Programando o IMU

O SDK 8.1 oferece novas classes e métodos que se aplicam universalmente a ambos os tipos de IMU. Uma vez configurado, o tipo de IMU não afetará sua programação. Os passos para programação incluem:

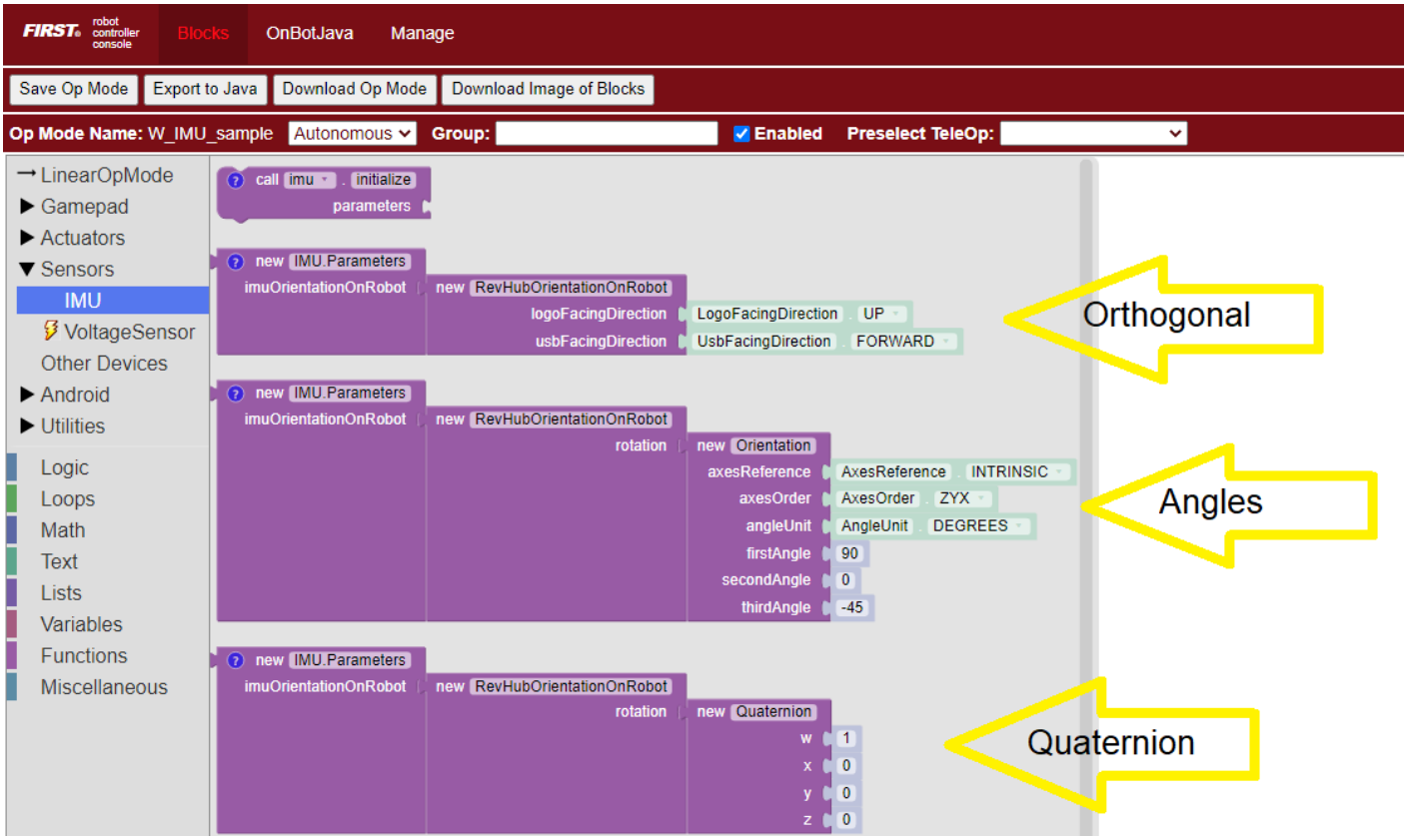
- Configurar os parâmetros do IMU ou usar os padrões.
- Inicializar o IMU.
- Ler os valores do IMU e usá-los conforme necessário para controlar o robô. Opcional: redefinir o Heading uma ou mais vezes.

As seções a seguir cobrem esses tópicos em ordem.

Parâmetros

Existem três maneiras de descrever a orientação do Hub usando os parâmetros do IMU. Uma é para montagem ortogonal, e as outras duas são para montagem não ortogonal. Escolha o método mais simples que se aplica ao seu robô.

Como exemplo, no menu Blocos do FIRST Tech Challenge, em Sensores e IMU, você pode encontrar esses três métodos para especificar parâmetros:



Parâmetros para método 1, ortogonal

O Método 1 consiste em fornecer uma configuração ortogonal simples. Isso exige determinar a direção para a qual o logotipo da REV está voltado. Para fazer isso, considere que o Hub está montado em um cubo imaginário alinhado com a "frente" do robô. Especifique a face de montagem do Hub: "Forward" significa frente do robô (face frontal), "Left" significa lado esquerdo do robô, e assim por diante.

Em seguida, escolha como o Hub está girado nessa face. Use as portas USB na “parte superior” do Hub para determinar essa direção; suponha que você está na parte traseira do robô, olhando para a “frente”.

> Certas combinações são fisicamente impossíveis. Por exemplo, se o logotipo da REV estiver voltado para CIMA, as portas USB não podem estar também voltadas para CIMA. O OpMode rejeitará essas combinações durante a inicialização do IMU.

