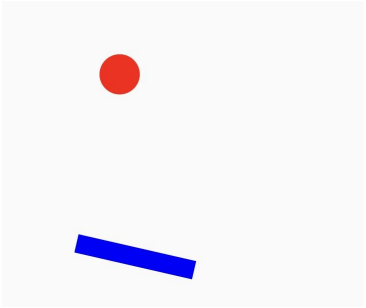


Motion Controlled Pong Game

05.27.2023

دانشگاه تهران

پردیس دانشکده های فنی

دانشکده برق و کامپیوتر

PONG!

پرنا اسدی (810198498)

مرتضی بهجت (810198363)

سید امین محمد اطیابی (810198559)

مرتضی نوری (810198481)

بخش صفر) توضیحات انجام پروژه

برای پیاده سازی این پروژه از زبان Kotlin استفاده شده است. قسمت امتیازی نیز پیاده سازی شده است.

بخش اول) نحوه شکست کار بین اعضا

اطیابی	پیاده سازی و دیباگ قسمت سنسور ها - تست
نوری	تحلیل و طراحی موارد مربوط به سنسور - تست
بهجت	پیاده سازی و دیباگ قسمت فیزیک توپ و حرکت - تست
اسدی	گزارش کار و تحلیل و فرضیات مربوط به فیزیک توپ - تست

بخش دوم) دموی پروژه

- ویدیو در ضمیمه ارسال شده است.

بخش سوم) مستندات و فرضیات پیاده سازی

- کتابخانه های استفاده شده

- **Jetpack compose:** A modern UI toolkit for building native Android apps with a declarative approach. It enables developers to build rich and responsive user interfaces using a simple, composable, and reactive programming model.
- **Hilt:** Hilt is a dependency injection library for Android that reduces the boilerplate of doing manual dependency injection in your project
- **Other default libraries**

- فرضیات

- برای حرکت توپ، بر اسی میشود.
- فضای حرکت دست به فاصله 50 سانتی متر تعریف شده است و سرعت حرکت دست بر سرعت حرکت تخته تاثیر خواهد داشت.
- فرضیات دیگر در عناوین بعدی شرح داده میشود.

- مستند کد های استفاده شده (ساختار برنامه اندرویدی)

- **PongApplication:** Entry point of the app
- **GameConfig:** Configuration
- **MainViewModel:** sets up the game configuration, the coordinates and objects (Ball, brick and board). The bridge between physical and software. Used MVVM architecture. Saving states of activity.
- **MainActivity:** Implement UI elements and redraw surfaces with given new data.
- **Models:** they are classes of objects which have their funtions. These models are Ball, Break, Board and Coordinate and Orientation (data structures).

- **سنسور های استفاده شده**

- TYPE_LINEAR_ACCELERATION: A sensor type in Android devices that measures the acceleration force applied to a device, excluding the force of gravity
- TYPE_ROTATION_VECTOR: A constant representing a sensor type in Android devices that measures the orientation of a device relative to the Earth's magnetic field.

- **قواعد فیزیکی در حرکت توپ**

با استفاده از قوانین زیر، سرعت حرکت توپ محاسبه میشود. در دو راستای x و y ، توپ دارای سرعت حرکتی است که شامل مقدار و جهت آن است. به دلیل تغییر مبدا صفحه به گوشه شمال-غربی صفحه، فرمول ها به صورت زیر محاسبه می شوند، سرعت جدید بر حسب زاویه تخته و زمین و سرعت قبلی توپ به دست می آید.

val newVx = cos(2 * angle) * vx + sin(2 * angle) * vy

val newVy = -cos(2 * angle) * vy + sin(2 * angle) * vx

- **قواعد تغییر زاویه تخته**

تخته بازی در ابتدا دارای زاویه و سرعت 0 است. با کمک سنسور TYPE_ROTATION_VECTOR می توان زاویه دستگاه را در دنیای فیزیکی بر حسب رادیان به دست آورد و بر تخته اعمال کرد.

