

Lego Mindstorms Segway Project

Authors of project:

Łukasz Bondyra,
Paweł Górka,
Jakub Tutro,
Krzysztof Wesołowski

Under the supervision of:

Adam Piłat Ph.D.

Documentation drawn up by:

Łukasz Bondyra

WEBSITE:

<http://home.agh.edu.pl/~ap/segway>

2009-04-20

Basic Lego Mindstorm project was inspired by Yoriyama Yamamoto.

It was developed in the academic research club INTEGRA, working in the Electrical faculty in the AGH – University of Science and Technology.

The Lego Mindstorms project is part of the bigger one, which aims to create a Segway human transporter machine.

The following works were made:

I. Installation of necessary software.

All applications need to be installed/extracted to non – space directories.

1. **Cygwin 1.5.25-15** - a Linux - like environment for Windows which enables to run some Linux applications in Windows environment. It includes GCC compiler.
INSTALATION: Devel Default -> “make: The GNU version of the ‘make’ utility” must be marked.
IMPORTANT: Cygwin has to be 1.5.x or newer version.
WEBSITE: <http://www.cygwin.com>
2. **GNU ARM toolchain 4.0.2** - a distribution of GCC (GNU Compiler Collection) for ARM core which supports an ARM7 CPU in the NXT
INSTALATION: “Floating point unit” doesn’t have to be marked
IMPORTANT: Using other version of GCC may cause unexpected errors.
WEBSITE:
<http://www.gnuarm.com/bu-2.16.1 gcc-4.0.2-c-c++ nl-1.14.0 qi-6.4.exe>
3. **MINDSTORMS NXT Driver v1.02** - This software installs/updates the LEGO MINDSTORMS NXT driver, which enables windows to recognize the NXT.
IMPORTANT: While downloading new firmware to the NXT, after re-inserting batteries, NXT is clicking. It requires downloading firmware again.
WEBSITE: <http://mindstorms.lego.com/Support/Updates/>
4. **NeXTTool** - a PC console applications which enables uploading .rx (executable files) and .r (firmware) files to the NXT.
INSTALATION: Only extraction of archive is needed.
IMPORTANT: NeXTTool does not display any messages about successing/failing commands.
WEBSITE: <http://bricxcc.sourceforge.net/utilities.html>
5. **BricxCC 3. 3 (Bricx Command Center)** - integrated development environment (IDE) for programming the RCX, Scout, Cybermaster, and Spybot programmable bricks using Dave Baum's Not Quite C (NQC) language. It also supports programming Not eXactly C (NXC) language.

Version 3.3 of BricxCC is an enhanced revision to Mark Overmars' original application.

IMPORTANT: BricxCC setup file includes NeXTTool, so installing one makes step 4 unneeded.

WEBSITE: <http://bricxcc.sourceforge.net>

6. **MATLAB/Simulink 7.6.0 with Real Time Workshop**

7. **nxtOSEK 2.09** - provides C/C++ API which is called ECRobot. Originally designed to develop a MATLAB&Simulink based Model-Based Design environment (called Embedded Coder Robot) for LEGO MINDSTORMS NXT.

WEBSITE:

http://lejos-osek.sourceforge.net/download.htm?group_id=196690

8. **Embedded Coder Robot NXT 3.17** by Takashi Chikamasa – Model-Based Design with Production Code Generation.

INSTALLATION: Open ecrobotnxtsetup file in Matlab and follow tips.

IMPORTANT: Extract archive and paste “nxtOSEK” directory to
.../ecrobotNXT/environment/

WEBSITE: <http://www.mathworks.com/matlabcentral/fileexchange/13399>

9. **NXTway-GS (Self-Balancing Two-Wheeled Robot) Controller** by Yori-hisa Yamamoto based on Embedded Coder Robot Robot NXT- presents sample models and documents describe:

How to Build NXTway-GS

Mathematical Dynamics Model of NXTway-GS

Controller Design for Balance and Drive Control

NXTway-GS Model Illustration

Simulation and Experimental Results

WEBSITE: <http://www.mathworks.com/matlabcentral/fileexchange/19147>

10. **NXT GamePad 1.04** by Tomoki Fakuda – application which lets to control robot via bluetooth and analog gamepad

WEBSITE:

http://lejososek.sourceforge.net/download.htm?group_id=196690

II. Identification and weight measurement of each brick which robot consists of.

III. Robot re-construction.

We added to standard NXT Mindstorms robot a servo –motor and put the head of robot on the top of one. It made possible to nod by a robot what makes moment of inertia variable.