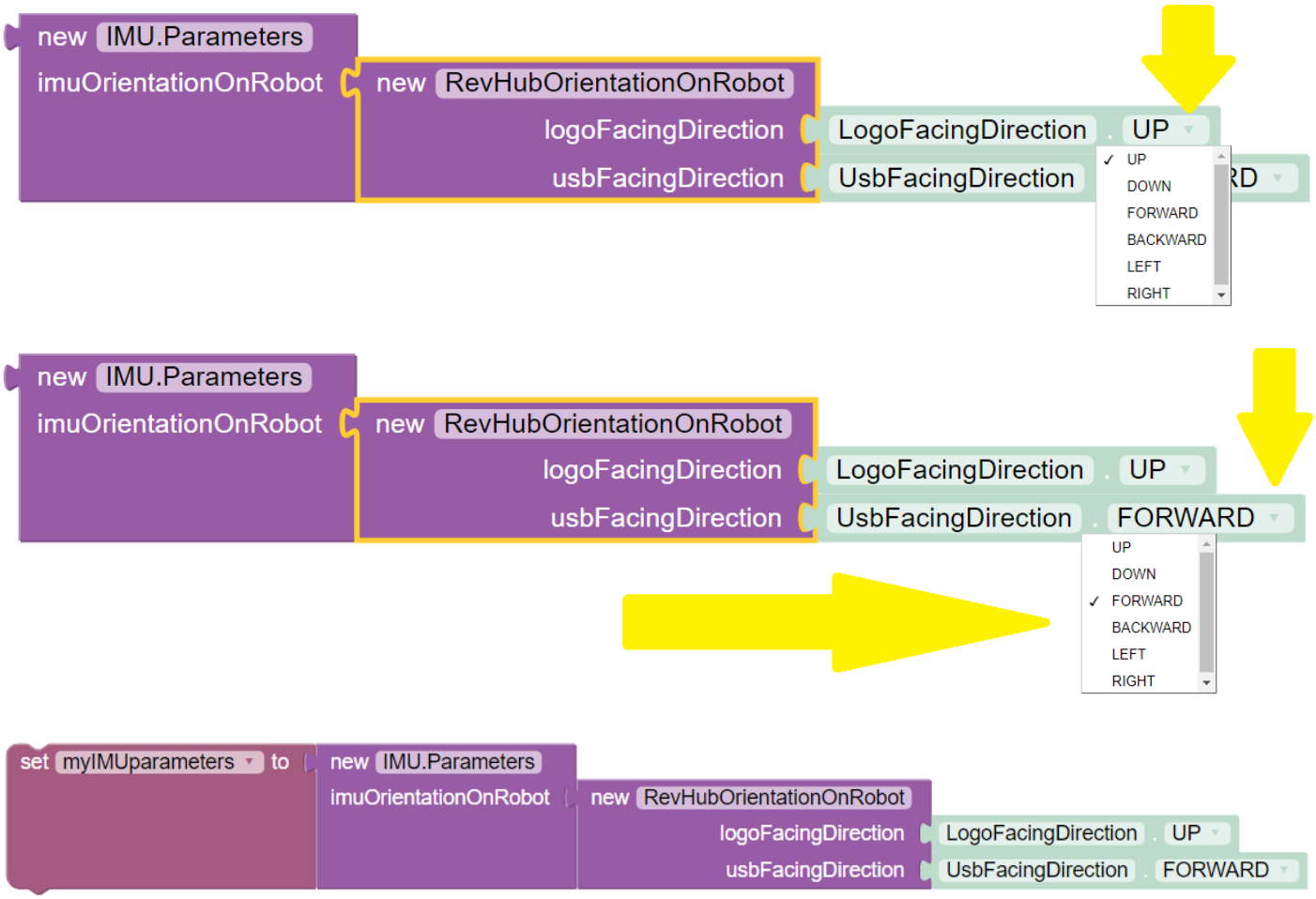
É opcional salvar os parâmetros em uma nova variável chamada, por exemplo, "meusParametrosIMU". Essa variável pode ser usada na próxima etapa (inicialização do IMU).

Java

```
IMU.Parameters myIMUparameters;
```

```
myIMUparameters = new IMU.Parameters(
     new RevHubOrientationOnRobot(
          RevHubOrientationOnRobot.LogoFacingDirection.UP,
          RevHubOrientationOnRobot.UsbFacingDirection.FORWARD
     )
);
```

Blocks



Eixos do Hub para definir parâmetros

Apenas para as próximas duas seções de Parâmetros (Ângulos e Quaternion), devemos usar temporariamente os eixos do Hub em vez dos eixos do Robô. Os eixos do Hub também estão a 90 graus entre si, com a origem dentro do próprio Hub.

A posição inicial assumida do Hub é com o logotipo da REV voltado para CIMA (Eixo +Z do Robô) e as portas USB voltadas para a FRENTE (Eixo +Y do Robô). Para os métodos de Ângulos e Quaternion, todas as rotações começam a partir desta posição.

Novamente, o termo "frente" é baseado na definição acordada pela sua equipe.

Nesta orientação inicial, os eixos do Hub estão alinhados com o Sistema de Coordenadas do Robô:

- Heading, ou Yaw, é a medida da rotação em torno do eixo Z, que aponta para cima através da placa frontal ou logotipo do Hub.
- Pitch é a medida da rotação em torno do eixo X, que aponta na direção das portas do sensor I2C no lado direito.
- Roll é a medida da rotação em torno do eixo Y, que aponta na direção das portas USB localizadas na borda superior.

As rotações do Hub também seguem a regra da mão direita.

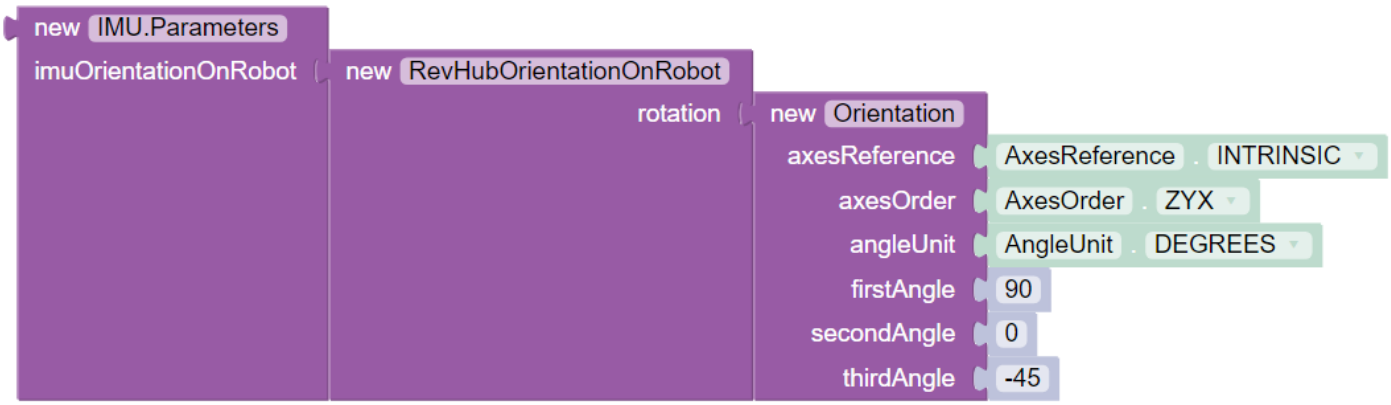
O driver legado BNO055IMU utilizava eixos diferentes para o Hub: o eixo X apontava para a porta USB, e o eixo Y apontava para as portas dos motores no lado esquerdo.

