



This guide details step-by-step how to install and use the Blocks programming interface for the Pinpoint Odometry Computer. Please note that this guide is specific to FTC SDK/App version 10.1.

This example leverages the FTC “MyBlocks” system to allow users to add custom blocks to their library which implement Java functions. We have written a Java file to act as a companion to our Java driver. It requires installing two files to your robot via Onbot Java, allowing these custom blocks to be added to your library.

**For Pinpoint Setup, please refer to the Pinpoint User guide:**

[https://www.gobilda.com/content/user\\_manuals/3110-0002-0001%20User%20Guide.pdf](https://www.gobilda.com/content/user_manuals/3110-0002-0001%20User%20Guide.pdf)

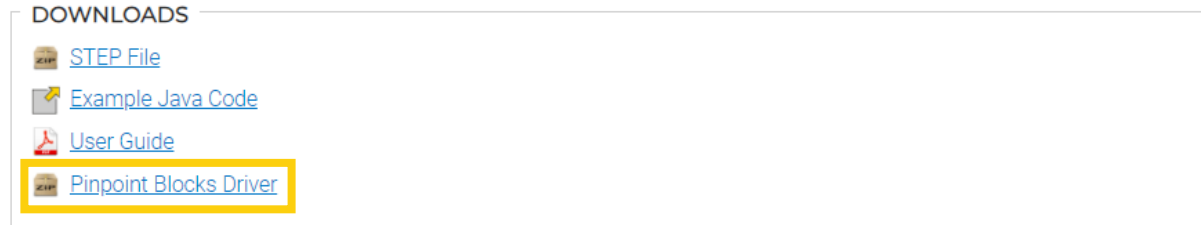
## Table of Contents:

<b>Step 1: Download Zip Blocks Files from goBILDA.com</b>	<b>-- 2</b>
<b>Step 2: Upload Driver files to your Robot</b>	<b>-- 2</b>
<b>Step 3: Build OnBot Java Files</b>	<b>-- 3</b>
<b>Step 4: Create new OpMode</b>	<b>-- 4</b>
<b>Step 5: Start Inserting Pinpoint Blocks!</b>	<b>-- 5</b>
<b>Blocks Functionality Overview</b>	<b>-- 6</b>
<b>Blocks Functionality Overview Continued</b>	<b>-- 7</b>
<b>Blocks Functionality Overview Continued</b>	<b>-- 8</b>

## Step 1: Download Zip Blocks Files from goBILDA.com

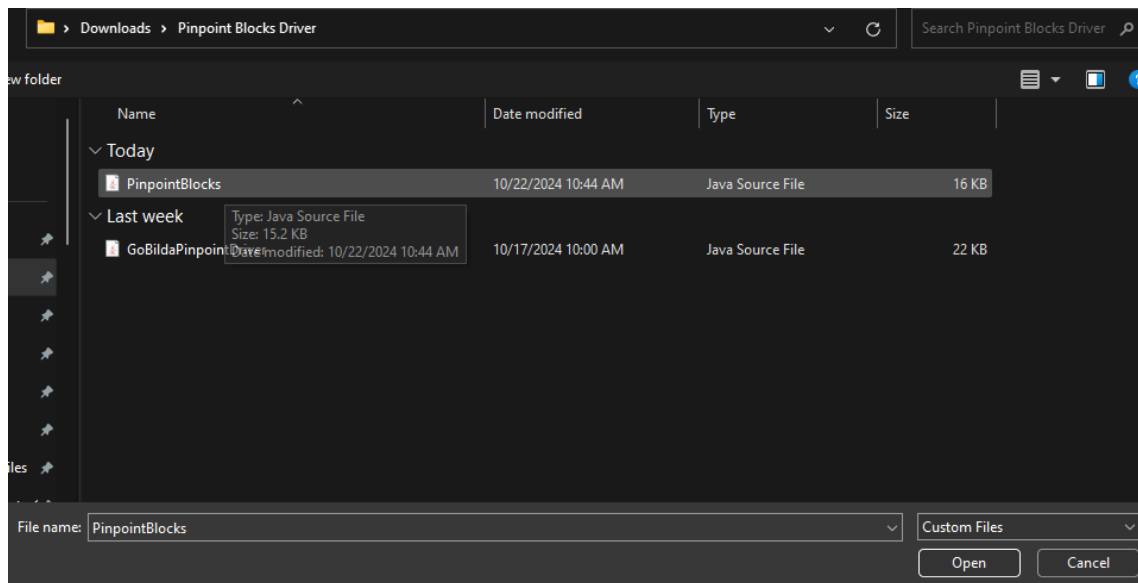
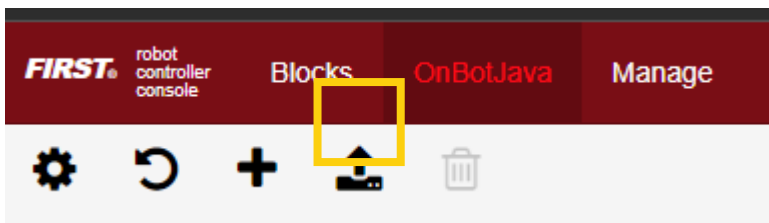
Follow this link to download the blocks driver and Pinpoint driver from our website. And once it is finished downloading, unzip it.

