GyroBoy – a self-balancing robot programmed in JAVA with leJOS EV3

Christensen, Bjørn Klint

Publication date:
2017

Document Version
Publisher's PDF, also known as Version of record

GyroBoy – a self-balancing robot programmed in JAVA with leJOS EV3.

by
Bjørn Christensen, DTU. March 2017.

Abstract

GyroBoy is a self-balancing robot created by LEGO as a demonstration of what you can build with the EV3 LEGO Mindstorms education kit. The kit includes building instructions as well as a control program for GyroBoy developed in LEGO´s so called block-programming language. In this article I will present a similar control program developed in Java using the leJOS EV3 class library.

Figure 1: GyroBoy. The robot is 23 cm high and weighs 822 gram.

What you'll need.

To build GyroBoy you need the LEGO Mindstorms EV3 Core Education Set. The LEGO product number is 45544. Please note that there also exists a retail set with product number 31313. This set is somewhat cheaper, but it does not include the gyro sensor needed to keep the robot upright. Building instructions comes with the Core set. You can also google them on the internet.

From the core set GyroBoy uses the EV3 gyro sensor and two large EV3 motors. GyroBoy also makes use of a few other sensors and motors that do not take part in balancing the robot. Hence, they are not described further in this article.

In addition to the Core set, you'll need a SD card for the leJOS installation[i] and a Wi-Fi dongle. I have a 32 MB Micro SD card from Kingston, and I use the Edimax n150 nano Wi-Fi dongle. For program development I have a Windows Laptop with Eclipse Neon IDE. I use leJOS version 0.9.0.

Complete source code can be found on Github: https://github.com/BjornChristensen/Gyroboy

Recent work

The act of balancing the robot is very similar to that of balancing a Segway[ii]. It uses a feedback control loop that continuously computes motor power based on the robots position and how much it tilts.

A few other Segway-like LEGO robots have been implemented in Java with the leJOS class library. They are NXTway[iii] , Marvin[iv] and GELway[v]. All of these robots are built from the LEGO Mindstorms NXT kit. This kit is an earlier generation of the EV3 kit and has a different computer and a bit different sensors. The anatomies of the robots are also different from GyroBoy. This means that the control programs for these robots are unsuitable for GyroBoy, and can only serve as an inspiration.

The balancing act

To balance the robot we need readings from the gyro sensor and from the motors. From the gyro sensor we get the angle velocity. The angle φ is how much the robot tilts forward or backwards and the angle velocity $\dot{\varphi}$ is how fast it falls. See Figure 2. The angle is measured in degrees. An angle value of zero means that the robot is perfect upright. A positive angle means that the robot is tilting forwards, and a negative angle means it is tilting backwards. The angle speed is measured in degrees pr. second. A positive angle velocity means that the robot is falling forward, and a negative angle velocity means that the robot is falling backwards.

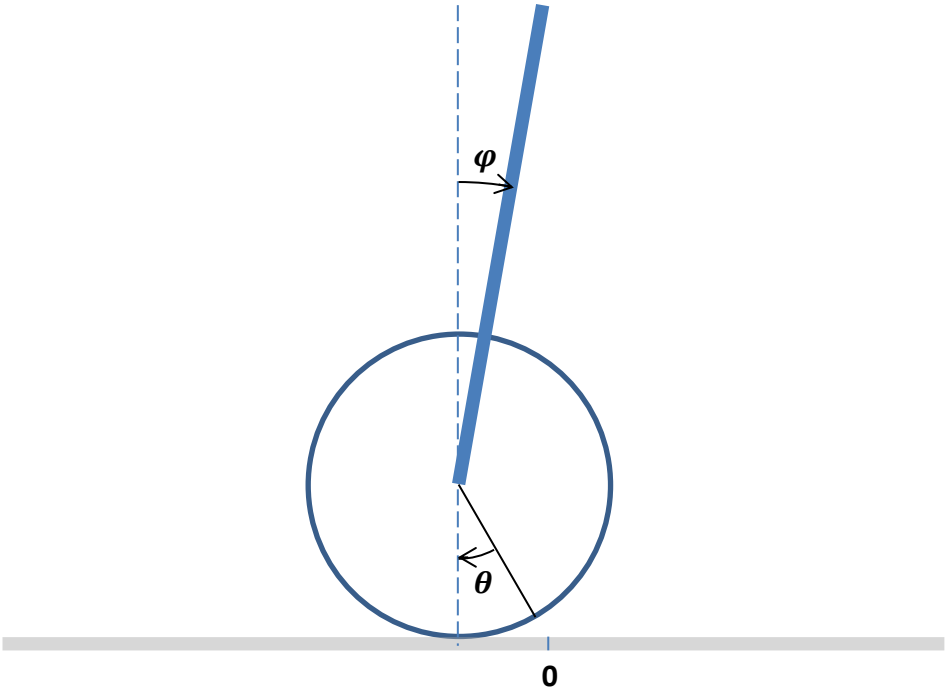

Figure 2: Tilt and motor angels of the robot.

