مینی پروژه یک درس رباتیک – علی رشیدی مقدم – 810100357

توضیحات اولیه:

برای سخت افزار این پروژه از یک برد آردوینو مدل UNO و یک سنسور IMU با مدل MPU6050 استفاده شد. برای سهولت در تغییر دادن زاویه ها و همچنین جذاب تر کردن بازی طراحی شده، سنسور بر روی یک بطری نصب شده است. نمایی از ستاپ را در عکس زیر می بینید:

همچنین در تابع setup که در آردوینو نوشته شد، مراحلی را برای کالیبراسیون سنسور و تخمین بایاس ژیروسکوپ در نظر گرفته شد و به همین دلیل در هنگام اجرای کد ها چند ثانیه طول میکشد تا دیتای سنسور ارسال شود و پس از آن بطور مداوم دیتا ارسال می شود.

همچنین با بررسی های عملی انجام شده، مشخص شد که این کتابخانه مقدار زاویه Yaw و Pitch را بطور قرینه برمیگرداند در صورتی که اگر از داده های ژیروسکوپ انتگرال بگیریم مقدار آنها در جهت استاندارد آن بدست می آید. بنابراین مقداریاو و پیچ خروجی کتابخانه را قرینه کردیم. خروجی کواترنیون صحیح بود و نیازی به اصلاح نداشت.

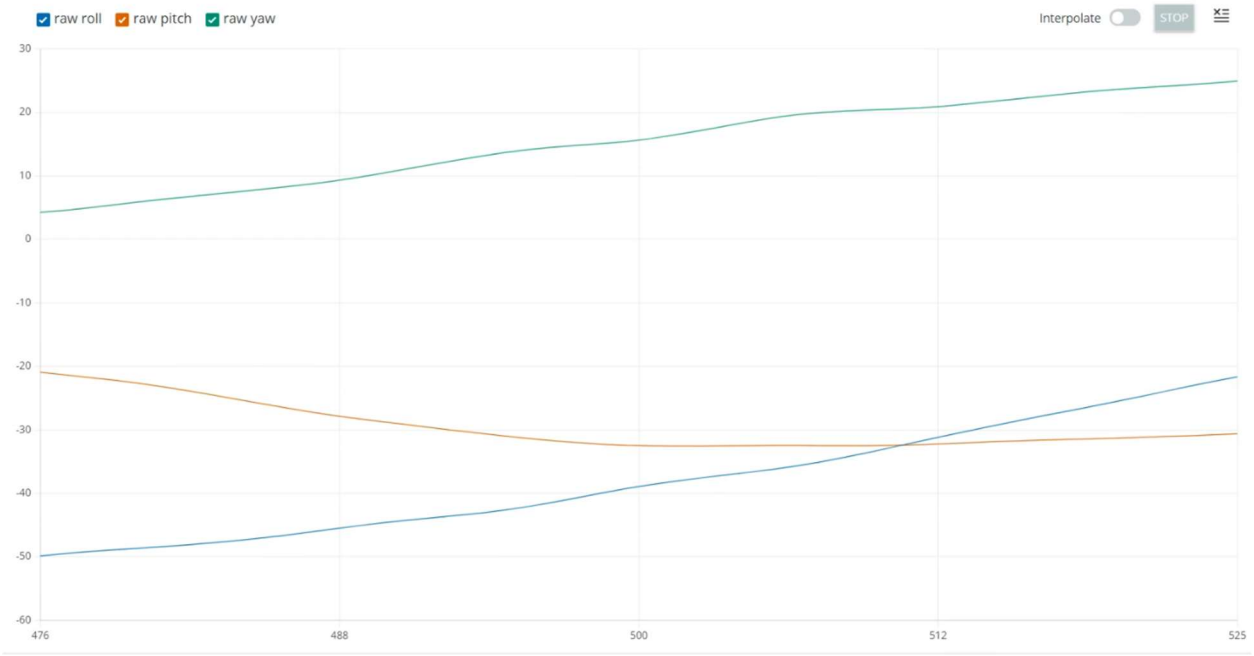
سوال 1:

