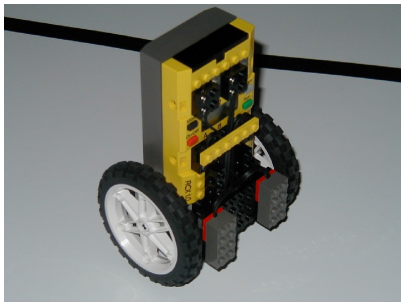
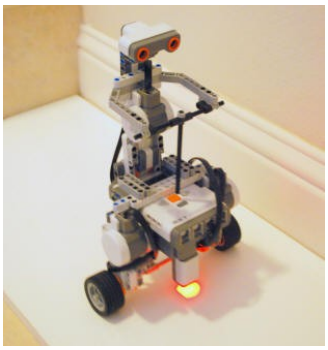
Introduction



The challenge of implementing Segway with LEGO robotics has been taken up many times. First came Steve Hassenplug's LegWay using the RCX and two EOPD sensors.

With the NXT, Philippe E. Hurbain built the NXTWay using the NXT Light Sensor in a way similar to Steve Hassenplug's LegWay.



Laurens Valk has recently published a Segway type robot that uses the HiTechnic Gyro Sensor. He calls his creation the AnyWay.

Now it is our time to come up with the original implementation of segway using LEGO robotics. We have called it Beerway.

Goals

The original goal was to implement gyro-stalilized remote control robot-segway using Android phone with a robotic hand.
In the end, we implemented gyro-stalilized robot-segway using Android phone. Remote control is working, but optimisation is needed.

Hardware used:
1) NXT brick
2) 2x motors for movement;
3) Samsung GT-N7000;
4) USB-OTG cable;
5) remote control device — web browser;

Software used:
LeJOS
Eclipse with Android and LeJOS plugins
Android 4.4
adb over wifi

Language:
Java

Project source code structure:

Beerway/GyroSensorApp — application running on the Android phone. Performs like USB client. Executes gyro sensor data and web interface commands transmission to the NXT device.

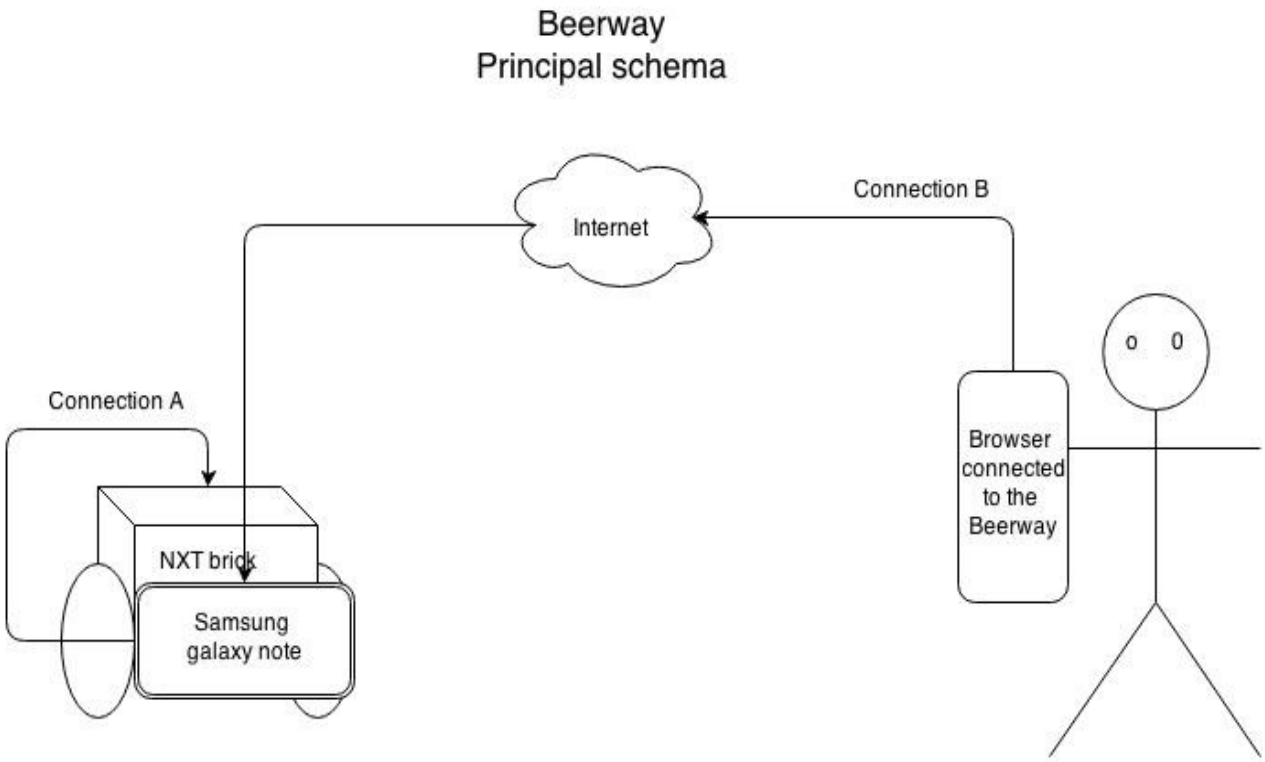
Beerway/NXTapp — application running on the NXT brick. Performs like USB server. Executes gyro sensor data readings and commands received from the phone.