<https://www.gobilda.com/content/downloads/Pinpoint%20Blocks%20Driver.zip>



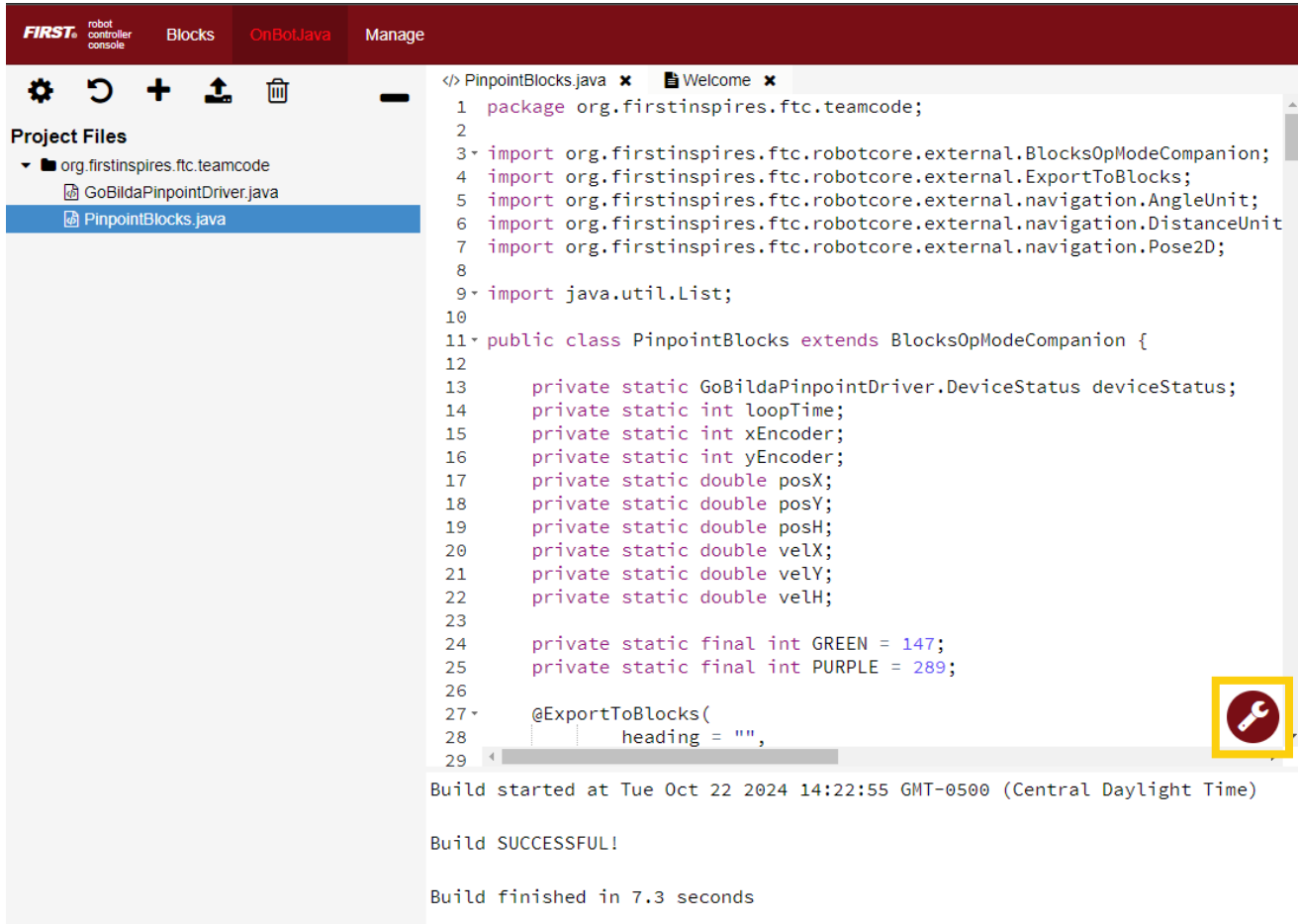
## Step 2: Upload Driver files to your Robot