- قواعد حرکت تخته

زمینه بازی، شتاب حرکتی در سه محور را دریافت میکند و با توجه به مقادیر آنها، به تخته دستور حرکت در جهت شتاب اعمال شده میدهد. تخته بازی در ابتدا دارای زاویه و سرعت 0 است. شتاب اعمال شده در دنیای فیزیکی مقدار خیلی کمتری نسبت به نرم افزار دارد و با یک ضریب و محاسبه مشخص، مقدار آن در زمینه بازی تراز خواهد شد. همچنین سرعت قبلی تخته نیز در این حرکت جدید تاثیر میگذارد و سرعت جدید برای تخته اعمال خواهد شد. فرمول آن به صورت زیر است. این مقادیر با سنسور TYPE_LINEAR_ACCELERATION به دست آمده است.

val output = alpha * acceleration + (1 - alpha) * lastAx

خروجی حاصل از سرعت جدید، در 4 ضرب شده و با توجه به زاویه تخته با مبدا و محدودیت برخورد به دیواره مشخص شده، به تخته اعمال خواهد شد(تابع applyAcceleration)

- قواعد برخورد توپ با تخته و دیوار

در صورت برخورد با دیوار، سرعت حرکتی در دو محور، قرینه خواهد شد و توپ در جهت قرینه حرکت خود ادامه خواهد داد. زمینه بازی، برخورد با توپ و تخته را در فرکانس های مشخص چک میکند. نحوه برخورد توپ با تخته نیز، به دو حالت تقسیم میشود 1- برخورد با طول 2- برخورد با عرض. که در هر دو حالت، سرعت زاویه ای توپ محاسبه شده و توپ با سرعت جدید حرکت خواهد کرد. البته در جریان محاسبات، توپ به صورت یک نقطه و تخته به صورت یک خط مدلسازی شده است، برخورد توپ و تخته بر حسب فرمول های فاصله نقطه از خط، و فاصله نقطه از نقطه محاسبه میشود و پس از آنکه نقطه توپ در محدوده مجموعه شعاع و نصب عرض تخته قرار گرفت، برخورد تشخیص داده میشود.

```
if (lineDistance < height / 2 + ball.radius && pointDistance < lastDistance) {

    val xOffset = sqrt(pointDistance - lineDistance.pow(2))
    if (xOffset < width / 2) {
        ball.rotate(zAngle)
    } else if (xOffset < width / 2 + ball.radius) {
        ball.rotate(zAngle + Math.PI.toFloat() / 2)
    }
}
```