O novo driver universal IMU do SDK 8.1 usa os eixos do Hub descritos anteriormente para os sensores BNO055 e BHI260AP.

Parâmetros para método 2, ângulos

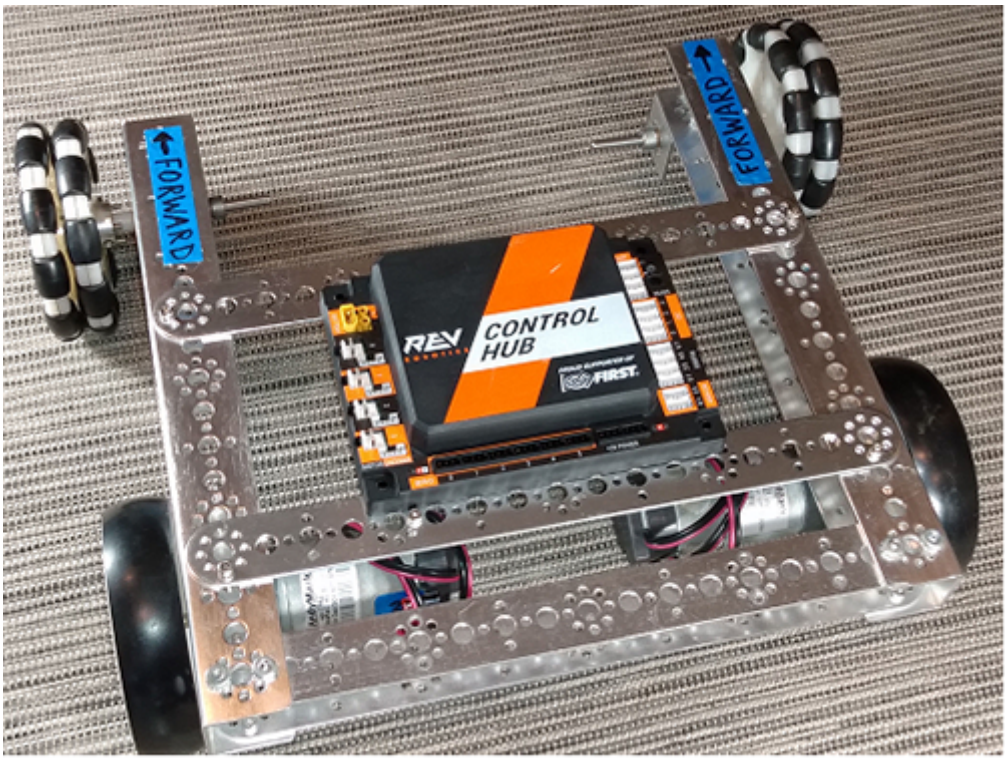
Se o seu Hub não estiver montado de forma ortogonal, você pode especificar a rotação do Hub sobre um ou mais eixos do Hub (X, Y, Z). Essas rotações são expressas em graus, e a ordem em que as rotações são aplicadas importa!

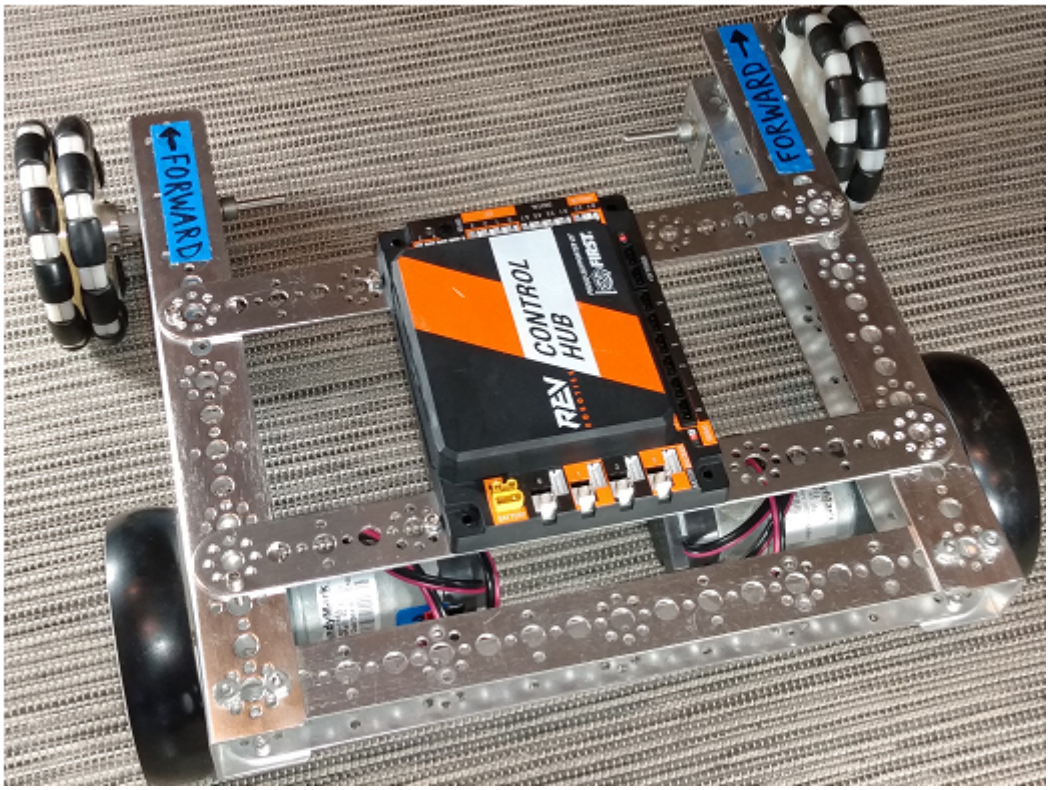
A paleta de Blocos do IMU contém um bloco com parâmetros padrão para o método de Ângulos de descrever a orientação do Hub no robô. Vamos revisar essa função da paleta de Blocos agora, como um bom exemplo. A API Java se assemelha bastante ao método de Blocos.