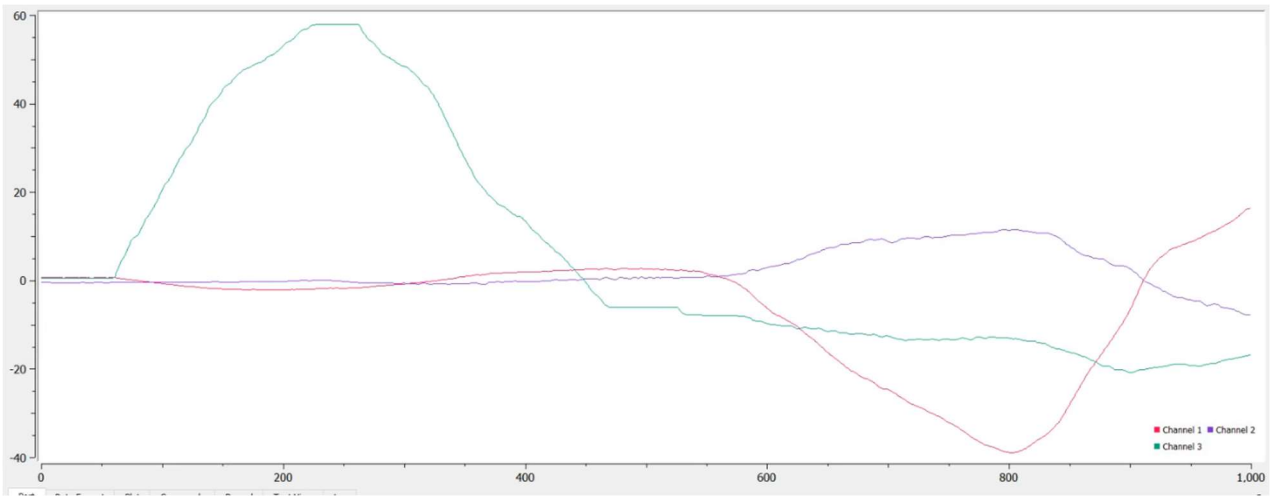
الف) در این قسمت سنسور بطور تصادفی حول محور های مختلف چرخانده می شود و مقادیر کواترنیون و زاویای اویلر نیز در سریال مانیتور چاپ می شود.

تصویری از سریال مانیتور را در زیر می توان مشاهده کرد. همچنین ویدویوی اجرای real time نیز در پوشه مربوط به ویدیوها با نام 1-A موجود است.

```
13:57:47.172 -> raw w : 0.91, raw x : 0.08, raw y : -0.38, raw z : 0.14,raw roll : 3.09, raw pitch : 45.63, raw yaw : -25.33
13:57:47.172 -> raw w : 0.91, raw x : 0.08, raw y : -0.39, raw z : 0.15,raw roll : 2.06, raw pitch : 46.55, raw yaw : -27.62
13:57:47.205 -> raw w : 0.90, raw x : 0.07, raw y : -0.39, raw z : 0.17,raw roll : -0.44, raw pitch : 46.79, raw yaw : -29.07
13:57:47.205 -> raw w : 0.90, raw x : 0.05, raw y : -0.39, raw z : 0.18,raw roll : -4.20, raw pitch : 46.38, raw yaw : -29.83
13:57:47.205 -> raw w : 0.90, raw x : 0.02, raw y : -0.39, raw z : 0.19,raw roll : -8.92, raw pitch : 45.34, raw yaw : -30.03
13:57:47.238 -> raw w : 0.90, raw x : -0.01, raw y : -0.39, raw z : 0.20,raw roll : -14.13, raw pitch : 43.76, raw yaw : -29.53
13:57:47.238 -> raw w : 0.90, raw x : -0.05, raw y : -0.38, raw z : 0.21,raw roll : -19.77, raw pitch : 41.66, raw yaw : -28.69
13:57:47.238 -> raw w : 0.89, raw x : -0.10, raw y : -0.38, raw z : 0.22,raw roll : -25.79, raw pitch : 38.99, raw yaw : -27.70
13:57:47.238 -> raw w : 0.89, raw x : -0.14, raw y : -0.37, raw z : 0.23,raw roll : -31.83, raw pitch : 35.76, raw yaw : -26.66
13:57:47.270 -> raw w : 0.88, raw x : -0.19, raw y : -0.36, raw z : 0.25,raw roll : -37.59, raw pitch : 32.58, raw yaw : -25.59
13:57:47.270 -> raw w : 0.86, raw x : -0.24, raw y : -0.36, raw z : 0.26,raw roll : -43.45, raw pitch : 29.56, raw yaw : -24.57
```