Navigate to the OnBot Java tab of your Robot Controller either by using the REV Hardware Interface and a USB A to C cable, or by connecting to the Robot Controller's wifi network and going to 192.168.43.1:8080 in the browser of your choice. Before clicking the upload button and uploading both files one-at-a-time.



### Step 3: Build OnBot Java Files

Once both files have been uploaded, click either one and click the Build All button. You should see the "Build SUCCESSFUL!" message after a few seconds.



The screenshot shows the FIRST robot controller console interface. The top navigation bar includes "FIRST robot controller console", "Blocks", "OnBotJava", and "Manage". The left sidebar shows "Project Files" with a tree view containing "org.firstinspires.ftc.teamcode", "GoBildaPinpointDriver.java", and "PinpointBlocks.java". The main editor displays the Java code for "PinpointBlocks.java". A yellow box highlights the wrench icon in the bottom right corner of the editor, which is the "Build All" button. Below the code editor, the console output shows the build process starting at "Tue Oct 22 2024 14:22:55 GMT-0500 (Central Daylight Time)", followed by "Build SUCCESSFUL!" and "Build finished in 7.3 seconds".

```

1 package org.firstinspires.ftc.teamcode;
2
3 import org.firstinspires.ftc.robotcore.external.BlocksOpModeCompanion;
4 import org.firstinspires.ftc.robotcore.external.ExportToBlocks;
5 import org.firstinspires.ftc.robotcore.external.navigation.AngleUnit;
6 import org.firstinspires.ftc.robotcore.external.navigation.DistanceUnit;
7 import org.firstinspires.ftc.robotcore.external.navigation.Pose2D;
8
9 import java.util.List;
10
11 public class PinpointBlocks extends BlocksOpModeCompanion {
12
13     private static GoBildaPinpointDriver.DeviceStatus deviceStatus;
14     private static int loopTime;
15     private static int xEncoder;
16     private static int yEncoder;
17     private static double posX;
18     private static double posY;
19     private static double posH;
20     private static double velX;
21     private static double velY;
22     private static double velH;
23
24     private static final int GREEN = 147;
25     private static final int PURPLE = 289;
26
27     @ExportToBlocks(
28         heading = "",
29