بخش چهارم) بخش امتیازی با استفاده از شتاب سنج

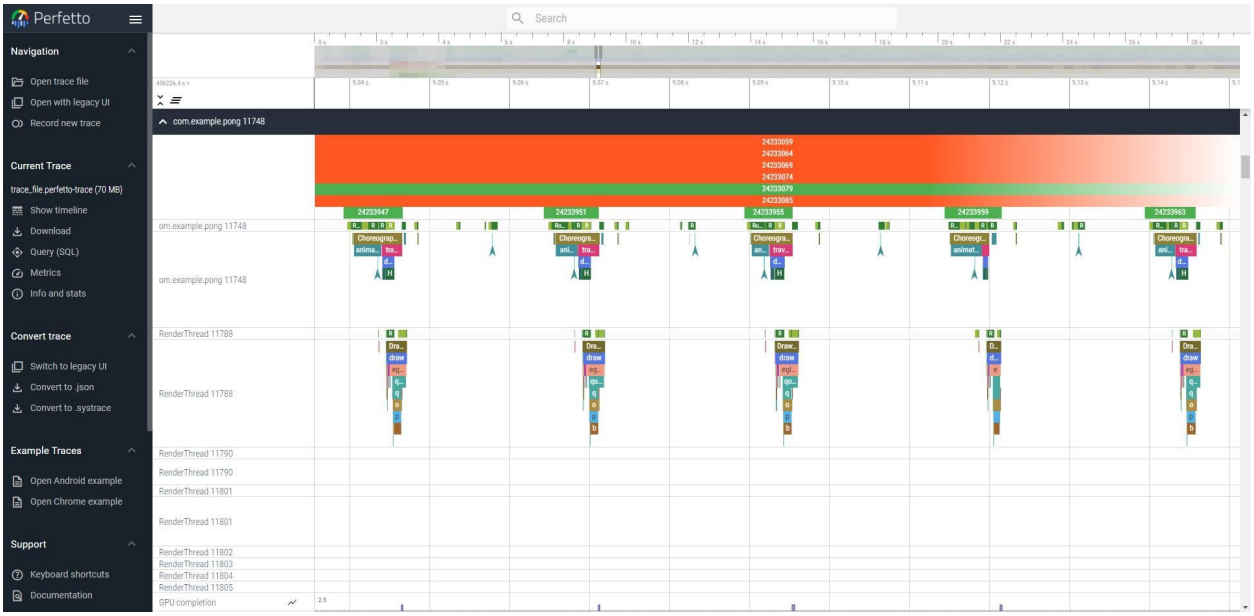
در توضیحات بخش حرکت تخته آورده شده که بر اساس اطلاعات شتاب سنج در سه محور اصلی، سرعت حرکتی تخته محاسبه می شود. همچنین زاویه تخته با توجه به سه زاویه در سه محور اصلی به دست می آید.

```
fun applyAcceleration(
    xAcceleration: Float,
    yAcceleration: Float,
    zAcceleration: Float,
    timestamp: Long
) {
    brick.applyAcceleration(xAcceleration, yAcceleration, zAcceleration, width, timestamp)
}
```

```
fun onRotation(values: FloatArray) {
    val floatPI = PI.toFloat()
    brick.applyAngle(values[0] * floatPI, values[1] * floatPI, values[2] * floatPI)
}
```

بخش پنجم) آزمایش ها، سناریوهای تست و سوالات

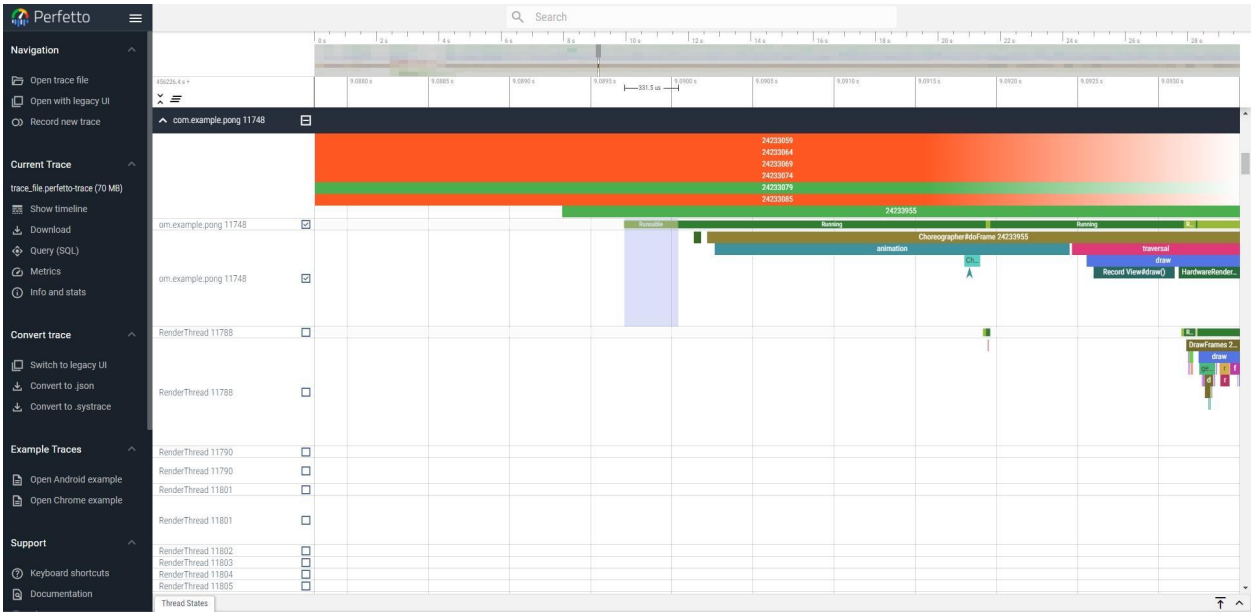
با دسترسی و تنظیمات نرم افزار perfetto، نتایج زیر حاصل شده است:



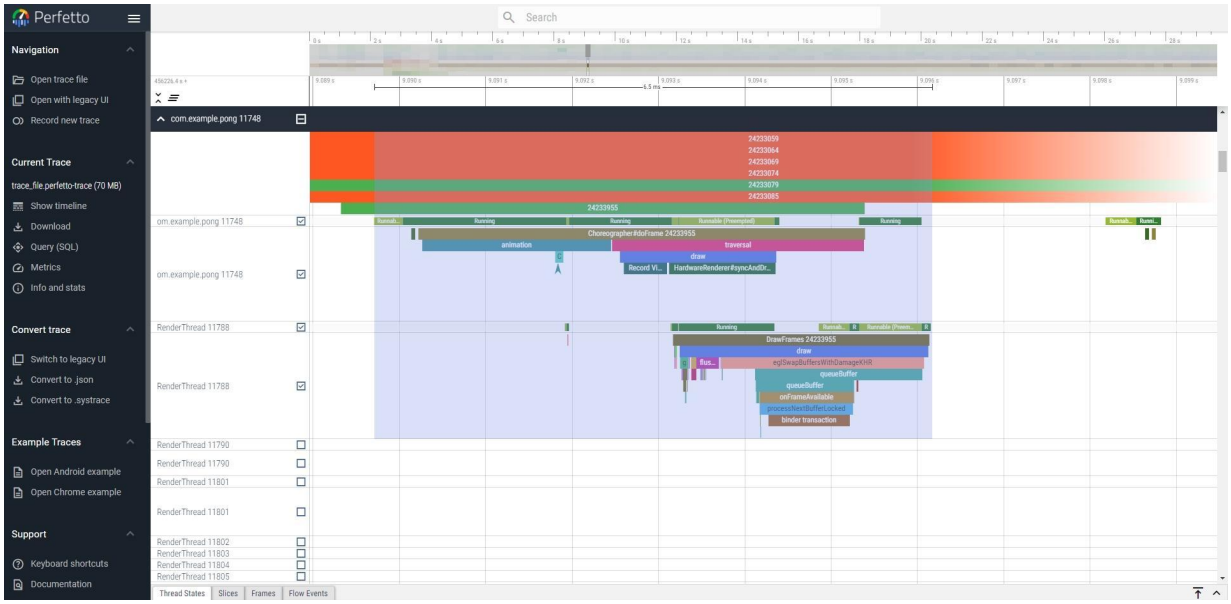
سوال اول:

- با توجه به عملیات Trace که تا انتها صورت گرفته، همانطور که در تصویر مشخص است، threat هایی برای اجرا شدن گرافیک و سنسور تشکیل شده است که در هر اجرا، زمانی برای عملیات به خود اختصاص داده است. سیستم عامل فقط به انجام گرافیک فعلی نمی پردازد و بین آنها زمان اجرا تقسیم میشود (بین سنسور و گرافیک). بدین صورت که ابتدا برنامه اجرا شده و انیمیشن را انجام میدهد و درخواست برای خواندن سنسور داده میشود، سنسور داده ها را اندازه گیری، ضبط و آماده کرده و سپس مجددا برنامه به عملیات خود ادامه میدهد.
- با توجه به بررسی یک فریم از اجرای برنامه و سنسور ها در تصویر دوم، در فراخوانی های سیستمی، تعارض می‌تواند بین استفاده از کتابخانه مربوط به گرافیک و بروزرسانی سنسورها وجود داشته باشد(حدود 300 میکرو ثانیه تاخیر دارد). دلیل این تعارض این است که هر دو فعالیت به منظور پاسخگویی به درخواست های مختلف از Thread های مختلف اجرا می‌شوند و برای انجام آن‌ها به منابع سخت‌افزاری مشترکی نیاز دارند. به عنوان مثال، برای رسم شکل های گرافیکی در صفحه نمایش، نیاز به استفاده از GPU