ب) در این قسمت هم مانند قبل عمل شد. تصاویر مربوط به سریال پلاتر آردوینو و SerialPlot را در زیر می توان مشاهده نمود. فیلم های این قسمت در پوشه مربوط به ویدیوها با نام های 1-B-Plotter و موجود است.





تفاوت بین این دو پلات این است که در آردوینو محور x مطابق با نرخ ارسال داده های سریال تنظیم می شود یعنی مثلا داده ای که در تایم استپ 300 نمایش داده می شود، 300 امین داده ای است که آردوینو ارسال کرده است اما در SerialPlot محور x ، بر حسب میلی ثانیه است.

سوال 2:

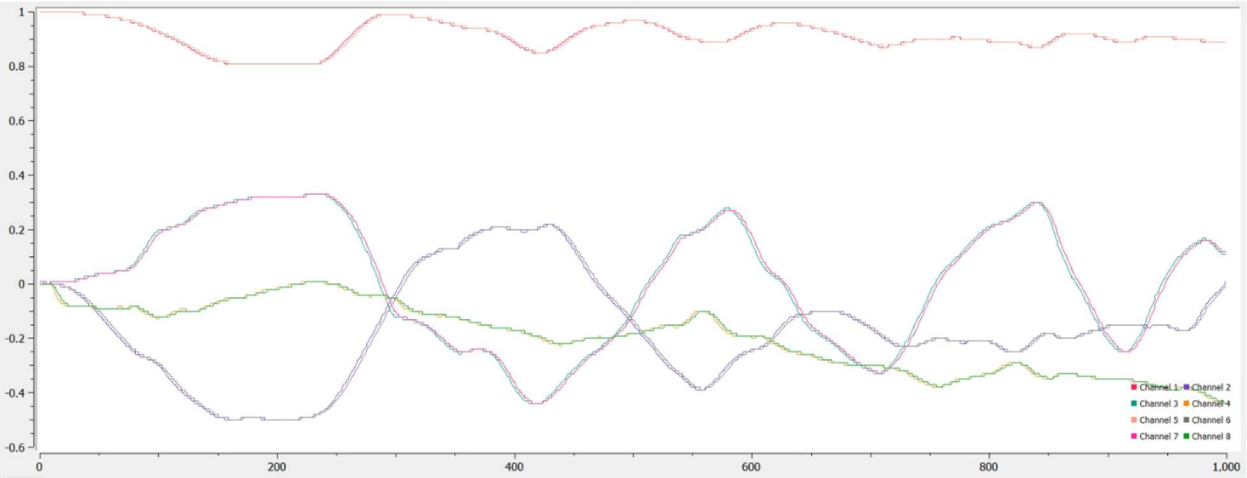
الف) برای پیاده سازی فیلتر کالمن از روابط بازگشتی زیر استفاده شد:

1: **while** true **do**
2: $\Rightarrow$ Obtain noisy U_1 (at each time step)
3: $\Rightarrow$ GOTO *KALMAN*, obtain *filtered* U_1 (at each time step)
4: **end while**
5: **function** $KALMAN(U)$
6: $\Rightarrow \mathcal{K} = \frac{P \cdot H}{H \cdot P \cdot H + \mathcal{R}}$ (update Kalman Gain)
7: $\Rightarrow \hat{U}+ = \mathcal{K} \cdot [U - H \cdot \hat{U}]$ (update estimate)
8: $\Rightarrow P = (1 - \mathcal{K} \cdot H) \cdot P + Q$ (update error estimate)
9: $\Rightarrow$ *We now have* P_{k+1} *to use with the next* U_{k+1}
10: $\Rightarrow$ **return** $\hat{U}$

که در آن R کوواریانس نویز، H مقیاس اندازه گیری ،Q کواریانس خطا،P کوواریانس اولیه خطا و K گین کالمن است. در ابتدا K وP را صفر در نظر میگیریم و با توجه به رابطه بازگشتی بالا و عملکرد تخمین زن مقادیر آنها آپدیت می شود. کوواریانس نویز و خطا بطور تجربی و با آزمون و خطا R=10 و Q=1 در نظر گرفته شد.