```

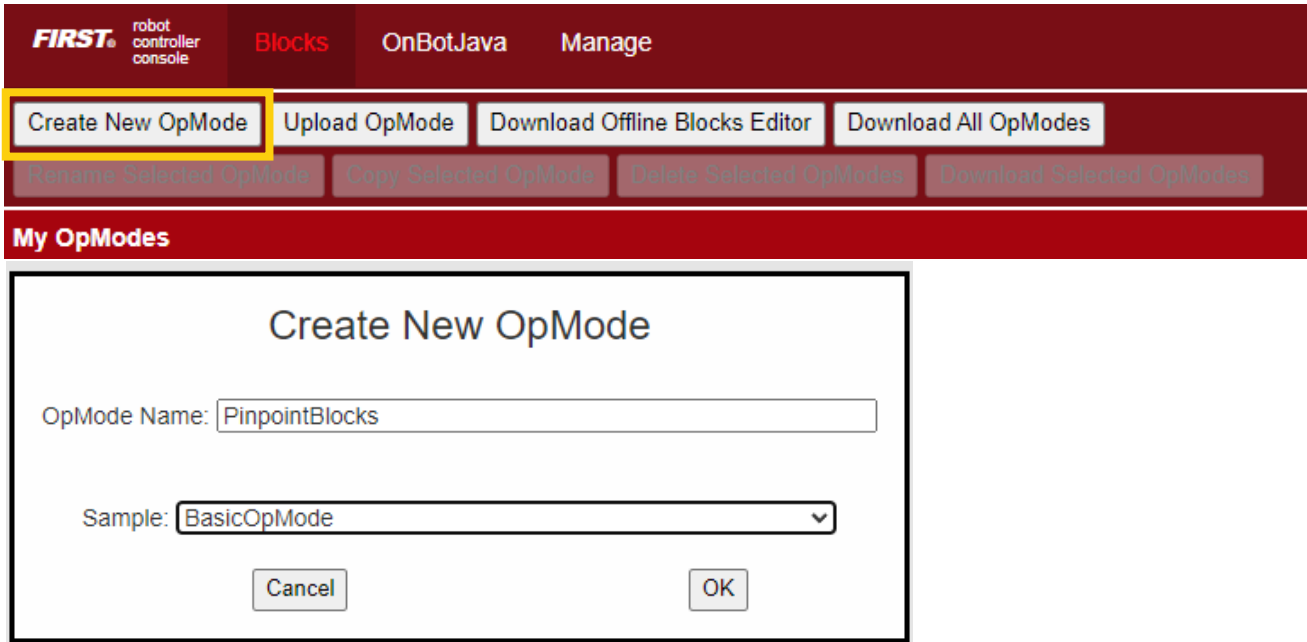
Build started at Tue Oct 22 2024 14:22:55 GMT-0500 (Central Daylight Time)

Build SUCCESSFUL!

Build finished in 7.3 seconds

#### Step 4: Create new OpMode

Navigate to the “Blocks” Tab, and create a new OpMode based on the BasicOpMode Sample.



The screenshot shows the FIRST robot controller console interface. The top navigation bar includes the FIRST logo, "robot controller console", and tabs for "Blocks", "OnBot.Java", and "Manage". Below the navigation bar, a row of buttons is visible: "Create New OpMode" (highlighted with a yellow border), "Upload OpMode", "Download Offline Blocks Editor", and "Download All OpModes". A second row of buttons includes "Rename Selected OpMode", "Copy Selected OpMode", "Delete Selected OpModes", and "Download Selected OpModes". Below these buttons is a section titled "My OpModes". A dialog box titled "Create New OpMode" is open, featuring a text input field for "OpMode Name:" containing the text "PinpointBlocks", and a dropdown menu for "Sample:" with "BasicOpMode" selected. At the bottom of the dialog box are "Cancel" and "OK" buttons.

## Step 5: Start Inserting Pinpoint Blocks!

Under the new “Java Classes” dropdown, you should see a PinpointBlocks section. This is where the interface for the Pinpoint sensor are stored. Each of these blocks have comments, which you can explore by clicking the blue “?” icon.

The screenshot shows the FIRST robot controller console interface. At the top, there's a navigation bar with 'FIRST robot controller console', 'Blocks', 'OnBotJava', and 'Manage'. Below this are buttons for 'Save OpMode', 'Export to Java', 'Download OpMode', and 'Download Image of Blocks'. The 'OpMode Name' is set to 'PinpointBlocks' and 'TeleOp' is selected. The 'Group' field is empty, and the 'Enabled' checkbox is checked. The left sidebar shows a tree view with 'Java Classes' expanded to 'PinpointBlocks'. The main workspace displays several PinpointBlocks blocks: 'deviceStatus', 'deviceVersion', 'encoderResolution' (set to 13.26291192 and CM), 'FourBarOdometryPod' (set to CM), 'frequency', 'headingVelocity' (set to DEGREES), and 'loopTime'. A 'set' block for 'offsets' is also visible, with 'Units to use' set to CM, 'X Pod Offset' set to -84, and 'Y Pod Offset' set to -168.

