد و زاویه دقیق را میدهد. حال بعد از پیدا کردن roll , pitch , yaw از همان روابط سوال قبل مقادیر بردار ها را اپدیت میکنیم.



برای پیدا کردن ماتریس دوران هم همانند سوال قبل از روش زیر استفاده میکنیم و مشکل مطرح شده حل می شود.

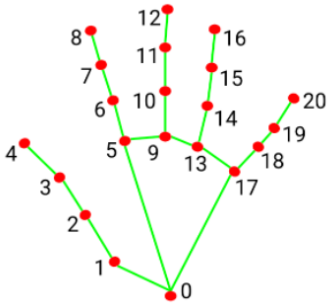
$$R(Q) = \begin{bmatrix} 2(q_0^2 + q_1^2) - 1 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & 2(q_0^2 + q_2^2) - 1 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & 2(q_0^2 + q_3^2) - 1 \end{bmatrix}$$

کاربردهای حسگرهای حرکتسنج

- بنظر من از مهمترین کاربرد های حرکتسنج در زندگی روزمره این است که در تلفن همراه کاربرد دارد و در برنامه health گوشی حرکت و قدم ها را تشخیص میدهد. از طرفی حرکت و چرخش و جهت گیری را تشخیص میدهد . از حرکت سنج ها در موشک ها و هواپیما ها نیز استفاده می شود تا بتوانند به صورت دقیق مقدار دوران جسم را اندازه گیری کنند . در عکاسی 360 درجه این حسگر ها کمک میکنند تا زاویه را تشخیص دهیم. همچنین این حسگر به جهت یابی و کاهش لرزش دوربین هنگام عکس برداری کمک میکند.

بخش دوم

- در این بخش با کمک لینک داده شده یک کد پیش فرض و یک نمایش از 21 نقطه داده شده میبینیم.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

رابطه ای که در این بخش میخواهم استفاده کنم این است که 3 نقطه از نقاط بالا را انتخاب کرده و دو بردار با ان میسازیم با کمک جمع و ضرب خارجی این دو بردار میتوان دو بردار عمود برهم پیدا کرد که میتواند به عنوان دو محور مورد نیاز برای اعمال تغییرات در vpython استفاده شود.

- من از نقاط 0 – 5 – 17 استفاده کردم به این دلیل که این نقاط کف دست هستند و بهترین محور ها را فراهم میکنند. طوری که بردار اول از نقطه 0 به 5 و بردار دوم از نقطه 0 به 17 در نظر میگیریم. حاصل جمع انها را بردار x و ضرب خارجی انها را z میگیریم. استفاده است حسگر های ژیروسکوپ مزیت بیشتری نسبت به mediapipe دارد زیرا این کتابخانه هنگام تشخیص این 21 نقطه دچار خطا می شود ولی ژیروسکوپ نتایج بهتری و قابل اطمینان تری دارد.