From the motor we can read how much the wheels have rotated since the start of the program. This is called the taco-count and is measured in degrees. It is shown as the angle θ in Figure 2. A positive angle means that the robot has moved forward from its starting position. The control program will try to drive

the robot back to its original position while at the same time keeping the balance. A negative angle, as shown in the figure, means that the robot has backed up from its starting position, and the control program will try to drive it forward again.

The EV3 gyro sensor can provide both angle and angle velocity readings, but we use only the angle velocity from the sensor. The angle is then computed from the angle velocity by summation. The EV3 gyro sensor is connected to our program by the EV3GyroSensor class in leJOS. To read the angle velocity EV3GyroSensor uses an inner class called RateMode. For some reason not clear to me, the EV3GyroSensor class negates the angle velocity before it is returned to the program. Therefore it has been necessary to re-negate the returned value to its true value before use.

The motors are connected via the UnregulatedMotor class. This allows us to directly set the power of the motors and read the taco-count.

The control loop

Balancing the robot is carried out in a so called feedback control loop shown in Figure 3.

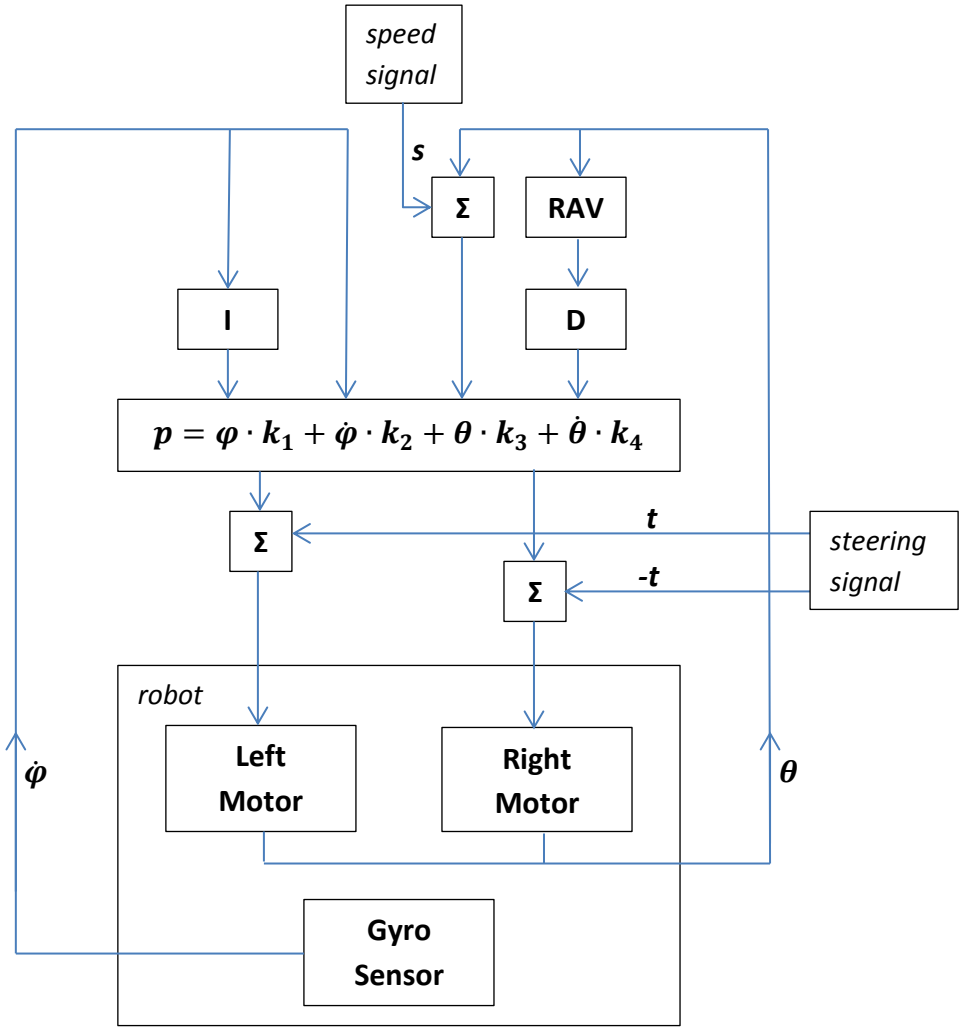

Figure 3: The feedback control loop

In the control loop the so called error signals, the gyro angle velocity $\dot{\varphi}$ and the motor rotation angle θ , (also referred to as the motors taco-count) are feed back into the equation that computes the new power p for the motors – also known as the control signal.