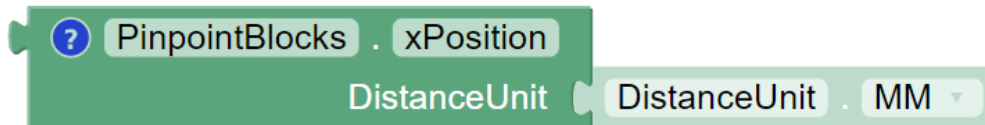
## Blocks Functionality Overview

For more information on the Pinpoint's behaviors, and for a troubleshooting guide, refer to the Pinpoint User Guide:

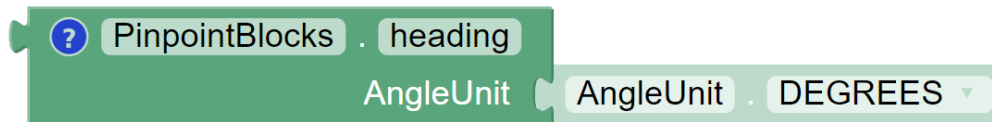
[https://www.gobilda.com/content/user\\_manuals/3110-0002-0001%20User%20Guide.pdf](https://www.gobilda.com/content/user_manuals/3110-0002-0001%20User%20Guide.pdf)

 call PinpointBlocks . update

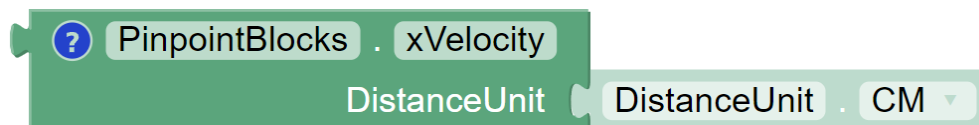
Call this function once per loop to get fresh data from the device. It updates your position, velocity, device status, loop time, frequency, and raw encoder data.

 PinpointBlocks . xPosition  
DistanceUnit DistanceUnit . MM

xPosition and yPosition both return their respective estimated robot positions. In the DistanceUnit input, specify the unit of measurement in which you'd like to receive the data.

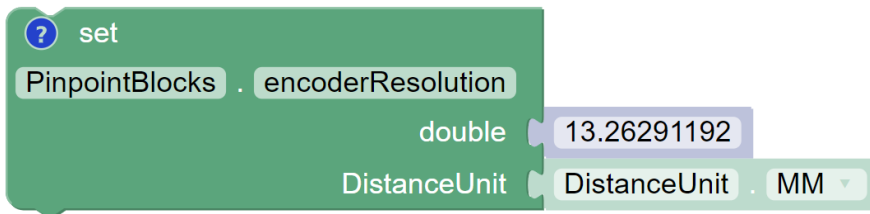
 PinpointBlocks . heading  
AngleUnit AngleUnit . DEGREES

This returns the current estimated heading of the robot. Specify the unit you'd like to receive with the AngleUnit input.

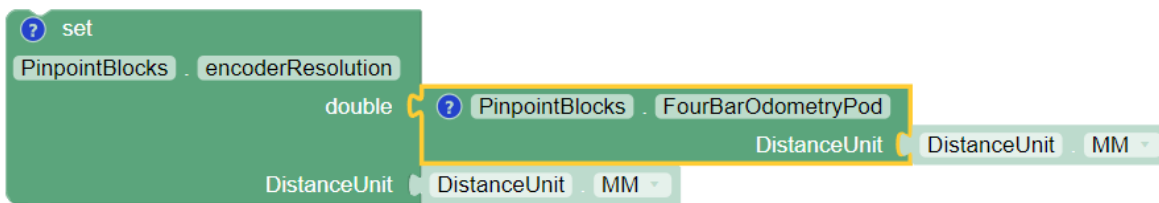
 PinpointBlocks . xVelocity  
DistanceUnit DistanceUnit . CM

Equivalent to xPosition, yPosition, and heading. You can get the xVelocity, yVelocity, and headingVelocity. These return the velocity in the Distance/Angle you request per second. So if you ask for inches, they return inches/second.