در اینجا استفاده از این فیلتر منجر به نرم تر کردن داده های سنسور می شود اما مقدار کمی تاخیر را وارد محاسبات می کند ولی سرعت قابل قبولی دارد و برای داده های Real Time مناسب است.

نمونه ای از عملکرد این فیلتر در تصویر زیر مشاهده می شود و فیلم مربوط به آن در پوشه ویدیو ها با نام 2-A-Kalman موجود است.

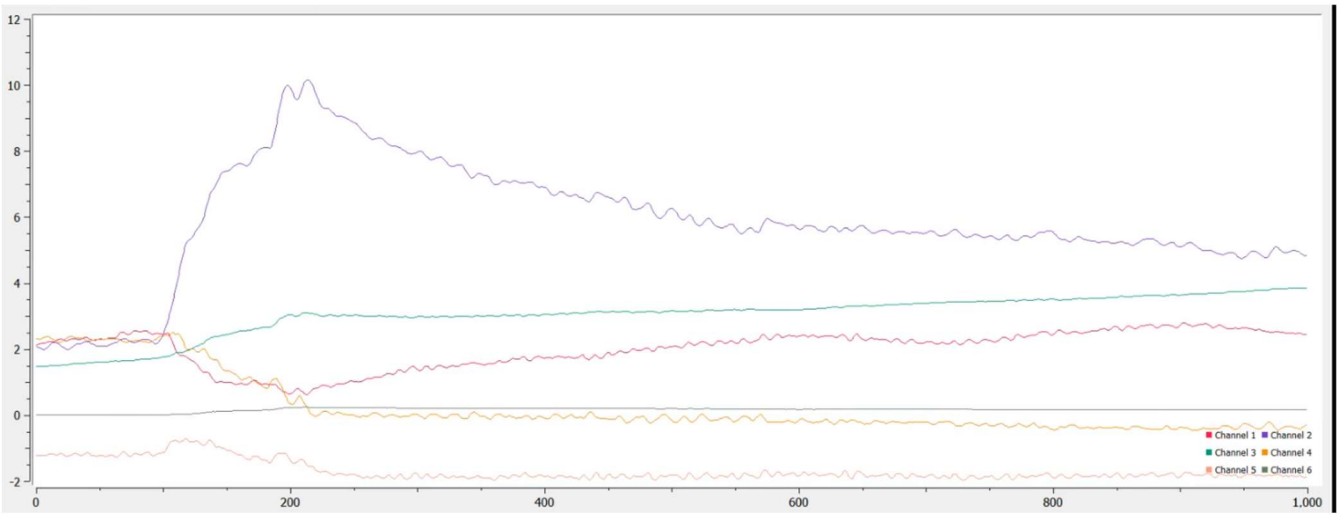


برای پیاده سازی فیلتر کامپلمنتری درجه یک از روابط زیر استفاده شد:

$$\text{Angle} = \alpha_1 \ (\text{angle} + \text{gyroscope data} * dt) + \alpha_2 \ (\text{accelerometer data})$$

در اینجا a1=0.96 و a2=0.04 در نظر گرفته شده است. در تغییرات کند از داده های ژیروسکوپ استفاده می شود و درتغییرات سریع از داده های شتاب سنج. به دلیل آن که زوایای دوران بطور مستقیم محاسبه نمیشوند و با انتگرال از ژیروسکوپ بدست می آیند همواره امکان انباشتگی وجود دارد و در اینجا سعی می شود تا حدی این مشکل بر طرف شود. این فیلتر قابلیت دنبال کردن تغییرات گذرا را ندارد.

نمونه ای از عملکرد این فیلتر در تصویر زیر مشاهده می شود و فیلم مربوط به آن در پوشه ویدیو ها با نام 2-A-comp1 موجود است.