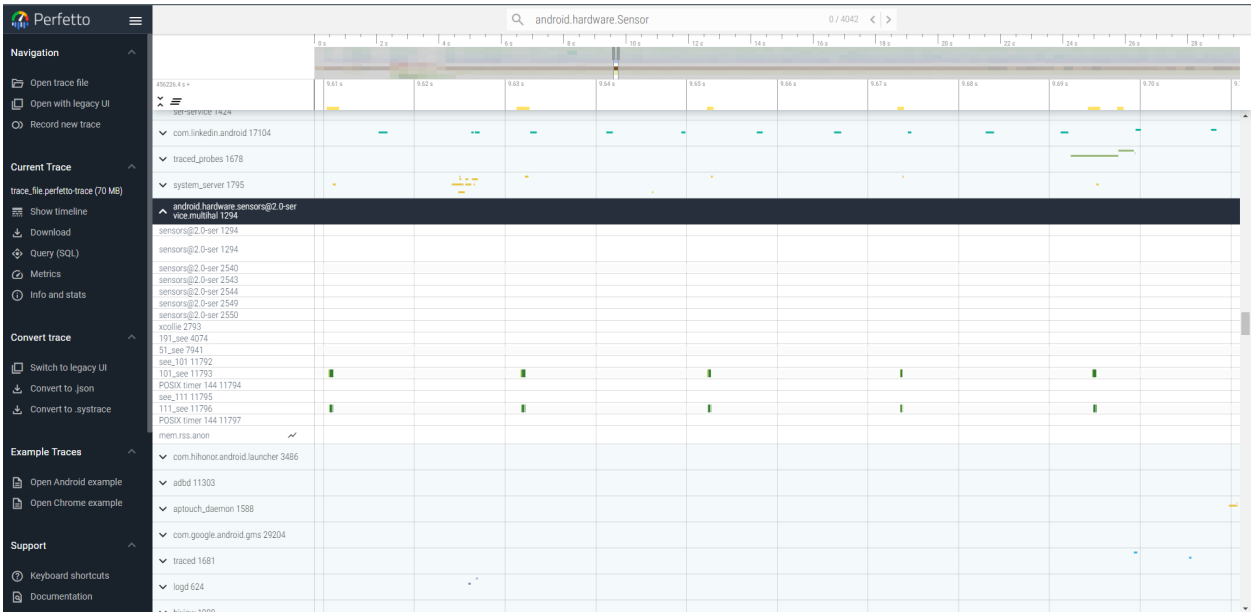
داریم که در عین حال سنسورهای مختلف نیز برای کارکرد صحیح خود به داده های حسگری ورودی نیاز دارند.



- طبق خروجی دریافت شده(تصویر دوم) زمانی که صرف سنسور ها شده کمتر از زمان عملیات های گرافیکی است و زمان صرف شده بیشتر برای گرافیک است.
- طبق عکس اول، زمان اجرای گرافیک بعد از اتمام کار سنسور کمتر از 6.5 میلی ثانیه خواهد بود.



همچنین با توجه به تصویر زیر و مطالعه سورس کد اندروید به پردازه مسئول داده های سنسور و پاسخگویی به callback ها را پیدا میکنیم . سپس با جست و جوی این پردازه در نتایج perfetto به تصویر زیر می رسیدم . همانطور که مشخص است به طور میانگین حدود هر 20 میلی ثانیه داده های سنسور ها دریافت شده و تحویل برنامه ما می شود .



سوال دوم: بهترین مقدار در برنامه ما، 6.5 میلی ثانیه ولی ما بیشتر از آن نیاز داریم. زمان کم می تواند منجر به frame drop شود و عملیات گرافیکی به صورت کامل اجرا نشود و تصویر لگ می افتد. اگر دقیقا 6.5 قرار دهیم، زمان های context switch و خواندن و ... را لحاظ نکرده ایم.

