**ORIGINAL**

private double GetCurrentSpeed(double totalDistanceOnMeters, double oldTotalDistanceOnMeters, double millisecondsPassed)

{

if (millisecondsPassed == 0)

return 0;

var unconvertedSpeed = (totalDistanceOnMeters - oldTotalDistanceOnMeters) \* 1000 / millisecondsPassed;

return 3.6 \* unconvertedSpeed / 100;

}

private double GetAccelerationAngle(double accelerationA, double accelerationB)

{

double accelerationAngle = Math.Atan2(accelerationA, accelerationB);

double accelerationAngleOnDregress = accelerationAngle \* 180 / Math.PI;

return accelerationAngleOnDregress;

}

/// <summary>

/// Removes the centrifugal acceleration from the acceleration

/// </summary>

/// <param name="acceleration">The acceleration of angle who contains centrifugal acceleration</param>

/// <returns>acceleration without centrifugal force</returns>

private double RemoveCentrifugalAcceleration(double acceleration, double gyroscope)

{

double PolToMeters = 0.0254;

double gravityAcceleration = 9.81;

double sensorRadiusPositionOnPol = 0.9 \* Car.WheelRadiusInternal;

double sensorRadiusPositionOnMeters = sensorRadiusPositionOnPol \* PolToMeters;

double centrifugalAccelerationOnMPS2 = sensorRadiusPositionOnMeters \* Math.Pow((Math.PI / 180) \* gyroscope, 2);

double centrifugalAccelerationOnG = centrifugalAccelerationOnMPS2 / gravityAcceleration;

return acceleration + centrifugalAccelerationOnG;

}

/// <summary>

/// Reduces the noise from factors.

/// </summary>

private double FilterNoise(double factor, double filtredFactor, double alpha)

{

return (1 - alpha) \* factor + alpha \* filtredFactor;

}

**MODIFICADO**

private double GetCurrentSpeed(double totalDistanceOnMeters, double oldTotalDistanceOnMeters, double millisecondsPassed)

{

if (millisecondsPassed == 0)

return 0;

var unconvertedSpeed = (totalDistanceOnMeters - oldTotalDistanceOnMeters) \* 1000 / millisecondsPassed;

return 0; // sergio AjusteMov

}

private double GetAccelerationAngle(double accelerationA, double accelerationB)

{

//sergio AjusteMov

double deadZone = 0.2;

if (Math.Abs(accelerationB) > deadZone)

AUX\_AccelerationAngle = AUX\_AccelerationAngle + 12 \* Math.Sign(accelerationB) \* (Math.Abs(accelerationB) - deadZone); // sergio AjusteMov

return AUX\_AccelerationAngle;

}

private double RemoveCentrifugalAcceleration(double acceleration, double gyroscope)

{

return acceleration + Math.PI \* gyroscope / 10099; //sergio AjusteMov

}

private double FilterNoise(double factor, double filtredFactor, double alpha)

{

return (1 - alpha\*0.02) \* factor + alpha \*0.02\* filtredFactor; // sergio AjusteMov

}