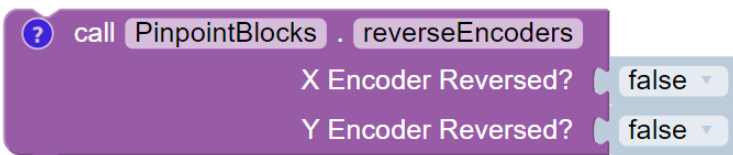
## Blocks Functionality Overview Continued



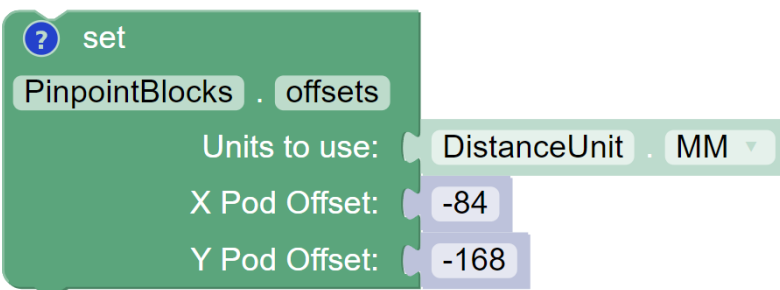
This sets the encoderResolution: the number of encoder ticks per DistanceUnit that your odometry pods report.



If you are using goBILDA® Odometry Pods, these numbers are already captured for you as two blocks titled FourBarOdometryPod and SwingarmOdometryPod. Make sure that both DistanceUnits match. If one is MM, both should be MM.



This reverses one or both of the encoders plugged into the Pinpoint. The X encoder should increase when the robot is moved forward, and the Y encoder should increase when it is moved left. If either of these are backwards, set their respective Boolean to true.



This sets the Odometry Pod Offsets for your robot. Select a DistanceUnit, and input how far to the left from the center of your robot the X pod is, and how far forward of the center your Y pod is. A graphic is available in the [Pinpoint User Guide](#).

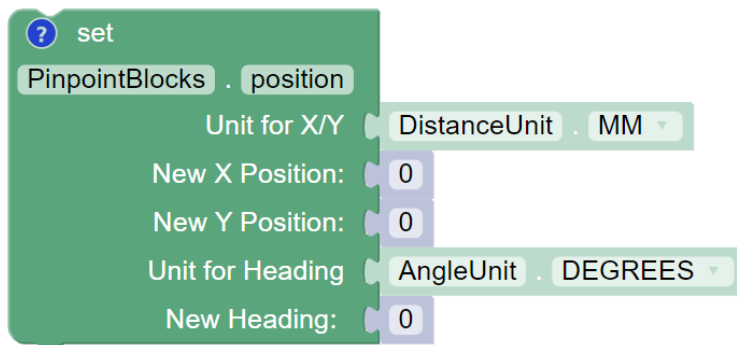
## Blocks Functionality Overview Continued



This recalibrates the IMU. If you get a bad calibration, the IMU will continually count up or down. If you notice that your heading seems to move when the robot is not moving, you may want to recalibrate it. It is a good idea to only call this when the robot is perfectly still.



This recalibrates the IMU and resets the estimated position to 0,0,0. This is a good idea to call while you are waiting for your OpMode to start if you get a bad initial calibration. Some drift may occur before you calibrate, so this ensures that your position is correct at the start of your autonomous.



This sets a new estimated position to your Pinpoint, which overwrites the current estimated position. This is useful when you have other sensors which feed back position data to your robot, or if you would like to apply an offset to what position your Pinpoint is reporting.

This concludes the overview of the Pinpoint Block Guide. If you have questions, please reach out to [tech@gobilda.com](mailto:tech@gobilda.com).