ج) برای بررسی موارد خواسته شده به این صورت عمل شد:

پیچیدگی: برای بررسی پیچیدگی فیلترها یک معیار میزان ظرفیتی از که از حافظه میکرو اشغال می کند می باشد. کامپلمنتری از داده های ژیروسکوپ و شتاب سنج استفاده می کند و کالمن از خود زوایا (که از طریق کواترنیون محاسبه می شوند) استفاده می کند. فیلتر کامپلمنتری تنها یک پارامتر قابل تنظیم دارد که می توان بطور تجربی آن را تنظیم کرد. در فیلتر کالمن باید کوواریانس نویز و خطا را یا محاسبه کنیم یا بطور تجربی معین کنیم. همچنین باید با برای K و P هم که یک مقدار دهی اولیه داشته باشیم و سپس باز هم در رابطه بازگشتی کالمن این مقادیر آپدیت می شوند.

فیلتر کامپلنتری 55 درصد از حافظه را اشغال می کند و فیلتر کالمن 53 درصد. از لحاظ محاسباتی هیچ کدام پیچیدگی زیادی ندارند. اما در مجموع کالمن پیچیده تر است.

دقت: برای بررسی این معیار، سنسور را حول محور رول به اندازه 90 درجه و به آرامی می چرخانیم و خروجی هر فیلتر را بررسی میکنیم. چنانچه حالت گذرای داده ها را در نظر نگیریم و تنها به مقدار نهایی آنها توجه داشته باشیم هر دوی این فیلتر عملکرد مشابهی خواهند داشت اما فیلتر کامپلمنتری سریع ترمقدار واقعی زاویه را نشان می دهد.

سرعت: برای بررسی این معیار، سنسور را حول محور رول به اندازه 90 درجه و به آرامی می چرخانیم و خروجی هر فیلتر را بررسی میکنیم. فیلتر کامپلمنتری عملکرد سریع تری دارد و برای فیلتر کالمن مدت زمانی طول می کشد تا دیتای خام سنسور پس از یک جهش به حدود زاویه واقعی برسد. از آنجا که فیلتر کالمن تنها نمودار دیتای خام را نرم تر می کند بنابراین در اینجا فیلتر کامپلمنتری عملکرد بهتری دارد.

عملکرد در تغییرات سریع: در تغییرات سریع فیلتر کالمن بهترین عمل می کند و تغییرات را با سرعت خوبی دنبال میکند اما فیلتر کامپلمنتری تاخیر زیادی دارد و برای کاربرد های که حالت های گذرا مهم هستند، مناسب نیست.

Filter	Complexity	Accuracy	Speed	Rapid change
Kalman	7	7	5	8
Complementary	5	7	8	4

در مجموع اینکه کدام فیلتر بهتراست بستگی به کاربرد دارد. در کاربرد های که سرعت تغییرات پایین است یا حالت گذرا اهمیتی ندارد و دقت خوبی نیاز است، کامپلمنتری انتخاب بهتری است در کاربرد هایی که سرعت تغییرات بالاست و حالت گذرا مهم است فیلتر کالمن بهتر است.

سوال 3:

در اینجا دوران از نوع محلی و با کانونشن XYZ است بنابراین ماتریس هر دوران از ابتدا به انتها ضرب می شود و ماتریس دوران محاسبه می شود.

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

دورانی که بر حسب کواترنیون ها محاسبه می شود هم بر اساس رابطه لاگرانژ زیر محاسبه می شود.

$$
R(Q) = \begin{bmatrix} 2(q_0^2 + q_1^2) - 1 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & 2(q_0^2 + q_2^2) - 1 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & 2(q_0^2 + q_3^2) - 1 \end{bmatrix}
$$