سوال سوم:

مزایا:

- عملکرد سریع‌تر: استفاده از NDK باعث می‌شود که قسمت‌هایی از کد که نیاز به عملکرد بالاتر و سرعت بیشتری دارند، به زبان C/C++ پیاده‌سازی شود. این باعث بهبود عملکرد و سرعت بازی می‌شود.
- قابلیت ایجاد کتابخانه‌های مجزا: اجزایی از کد که به طور مکرر استفاده می‌شوند، می‌توانند به صورت کتابخانه جداگانه پیاده‌سازی شده و در زمان نیاز مجدداً استفاده شوند.

- پشتیبانی از زبان C/C++: این زبان‌ها بهینه‌تر و دارای کارایی بالاتر هستند، به طوری که در مواردی از نظر سرعت و کارایی بسیار قابل توجه هستند.

معایب:

- کد پیچیده‌تر: استفاده از NDK باعث می‌شود کد پیچیده‌تری برای پیاده‌سازی بازی پینگ پونگ نوشته شود.
- پایداری کمتر: برای پیاده‌سازی بازی با استفاده از NDK، باید به صورت دقیق با حافظه و ریسمان‌ها کار کرد. اگر کد به درستی پیاده‌سازی نشود، ممکن است بازی پایداری کمتری داشته باشد.

سوال چهارم:

سنسورهای based-hardware و based-software دو نوع سنسور می‌باشند که هر یک با روش‌های مختلفی عمل می‌کنند:

1. سنسورهای based-hardware:

این نوع سنسورها هستند که توسط قطعات الکترونیکی سخت‌افزاری پشتیبانی می‌شوند. به عبارت دیگر، این سنسورها از روی اصول فیزیکی عمل می‌کنند و به صورت مستقیم به سیستم فیزیکی متصل می‌شوند. برخی از نمونه‌های سنسورهای based-hardware شامل سنسورهای حرارتی، سنسورهای فشار، سنسورهای رطوبت، سنسورهای شتاب‌سنج، سنسورهای ژئومغناطیسی و ... می‌باشند.

2. سنسورهای based-software:

این نوع از سنسورها به صورت کاملا نرم‌افزاری کار می‌کنند و از طریق الگوریتم‌های پیچیده‌ای که در نرم‌افزار پیاده‌سازی شده‌اند، اطلاعات موردنیاز جمع‌آوری می‌شود. یکی از مزیت‌های سنسورهای based-software، این است که بدون نیاز به قطعات الکترونیکی خاصی و تنظیمات سخت‌افزاری می‌توانند عمل کنند. از دیگر نمونه‌های سنسورهای based-software می‌توان به سنسورهای شنیداری (مانند سیستم‌های تشخیص صدا)، سنسورهای تصویری (مانند دوربین‌های حساس به حرکت) و ... اشاره کرد.

با توجه به فرق موجود در روش کاری دو نوع سنسور، هر کدام می‌توانند برای کاربردهای خاصی مناسب باشند. برای مثال، سنسورهای based-hardware برای اندازه‌گیری دقیق مقادیر فیزیکی مانند دما، فشار و شتاب مناسب هستند. از طرف دیگر، سنسورهای based-software برای تشخیص الگوها و نقشه‌برداری (mapping) مناسب هستند.

سنسور TYPE_ROTATION_VECTOR و TYPE_LINEAR_ACCELERATION در دستگاه هایی با حسگر شتاب سنج (Accelerometer)، ژیروسکوپ (Gyroscope) و مغناطیس سنج (Magnetometer) موجود هستند. این حسگرها برای اندازه گیری شتاب خطی، سرعت زاویه ای و جهت قطب نما استفاده می شوند.

سوال پنجم:

تعریف سنسور به صورت up-wake و up-wake-non بسته به نوع کاربری آن تعریف می‌شود. این دو نوع تعریف به شرح زیر هستند:

1. Up-wake sensor:

سنسور up-wake، یک نوع سنسور حسگر است که بر روی دستگاه قرار داده می‌شود و در هنگام بالا آمدن دستگاه (up) فعال می‌شود و از حالت خواب خارج می‌شود (wake). این نوع سنسور برای دستگاه‌هایی که باید به صورت مداوم کار کنند، مثل دستگاه‌های پزشکی، مانیتورهای پروژه، رایانه‌های شخصی و ... مناسب است.