We also exchanged original Lego accumulator to rechargeable batteries.

**Yorihisa Yamamoto
construction:**



**Lego Segway AGH Team
model:**



IV. Upload John Hansen's Enhanced NXT firmware lms_arm_nbcnxc_107.rfw instead of Lego original one.

Enhanced NXT firmware enables NXT to run .rxo applications.

Upload was done by original NXT Lego Mindstorms software.

Attempt of upload firmware by NeXTTool unfortunately failed.

IMPORTANT: Application size is up to 64KB.

WEBSITE: <http://lejos-osek.sourceforge.net/installation.htm#UPLOAD>

V. Identification of Gyro Sensor

We created a C++/QT for Bluetooth communication application which saved reads from Gyro Sensor to .csv file. We put robot on the floor in turn: on back, on front, on wheels and on head (he was lying motionlessly, supported by wall).

Afterwards we used robot as a pendulum with axis located in place of its wheel axis.

We ran our application and deflected robot of 180 degrees, to head – up position

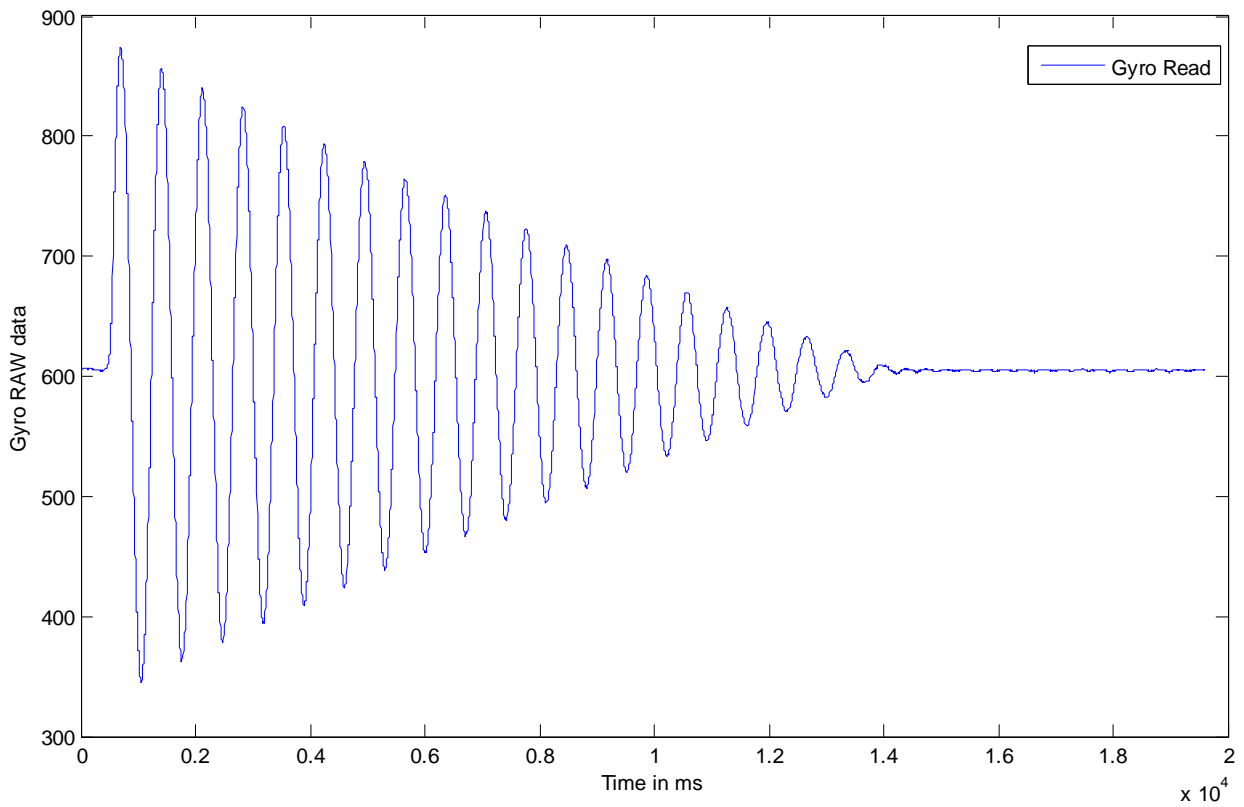
and released it. Each measurement last 50s and in each 10 000 reads with 200Hz frequency were done.