In addition the gyro angle φ is found by integration over time and feed into the equation. Finally, the motor rotational speed $\dot{\theta}$ is found by first taking a running average with length four of θ and then take the derivative with respect to time. The time step in the loop is set to Δt=10ms.

The control constants are taken directly from the earlier mentioned LEGO block program. The values are:

k_1=15
k_2=0.8
k_3=0.12
k_4=0.08

As can be seen from the constants, the gyro angle is by far the most important parameter.

Moving the robot around

External control signals can be given to set the robots forward speed, and to steer the robot.

By adding a displacement value ***s*** to the motor rotational angle in every iteration of the control loop, the robot can be driven forward or backward. Notice, that the control loop, in addition to keeping the robot upright, also will make the robot balance on its initial position. By adding a negative value of ***s***, the control loop is tricked to believe that the robots position is behind the starting position, and hence will drive it forward. A positive value of ***s*** will drive the robot backward.

Steering the robot left or right is done by adding a value ***t*** to the left motor power together with the same negative value ***–t*** to the right motor power. A positive value of ***t*** will turn the robot right, while a negative value will make it turn left.

Balance tests

A series of experiments have been carried out to test the robots balance. The following table lists the experiments and contains links to YouTube videos of them.

	Test	Video
1	Start up	https://youtu.be/T3V0LbOO54Q
2	Push the robot	https://youtu.be/Qf18qUppCq0
3	Carry a weight	https://youtu.be/I1ufGMO69uA
4	Cross a power cable	https://youtu.be/WPCPmn_Urt8 https://youtu.be/Pl5pxRvPfhk
5	Cross a Seesaw	https://youtu.be/ZZ4NTpCLefY
6	Go in circles	https://youtu.be/FKC-emTJI5M
7	Pirouette	https://youtu.be/YkczhQA-G5w
8	Difficult terrain	https://youtu.be/GdtXXitWZzc https://youtu.be/TteiU2HSt7o https://youtu.be/ucKSBQK34eY
9	Stand still	https://youtu.be/2w6GNVuPcc8

Figure 4: Balance tests carried out on March 14, 20117

Notice that the robot in the video clips is not exactly identical to the LEGO robot in Figure 1. It has a top mounted IR-sensor to receive steering commands. This of course, changes the dynamics of the robot a little bit, but it can still keep the balance with the same control system.

Conclusion

The tests show that the control program is able to balance GyroBoy quite well. Best results are seen when the robot is moving forward.

One of the most challenging tests was to balance the robot while keeping it still at the same spot. The video reveals that the robot will keep going forward and back with some overshoot. This problem might be helped with a fine tuning of the control constants to better suit the dynamics of this version of the GyroBoy robot.

Another issue is battery power. The robot and control program is not in itself very power consuming. But to react instantly on the error signals from the control loop, the motors need a nearly full charged battery. As power level drops, so does the stability of the robot. A time test showed that GyroBoy was able to go in a small circle for 47 minutes. By then the battery level had dropped from 8V (fully charged) to 7.8 V.

Yet another issue is gyro sensor drift. Many gyro sensors tend to drift causing the readings to be inaccurate over time. One way to deal with that is to continuously add a very small amount to the reading to compensate. This was not studied in this project because the robot did fine in the balance tests. These tests were rather small in duration. The time test mentioned above may have ended because of gyro sensor drift. This could be a subject for further study.

References

[i] For a description of leJOS se http://www.lejos.org/

[ii] Segway: See https://en.wikipedia.org/wiki/Segway_PT

[iii] NXTway: Yorihisa Yamamoto. NXTway-GS Model-Based Design-Control of self-balancing two-wheeled robot built with LEGO Mindstorms NXT. CYBERNET SYSTEMS CO., LTD., 2008.

[iv] Marvin: Johnny Rieper, Bent Bisballe Nyeng, and Kasper Sohn. Marvin - The Balancing Robot. Aarhus University, 2009.

[v] GELway: Steven J. Witzand. Coordinated LEGO Segways. 2009.