Code:
https://bitbucket.org/cloun/beerway/src

Documentation:
Documentation is in the standard javadoc format. Can be found at doc/ subdirectory of source packages.

Example video:
https://www.youtube.com/watch?v=T9TtKiDFx3Y

How it works



Connection B is a http connection between user's browser and phone's micro web server. User works with Beerway's web interface by calling http enpoints implemented in the Beerway http server. They allow to move beerway forward, backward and rotate it.

Connection A is a high speed low latent USB connection between phone and NXT brick. It transmits gyro data from phone to the NXT brick and also commands from web server to NXT controlling code.

Application flow

GyroSensorApp

When user opens GyroSensorApp, it creates BeerwayServer and SegwayControl. BeerwayServer is a thread serving web requests from client's browser. SegwayControl is an object which allows to control NXT brick over USB. It registers AndroidGyroscope as a listener for gyro sensor changes. System thread detects angle changes and calls AndroidGyroscope method with current angle. AndroidGyroscope transmits these data via SegwayControl. BeerwayServer also transmits driving commands via SegwayControl to the NXT brick.
SegwayControl uses modified LeJOS code with added support of USB under Android. (Beerway/GyroSensorApp/src/lejos/pc/comm/NXTCommAndroidUSB.java Beerway/GyroSensorApp/src/lejos/pc/comm/NXTCommFactory.java:NXTCommFactory.createNXTComm)
Before connecting NXT, the user should manually start web server using application interface.

Then he should start NXTapp on the NXT brick.
Then he may connect phone to the NXT brick. After physical connection is established, the user can start data transfer using GyroSensorApp interface. After a click on "Connect to NXT", SegwayControl opens LCP connection over USB. Application shows current USB connection and web server statuses using text on the main activity screen and popup windows. It creates USB connection and starts sending gyro data every 16 milliseconds on average. Sometimes this can take 30 milliseconds, in this case Beerway has a nice chance to fall down. We assume that this is because of garbage collector launches on the phone and the NXT.

Opened connection triggers NXTapp's code on the NXT brick. It starts running a balancing thread and listening to gyrosensor data. After a couple of bips, the NXT brick should start balancing. Then the user is able to control Beerway through web browser.

So we have 3 threads running in paralel:
1) System thread(sends gyro sensor updates using SegwayControl instance)
2) UI thread(shows UI, creates SegwayControl, BeerwayServer instances and starts BeerwayServer. Reacts to user actions, shows connection status using UI)
3) Web server thread(runs web server)

NXTapp

When the user turns on the NXT and starts the NXTapp, SegwayServer starts listening to USB connection. When USB connection is established, it creates SegowayPilot instance. SegowayPilot allows to drive Beerway. SegowayPilot creates two threads:
1) Segoway thread which runs balancing cycle.
2) MoveControlRegulator thread which corrects variables corresponding to Beerway movement.