نهایتا هر 10 ثانیه یک بار ماتریس دوران را هم از روی رول، پیچ و یاو و هم از روی کواترنیون ها نمایش می دهیم . دو نمونه از آن در زیر نمایش داده شده است.

```
Q(r,p,y) =
[[-0.9438273   0.31781638 -0.09045877]
 [-0.17138946 -0.704888   -0.68830127]
 [-0.28251672 -0.6341338   0.7197629 ]]

Q(w,x,y,z) =
[[-0.9334  0.3054 -0.1878]
 [-0.3498 -0.6614  0.6626]
 [ 0.0786  0.6842  0.721 ]]
```

```
Q(r,p,y) =
[[ 0.9397068   0.15944566  0.30253625]
 [-0.18114987  0.98243     0.04489894]
 [-0.29006177 -0.09699625  0.9520798 ]]

Q(w,x,y,z) =
[[ 0.921   0.1536  0.3152]
 [-0.16    0.972  -0.006 ]
 [-0.312  -0.0452  0.9336]]
```

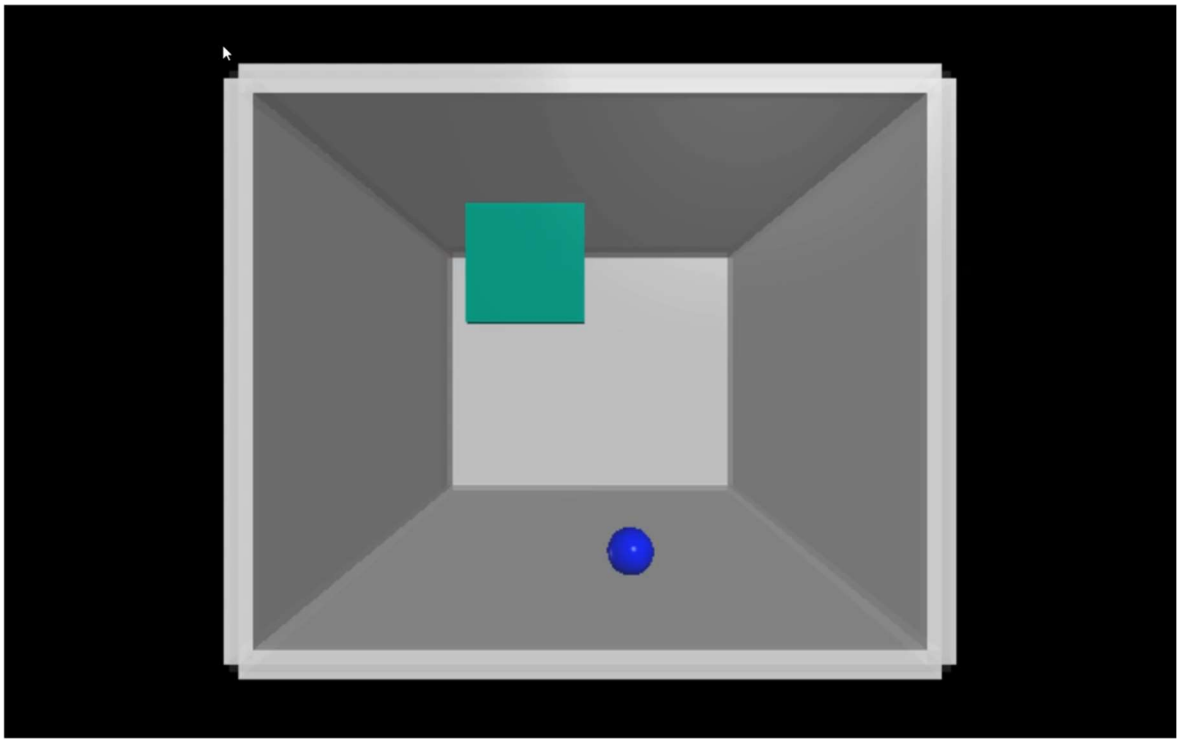
سوال 4:

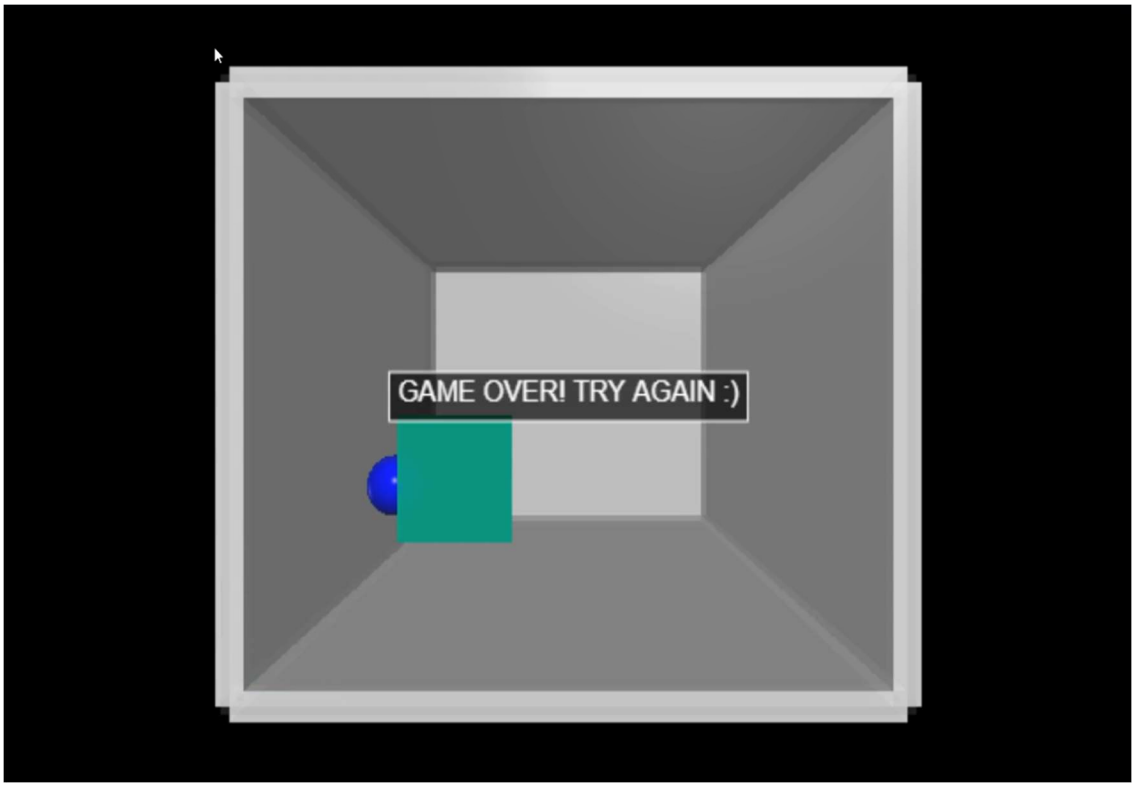
برای ساخت این بازی ابتدا سه دیوار ، یک توپ و یک صفحه می سازیم. توپ با سرعت ثابتی از نقطه ای شروع به حرکت می کند. چنانچه به دیوار یا به صفحه Paddle برخورد کند، بر میگردد. چنانچه توپ هم راستا با Paddle قرار گیرد اما به آن برخورد نکند به معنی باخت است. همچنین برای بهبود رابط گرافیکی هز زمان توپ به Paddle برخورد کند رنگ آن به زرد تغییر میکند.

برای جذاب تر کردن بازی، هر زمان که بازیکن ببازد توپ با جهت متفاوتی و بصورت تصادفی شروع به حرکت میکند.

برای اینکه بتوانیم با این سنسور، Paddle را تکان دهیم، تغییرات زوایای رول و پیچ را به تغییر مختصات Paddle تبدیل می کنیم و چون با تغییرات سریع و نرم نیاز داریم از داده های فیلتر کالمن استفاده می کنیم. همچنین برای کارکردن راحت تر با سنسور آن را روی یک بطری نصب کرده تا ظاهری شبیه به Joy Stick داشته باشد. برای راحتی زوایای دوران را بین 60- و 60+ محدود می کنیم و چنانچه بیش از 60 یا کمتراز 60- هم زاویه ای اعمال شود وارد ناحیه اشباع می شود.

تصاویری از محیط بازی را در زیر می بینید. فیلم انجام بازی بصورت عملی و همچنین با اسکرین رکورد در پوشه ویدیو ها با نام های 4-Real و 4-Screen وجود دارد.




GAME OVER! TRY AGAIN :)