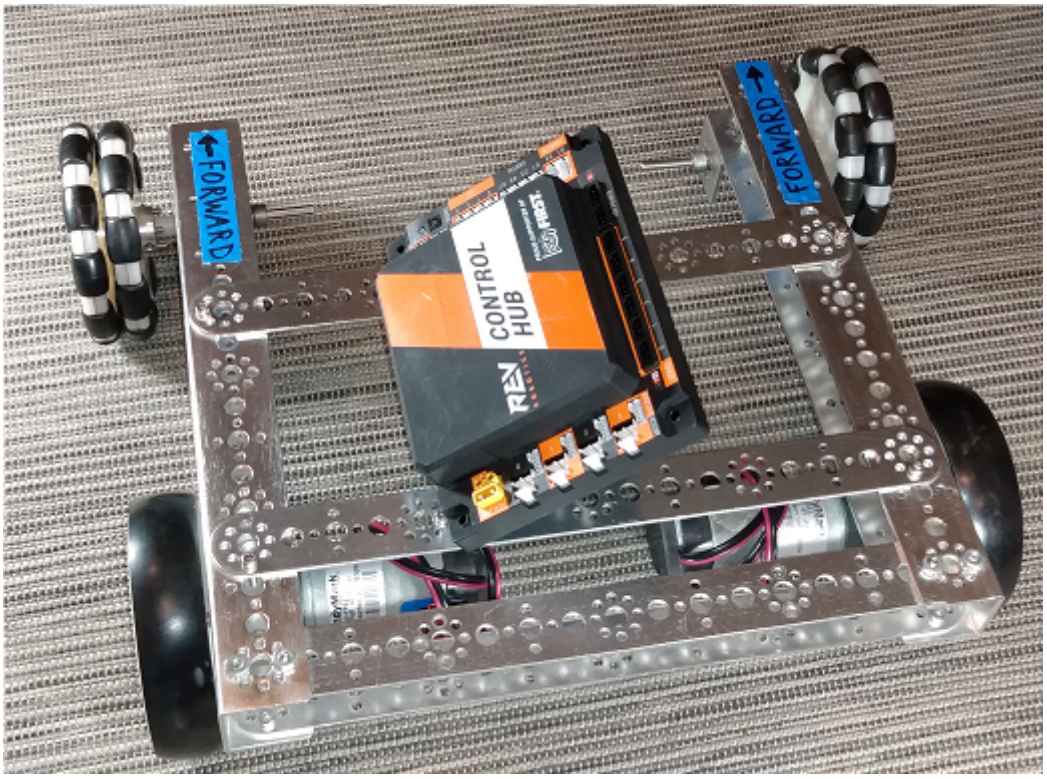


1. O segundo valor padrão listado é ZYX, o que significa que você fornecerá as rotações do Hub nessa ordem. Assim, o "primeiro ângulo" é sobre o eixo Z, o "segundo ângulo" é sobre o eixo Y, e o "terceiro ângulo" é sobre o eixo X.
2. Portanto, o Hub será girado da seguinte maneira: +90 graus sobre o eixo Z, sem rotação sobre o eixo Y, e depois -45 graus sobre o eixo X (na sua nova direção).
3. Para o método de Ângulos, a posição inicial assumida do Hub é com o logotipo da REV voltado para CIMA, e as portas USB voltadas para a FRENTE. As rotações adicionais começam a partir dessa orientação.

Aqui está a sequência completa:







Os parâmetros padrão restantes não precisam de atenção ou edição. O terceiro parâmetro listado é simplesmente GRAUS, que é fácil de trabalhar. O primeiro parâmetro listado é a referência dos eixos INTRÍNSECOS, o que significa que os eixos do Hub se movem com cada rotação do Hub. (A outra opção, raramente utilizada, é EXTRÍNSECO, para eixos globais que não se movem com cada rotação do Hub.)

Assim como no método ortogonal, é opcional salvar os parâmetros em uma nova variável chamada, por exemplo, "meusParametrosIMU". Essa variável pode ser usada na próxima etapa (inicialização do IMU).

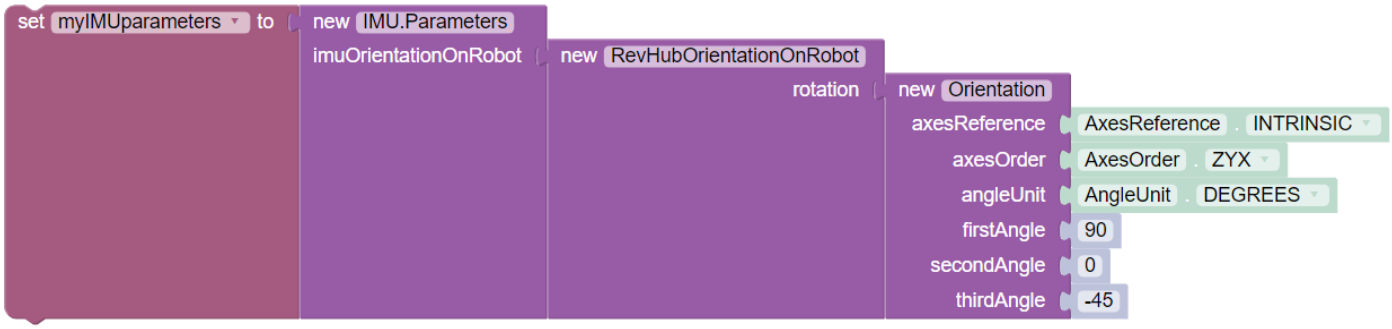
Java

```
IMU.Parameters myIMUparameters;


myIMUparameters = new IMU.Parameters(
     new RevHubOrientationOnRobot(
          new Orientation(
               AxesReference.INTRINSIC,
               AxesOrder.ZYX,
               AngleUnit.DEGREES,
               90,
               0,
               -45,
               0  // acquisitionTime, not used
```

```
          )
     )
);
```

Blocks



Parâmetros para método 3, Quaternion

Como alternativa ao método de Ângulos, a orientação não ortogonal do Hub pode ser descrita usando um Quaternion, uma técnica matemática avançada para descrever qualquer combinação de inclinação e rotação.

O Quaternion padrão (w=1, x=0, y=0, z=0) descreve um Hub na posição inicial assumida: logotipo voltado para CIMA e portas USB voltadas para a FRENTE. Ou seja, sem rotações.

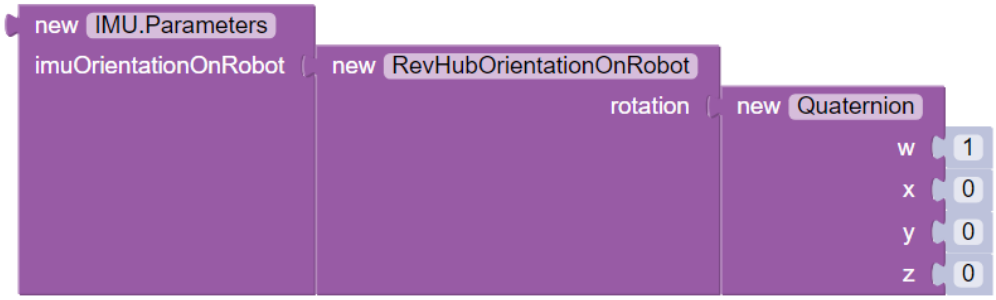
Java

```
IMU.Parameters myIMUparameters;

// Default Quaternion
myIMUparameters = new IMU.Parameters(
     new RevHubOrientationOnRobot(
          new Quaternion(
               1.0f, // w
               0.0f, // x
               0.0f, // y
               0.0f, // z
               0     // acquisitionTime
          )
     )
);
```

```
// Or, consider a single rotation of +30 degrees
// about the X axis. Namely, the Hub’s USB ports
// tilt 30 degrees upwards from the default starting
// position.
myIMUparameters = new IMU.Parameters(
     new RevHubOrientationOnRobot(
          new Quaternion(
               0.9659258f, // w
               0.258819f,  // x
               0.0f,       // y
               0.0f,       // z
               0           // acquisitionTime
          )
     )
);
```