SegwayServer is a thread responsible for receiving data from USB stream.
It remembers commands coming from USB and writes gyro sensor data to the RemoteGyroscope instance. RemoteGyroscope calculates current angular velocity using previous and current angles received from the USB and times of previous and current reception. Segoway thread uses RemoteGyroscope instance to hold the Beerway.

SegwayController runs in the main application thread. It waits until the

SegwayServer has established the connection, then pulls comands from it and executes them using SegowayPilot instance. SegwayController performs blocking calls of SegowayPilot methods, the calls wait until robot is making current move (forward, backward, rotate). Every movement has fixed completion time. So the Beerway moves for specified time period and then waits for the next command from the USB.

So we have 4 threads running in paralel:
1) SegwayServer
2) Main thread running SegwayController
3) Segoway thread
4) MoveControlRegulator thread

Segoway runs balancing code every 8 milliseconds. It executes for about 2 milliseconds. Gyrosensor data is actualized every 15-20 milliseconds. It is actually ok to somehow hold the Beerway. The next step of project development should include the optimization of gyrosensor data rate. It could be speeded up 1.5-2 times. Also it should optimize data transfer protocol to use less bytes. It should make communication less latent and robot more stable.

Comparison of phones' gyro sensors

http://stackoverflow.com/a/9748489/2791535

Actually we are lucky ones since we didn't know anything about phone's gyro sensor in the beginning of the project and, as you can see above, most of the phones have quite low rate of gyro sensor data measurements.

Samsung Galaxy Note
---
* Awake
  Fastest: 100 Hz
* Asleep
  The phone does not send anything. That's why application keeps screen turned on.
Original segway algorithm used in our code has been developed for 300Hz gyro sensor.

Project flow/problems:
1) since it was discovered that BT connection has really high latency (sometimes up to 1 sec) we decided to use USB-OTG connection.
2) broken brick slowed down our progress.
3) first of all we tried to run lejos code with balancing cycle on the Android. We discovered a few problems.
We started from strange trace thrown by LeJOS
http://stackoverflow.com/questions/23049622/error-while-trying-establish-lejos-

USB-connection-on-Android. The problem was in libusb absent in the Android. Then we were trying to run Linux's libusb on Android because the original implementation of LeJOS uses it. We managed to compile it, but did not succeed in sending data over USB using it. Then we decided to implement NXTCommAndroidUSB.java that solved the problem. Then we discovered that Berway stability is very poor when balancing cycle runs on the phone.

This is because BaseMotor.forward() is blocking. It can make blocks up to 10msecs. This causes balancing cycles run with a time shift between obtaining gyro sensor data and actual motor's reaction to it. Also balancing cycles run much longer than original 10 milliseconds(average 16 msec max ~40sec). This makes robot very unstable. The longest achieved standing time was about 5 seconds. A long play with coefitients has been performed but with no luck.

Then we decided to move balancing code to NXT brick as in the original example. And bingo — it has become very stable but only for 5 seconds and after that it fell with exception..

We couldn't understand why this was happening. We thought that the problem was in the new rewritten code. It has lead us to the wrong way of reasoning. As we discovered after, the problem wasn't in our code but in LeJOS implementation.
We posted error on LeJOS forum
http://www.lejos.org/forum/viewtopic.php?f=7&t=6952

The error is caused by the fact that the LeJOS menu expects all USB communication to follow the LCP protocol. Since the menu is fed non-LCP-conforming data, LeJOS crashes after a while.

So we have solved this problem by sending data after the program on NXT has started listening.
Then the web server was implemented.
True story:)

During this project we discovered that:
1) Android base 4.1.2 rom has 2-times higher rate of sensor updates than cyanogen 11(Android 4.4)
2) Phone's touchscreen works ok with adhesive tape glued to:)

References

Android documentation https://developer.Android.com/guide/index.html

LeJOS documentation http://www.lejos.org/nxt/nxj/tutorial/index.htm

Explanation of both the physics of the problem and the linear equation that makes a solution possible on the NXT http://caxapa.ru/thumbs/320181/nxtway-g-1.pdf