مزایا:

- این نوع سنسور برای دستگاه‌های با مصرف انرژی بالا مناسب است.

- به دلیل عدم نیاز به فعال‌سازی دستی، این سنسور در کنترل مصرف انرژی و زمان به‌روزرسانی داده‌ها مفید است.

- این سنسور تاخیر در ارائه داده‌ها را به حداقل می‌رساند.

معایب:

- این سنسور ممکن است با نویز و شوش‌های محیطی مختلف، مانند لرزش جسم، تداخل کند.

- چون همیشه فعال و در حال بررسی وضعیت دستگاه است، ممکن است باعث بیشتر شدن مصرف انرژی شود.

2. Up-wake-non sensor:

سنسور up-wake-non کاملاً مشابه سنسور up-wake است با این تفاوت که در هنگام بالا آمدن دستگاه (up) فقط یک پیغام به دستگاه فرستاده می‌شود که باید خودش را فعال کند و سنسور به صورت خودکار فعال نمی‌شود. این نوع سنسور برای دستگاه‌هایی که مصرف انرژی کمتری دارند، مثل موبایل‌ها و تبلت‌ها مناسب است.

مزایا:

- مصرف انرژی کمتری نسبت به سنسور up-wake دارد.
- عدم فعال-سازی سنسور به صورت خودکار باعث کاهش تداخلات محیطی و بهبود دقت پاسخ سنسور می‌شود.

معایب:

- برای استفاده از دستگاه، کاربر باید به صورت دستی آن را فعال کند که ممکن است زمان‌بر باشد.
- چون سنسور فقط در هنگام بالا آمدن دستگاه فعال می‌شود، منجر به تاخیر در ارائه داده‌ها می‌شود.

تفاوت اصلی بین سنسورهای up-wake و up-wake-non در نحوه فعال شدن آن‌ها است. در سنسور up-wake، وقتی دستگاه بالا می‌آید (up)، سنسور به طور خودکار فعال می‌شود و اطلاعات بروزرسانی شده را به دستگاه ارسال می‌کند. این بدان معناست که هر زمان دستگاه فعال شود، سنسور بلافاصله بروزرسانی‌های لازم را انجام می‌دهد.

در مقابل، در سنسور up-wake-non، وقتی دستگاه بالا می‌آید، سنسور فقط یک پیغام به دستگاه ارسال می‌کند تا دستگاه خودش را فعال کند. در این حالت، سنسور به صورت خودکار فعال نمی‌شود و بروزرسانی‌های لازم تا زمانی که کاربر دستگاه را فعال کند انجام نخواهد شد.

بنابراین، تفاوت اصلی بین دو نوع سنسور، زمانی است که اطلاعات بروزرسانی شده توسط سنسور به دستگاه فرستاده می‌شود. در سنسور up-wake، هر زمان دستگاه فعال شود، سنسور بلافاصله بروزرسانی‌های لازم را انجام می‌دهد. در حالی که در سنسور up-wake-non، بروزرسانی‌ها تا زمانی که کاربر دستگاه را فعال نکرده باشد، انجام نخواهد شد.

بخش ششم) سکوی نرم افزاری و سخت افرازی در پیاده سازی

شناسه	سکوی نرم افزار	سکوی سخت افزار
1	اندروید 12	Honor 70
2	اندروید 12	Huawei nova 9
3	اندروید 11	Xiaomi A3
4	اندروید 12	Xiaomi note 11
5	اندروید 10	Nokia 6.1

بخش هفتم) مراجع مورد استفاده

- https://developer.android.com/jetpack/compose/graphics/draw/overview
- https://semicolonspace.com/draw-circle-jetpack-compose-canvas
- https://developer.android.com/reference/kotlin/android/hardware/SensorManager
- https://stackoverflow.com/questions/75543273/acceleration-gyroscope-sensor-data-stopped-working