Blocks



Este tutorial básico não cobre a matemática por trás dos Quaternions, que são um substituto avançado para os Ângulos de Euler descritos acima. A interface IMU do SDK 8.1 dá suporte ao uso de Quaternions, para equipes e bibliotecas de terceiros que estão familiarizadas com essa técnica.

Inicializando o IMU

Isso prepara o IMU para operação, usando os parâmetros que você definiu.

No Blocos, use o primeiro bloco mostrado na paleta IMU, chamado imu.initialize. A maioria das equipes faz isso durante a fase INIT do seu OpMode, antes de waitForStart().

O IMU deve estar imóvel durante o processo de inicialização. O OpMode continuará quando a inicialização estiver completa.

> Fato curioso: Sob a interface legada BNO055IMU, a inicialização leva cerca de 900 milissegundos. Já sob a nova interface universal IMU, o BNO055 leva cerca de 100

> milissegundos, enquanto o BHI260AP leva cerca de 50 milissegundos.

Para qualquer um dos três métodos (Ortogonal, Ângulos, Quaternion), inicialize com os parâmetros IMU do novo bloco ou da sua variável opcional.

Java

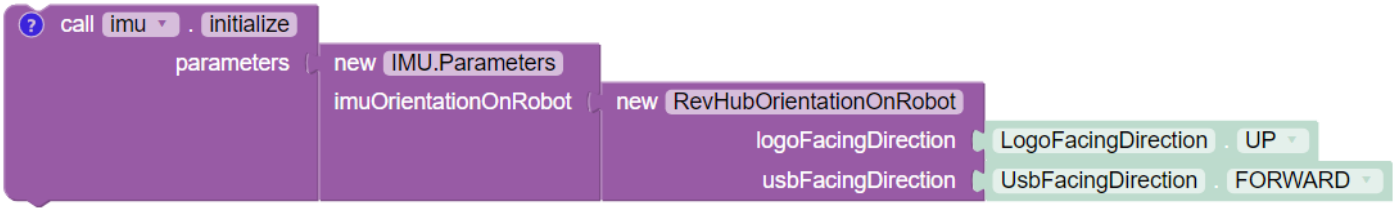
```
// Two methods for Initializing the IMU:

// Initialize IMU directly
imu.initialize(
     new IMU.Parameters(
          new RevHubOrientationOnRobot(
               RevHubOrientationOnRobot.LogoFacingDirection.UP,
               RevHubOrientationOnRobot.UsbFacingDirection.FORWARD
          )
     )
);

// Initialize IMU using Parameters
imu.initialize(myIMUparameters);
```

Blocks

Inicializar IMU diretamente:



Inicializar IMU utilizando parâmetros:



Leitura dos ângulos do IMU - Básico

Agora você pode ler os valores do IMU para a orientação do robô, expressos como Heading (Yaw ou ângulo Z), Pitch (ângulo X) e Roll (ângulo Y). Não é mais necessário se preocupar com a orientação ou montagem do Hub — isso foi definido com os parâmetros, e o SDK está pronto para fornecer dados reais sobre o robô, usando os eixos do robô.

> Lembrete: Z do robô aponta para cima, em direção ao teto. Y do robô aponta para frente – o que quer que você defina como "frente" no seu robô (que pode ser redondo!). X do robô aponta para o lado direito do robô. As rotações do robô seguem a regra da mão direita.

Para todos os eixos, os ângulos do IMU são fornecidos no intervalo de -180 a +180 graus (ou de -Pi a +Pi radianos). Se você estiver trabalhando com valores que possam cruzar a transição de +/- 180 graus, trate isso em seu código. Isso está além do escopo deste tutorial do IMU.

Aqui está um exemplo de leitura dos Ângulos do IMU:

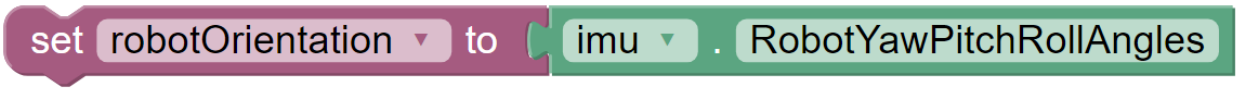
Java

```
// Create an object to receive the IMU angles
YawPitchRollAngles robotOrientation;
robotOrientation = imu.getRobotYawPitchRollAngles();

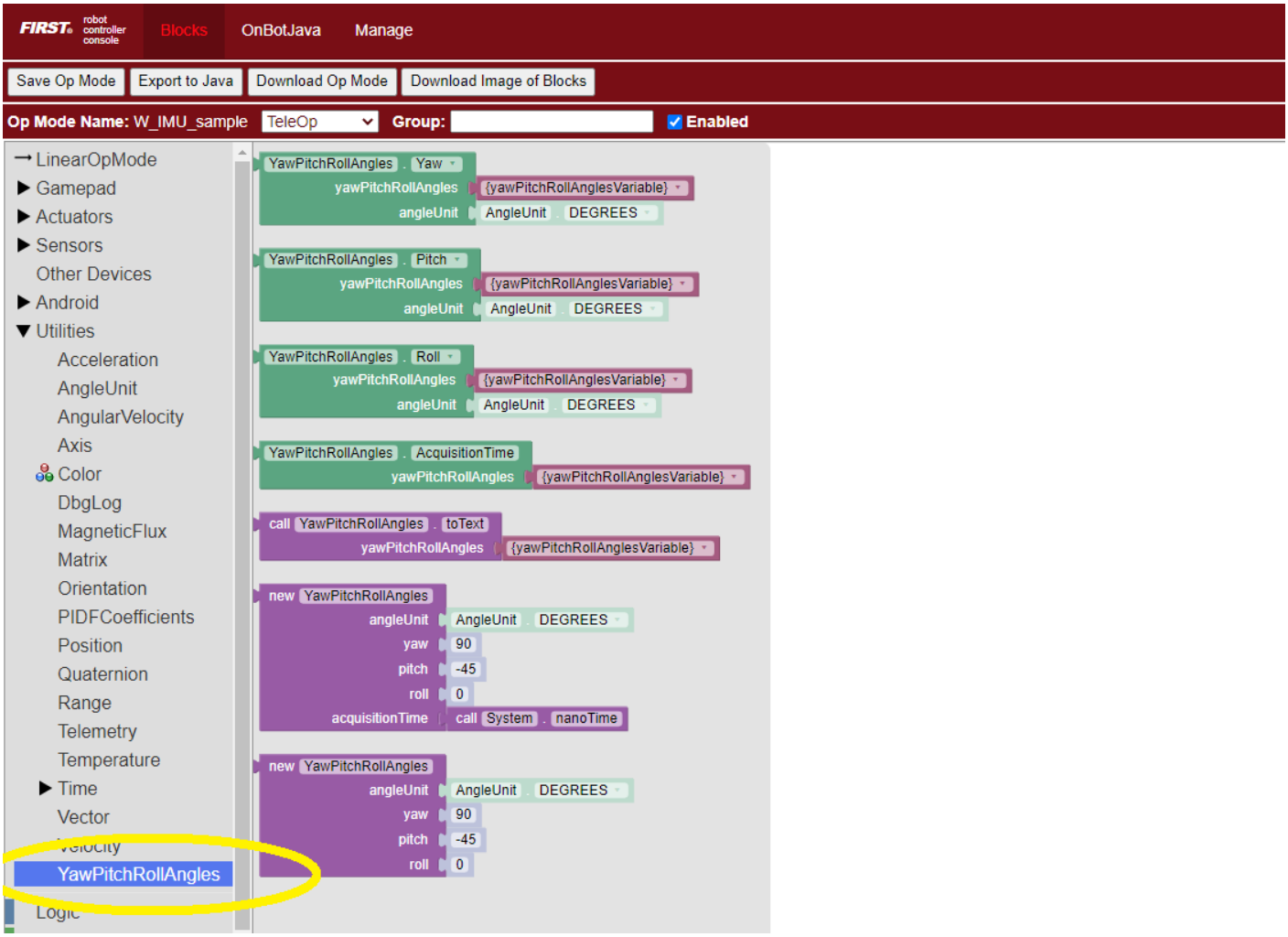
// Now use these simple methods to extract each angle
// (Java type double) from the object you just created:
double Yaw   = robotOrientation.getYaw(AngleUnit.DEGREES);
double Pitch = robotOrientation.getPitch(AngleUnit.DEGREES);
double Roll  = robotOrientation.getRoll(AngleUnit.DEGREES);
```

Blocks

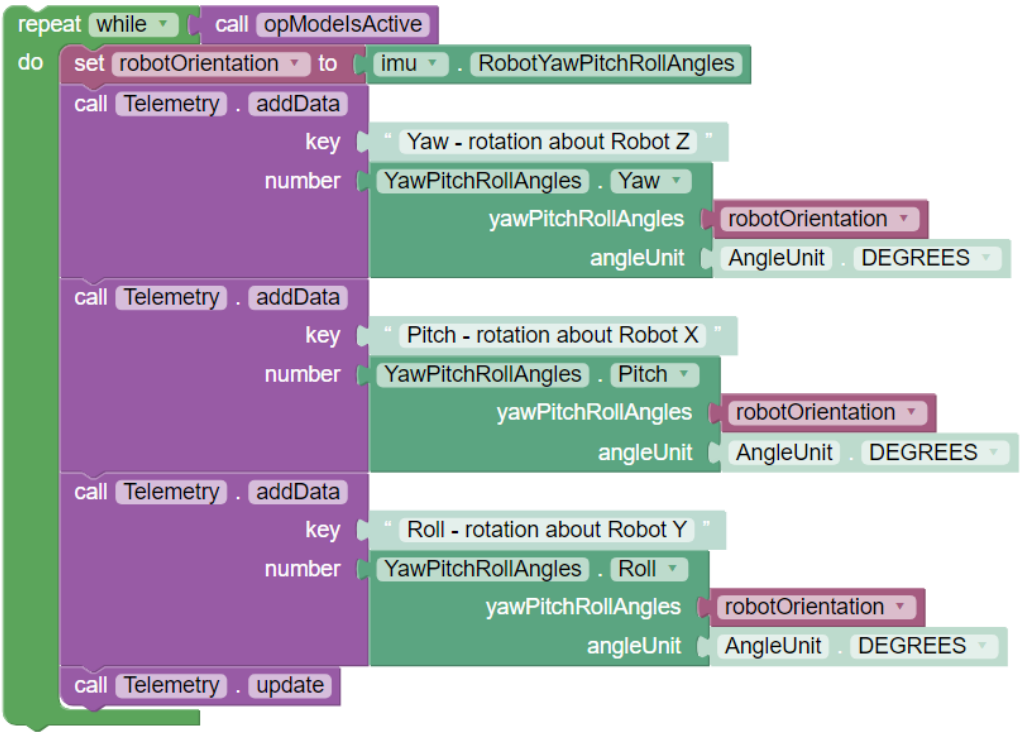
No Blocos, crie uma nova variável para receber dados deste bloco verde na paleta IMU:



Na paleta YawPitchRollAngles em Utilities, use os blocos verdes para ler cada ângulo da variável que você acabou de criar.



Esses blocos são usados aqui em um Loop de Repetição para exibir os ângulos na Driver Station:



Esses blocos são mostrados no OpMode de exemplo chamado SensorIMU.

Observe que a orientação do robô é descrita aqui de forma intrínseca; os eixos se movem com cada rotação. Aqui está um exemplo da Javadocs:

"Como exemplo, se o yaw for 30 graus, o pitch for 40 graus e o roll for 10 graus, isso significa que você atingiria a orientação descrita da seguinte forma:

- Primeiro, rotaciona-se o robô 30 graus no sentido anti-horário a partir do ponto de partida, com todas as rodas continuando a tocar o chão (rotação ao redor do eixo Z).
- Em seguida, você faz o robô apontar 40 graus para cima (girando-o 40 graus ao redor do eixo X). Como o eixo X se move com o robô, o pitch não é afetado pelo valor do yaw.
- A partir dessa posição, o robô é inclinado 10 graus para a direita, ao redor do novo eixo Y, para produzir a posição final do robô."

Novamente, os resultados de saída do IMU são dados no Sistema de Coordenadas do Robô ou eixos do robô. Somente para uma orientação não ortogonal, os eixos do Hub foram usados temporariamente para os parâmetros de entrada, descrevendo a rotação do Hub para alcançar sua orientação montada.

Leitura dos ângulos do IMU - Flexível

Como alternativa à classe YawPitchRollAngles, o SDK também oferece uma classe mais flexível chamada Orientation. Ela permite que você especifique uma ordem personalizada de rotações dos eixos, além da escolha entre eixos intrínsecos ou extrínsecos.

Novamente, os ângulos do IMU são fornecidos no intervalo de -180 a +180 graus (ou de -Pi a +Pi radianos).

Aqui está um exemplo de uso dessas funções:

Java

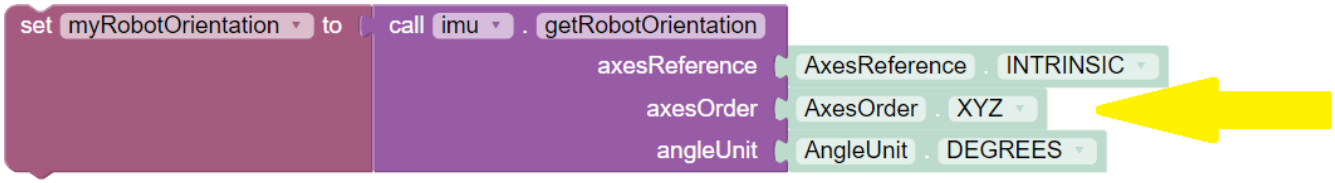
```
// Create Orientation variable
Orientation myRobotOrientation;

// Get Robot Orientation
myRobotOrientation = imu.getRobotOrientation(
     AxesReference.INTRINSIC,
     AxesOrder.XYZ,
     AngleUnit.DEGREES
);
```

```
// Then read or display the desired values (Java type float):
float X_axis = myRobotOrientation.firstAngle;
float Y_axis = myRobotOrientation.secondAngle;
float Z_axis = myRobotOrientation.thirdAngle;
```

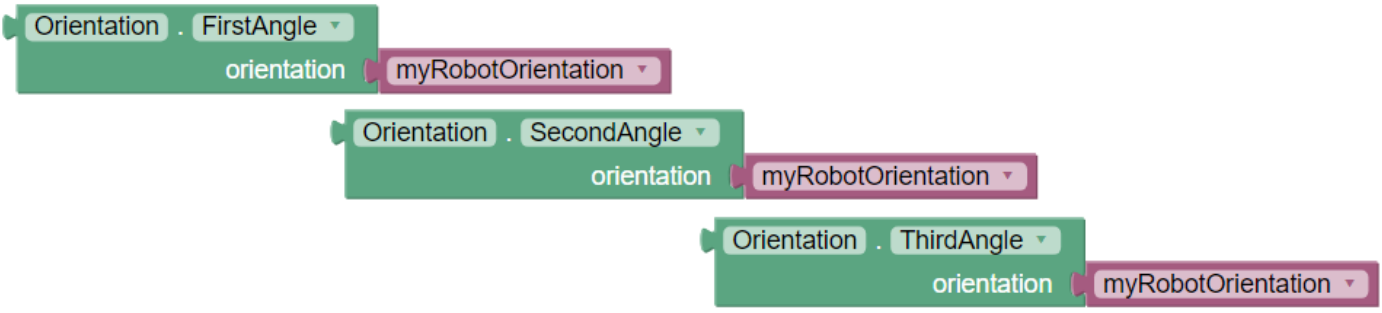
Blocks

Como antes, primeiro crie um objeto (variável de Blocos) contendo o array de valores de orientação (da paleta Sensores / IMU de Blocos):



Note a ordem dos eixos XYZ, diferente da ordem ZXY usada pela classe YawPitchRollAngles.

Em seguida, extraia as rotações dos eixos específicas que você deseja, da paleta Utilities / Orientation de Blocos:



> Preste muita atenção à seleção da ordem dos eixos, pois isso afeta significativamente os resultados do IMU. Se o que mais importa para você é o Heading (Yaw), escolha uma ordem de eixos que comece com Z.

Leitura da velocidade angular

O SDK também fornece valores para a velocidade angular, que é a taxa de variação (em graus ou radianos por segundo) para Roll, Pitch ou Yaw.

Aqui está um exemplo de como ler a Velocidade Angular:

Java

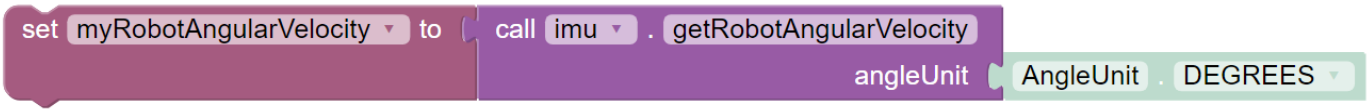
```
// Create angular velocity array variable
AngularVelocity myRobotAngularVelocity;

// Read Angular Velocities
myRobotAngularVelocity = imu.getRobotAngularVelocity(AngleUnit.DEGREES);

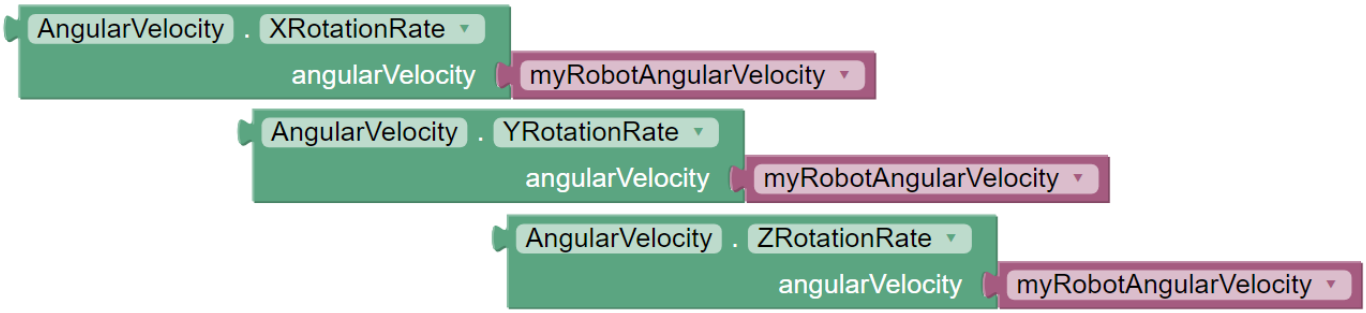
// Then read or display these values (Java type float)
// from the object you just created:
float zRotationRate = myRobotAngularVelocity.zRotationRate;
float xRotationRate = myRobotAngularVelocity.xRotationRate;
float yRotationRate = myRobotAngularVelocity.yRotationRate;
```

Blocks

Como antes, primeiro crie um objeto (Blocks Variable) contendo o array de valores de velocidade angular (do paleta de Blocos Sensors / IMU):



Em seguida, extraia as rotações específicas dos eixos que você deseja, a partir da paleta de Blocos Utilities / AngularVelocity:



Resetar orientação

Pode ser útil redefinir o Heading (ou Yaw ou ângulo Z) para zero em um ou mais pontos do seu OpMode.

Aqui está um exemplo de como redefinir o eixo Yaw:

Java

```
// Reset Yaw
imu.resetYaw();
```

Blocks



É mais seguro redefinir o Yaw apenas quando o robô não tiver se desviado significativamente de uma orientação plana/horizontal.

Este comando assume que a orientação real do Hub foi corretamente descrita com os parâmetros Ortogonais, Ângulos ou Quaternion.

Em outras palavras, um Hub não ortogonal que se afastou de sua orientação definida pelos parâmetros pode não fornecer resultados confiáveis para Heading/Yaw ou resetYaw(), mesmo após o robô retornar à sua orientação original definida.

Uma exceção, ou brecha, é que os valores de Heading/Yaw "redefinidos" podem ainda ser válidos se o Hub estiver montado em uma orientação Ortogonal descrita incorretamente e o robô permanecer nivelado. Isso pode beneficiar uma equipe iniciante que tenha ignorado os Parâmetros da IMU ou movido o Hub para uma posição Ortogonal diferente, ainda confiando apenas no Heading. Essa exceção do resetYaw() não se aplica à velocidade angular do Yaw (eixo Z).

Aqui está a descrição oficial do Javadoc para o método resetYaw():

Redefine o ângulo de yaw do robô para 0. Após chamar este método, a orientação reportada será relativa à posição do robô no momento em que este método for chamado, como se o robô estivesse perfeitamente nivelado naquele momento. Ou seja, o pitch e o yaw serão ignorados quando este método for chamado.

A declaração dos Javadocs "redefine para 0" deve ser interpretada no contexto da discussão anterior. Em certas orientações do Hub fora do eixo, um valor de Yaw redefinido pode não ser exibido como zero de fato.

Se o resetYaw() não atender às suas necessidades, outras opções baseadas em código (possivelmente menos eficazes) incluem:

- Salvar e Subtrair para estabelecer o Heading atual como uma nova linha de base "zero" para navegação futura.
- Usar o Heading original durante toda a partida, utilizando apenas alvos absolutos (globais).

>

> Para todas as opções, esteja ciente do "desvio do giroscópio". A maioria dos IMUseletrônicos fornece resultados de ângulo Z que mudam lentamente com o tempo, por vários motivos. Embora os eixos Pitch e Roll possam usar a direção da gravidade para corrigir o desvio, o Yaw (Heading ou ângulo Z) não pode.

Exemplos de OpModes

O SDK 8.1 e versões mais recentes contêm OpModes de exemplo demonstrando o que foi dito acima.

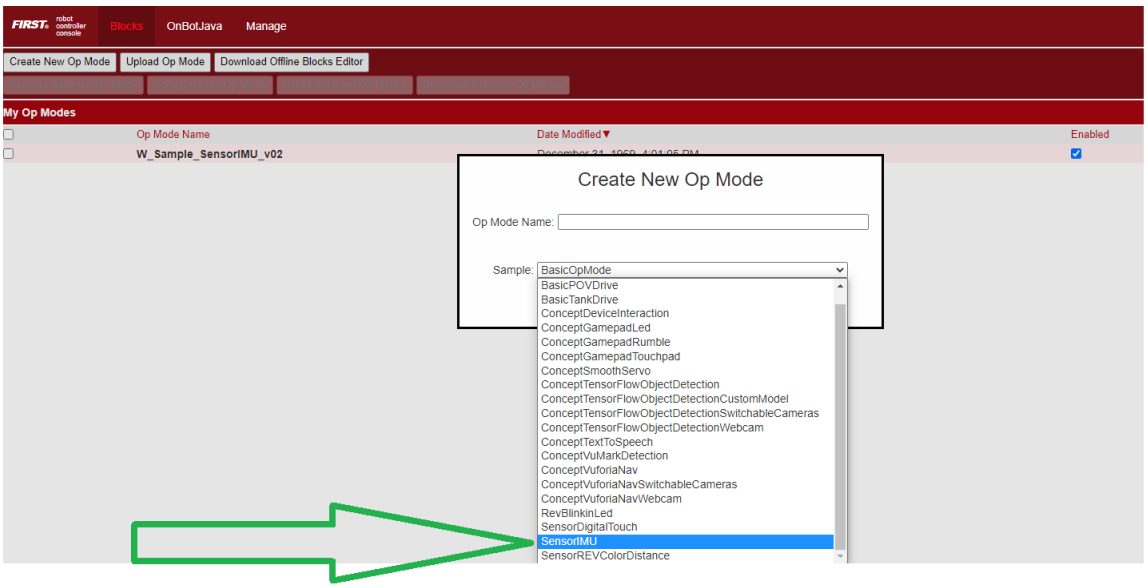
Java

- **ConecptExploringIMUOrientation.java**
- **SensorIMUOrthogonal.java**
- **SensorIMUNonOrthogonal.java**

Esses três exemplos em Java incluem comentários detalhados descrevendo a interface do IMU, consistentes com este tutorial. Em particular, o SensorIMUNonOrthogonal.java descreve três exemplos úteis.

Blocks

No Blocks, um exemplo simples é chamado SensorIMU.



Recursos do SDK

Programadores avançados são convidados a explorar a documentação Javadocs (API), particularmente em:

- com.qualcomm.robotcore.hardware

- org.firstinspires.ftc.robotcore.external.navigation

As novas classes universais de IMU para o SDK 8.1 são:

- IMU
- ImuOrientationOnRobot
- YawPitchRollAngles
- RevHubOrientationOnRobot

Os Javadocs descrevem outros métodos e variáveis do IMU que não são abordados neste tutorial básico.

Resumo

O SDK 8.1 fornece uma interface universal que suporta tanto o BHI260AP quanto o BNO055 IMU. Este tutorial básico introduziu algumas novas funcionalidades:

- A configuração do robô permite a seleção do tipo de IMU.
- Três maneiras de especificar a orientação de montagem do Hub no robô.
- Novos métodos em Blocks e Java para ler dados de ambos os tipos de IMU.

Revisão #2
Criado 2024-12-12 11:05:12 UTC por Enzo
Atualizado: 2025-05-16 13:40:35 UTC por João Vitor Loeblein