Lego Mindstorms Segway Project

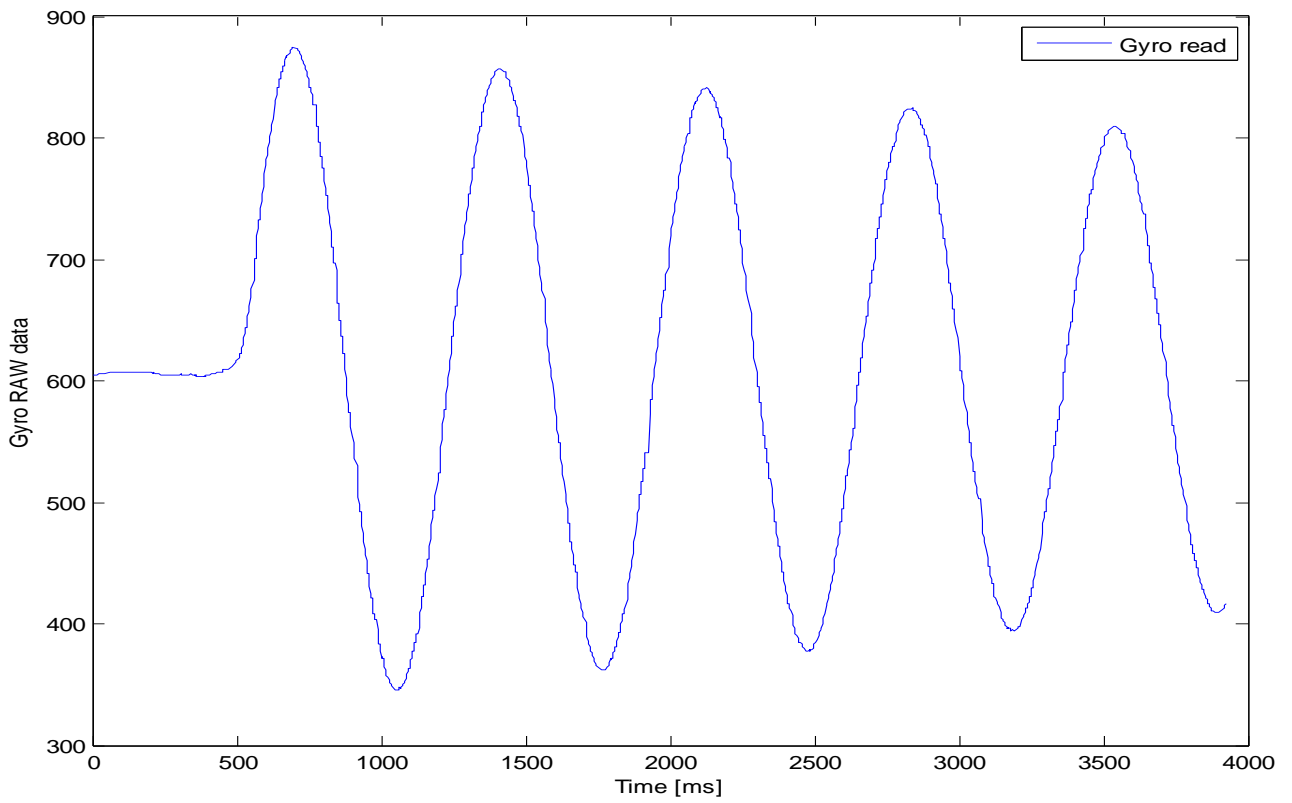
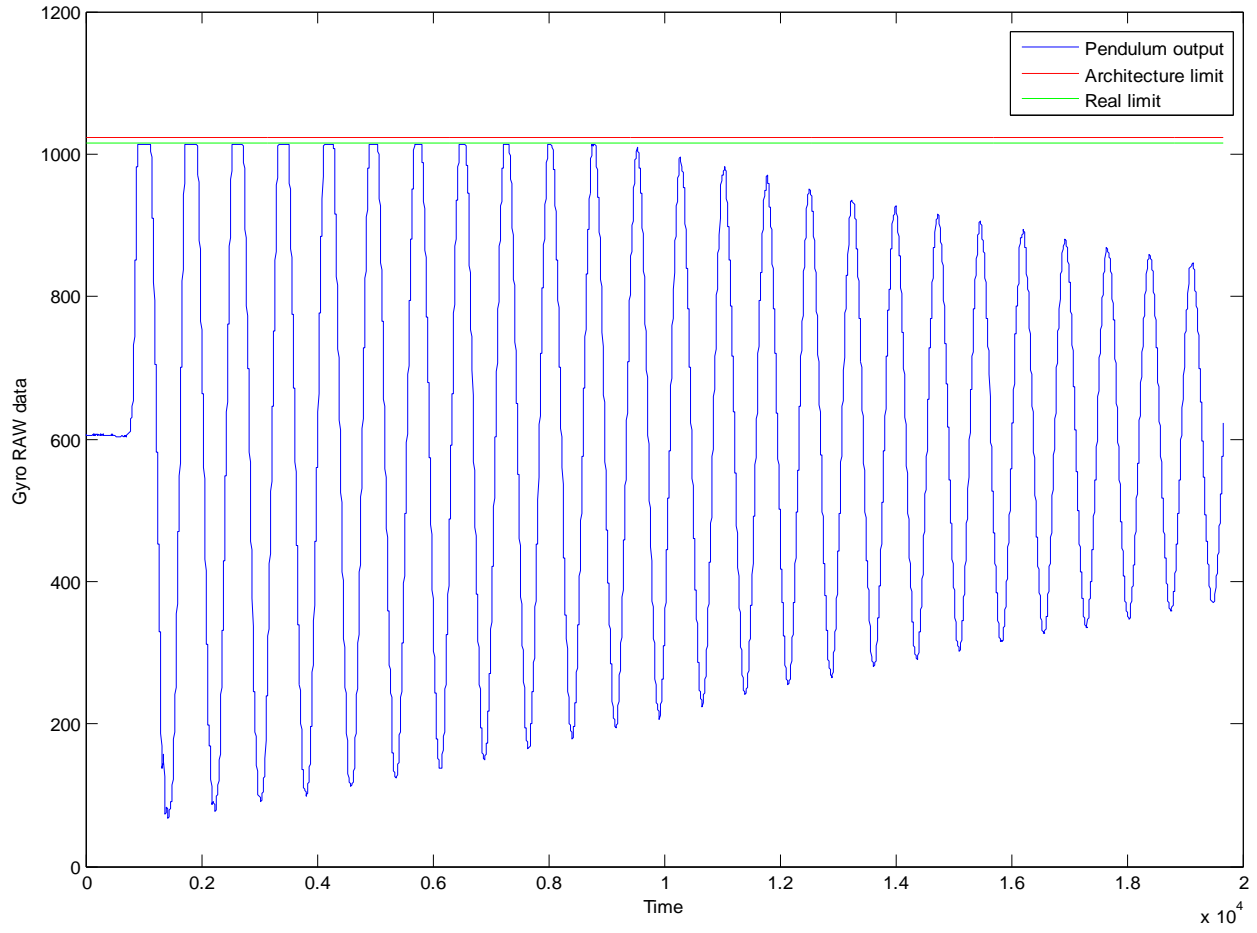
Raw data results:

Position	Mean	Standard deviation
On wheels	605.0847	0.3221
On head	605.0682	0.3158
On back	605.0690	0.3207
On forward	605.0528	0.3153

Plots:



Lego Mindstorms Segway Project



Summary:

1. Gyro Sensor is sensor of angular velocity.
2. GS has an offset – when angular velocity is zero, read of sensor is about 605
3. When lying motionlessly, noises amount to about 0.053% of mean read
4. Official GS measurement range is from 0 to 1024
5. Real GS measurement range is from 0 to 1015
6. Ability of measuring top and down value of angular velocity is not the same for each. Real range of negative velocity is 605, while range of positive velocity is only 410. This causes value-cut when positive velocity crosses over 1015 value (may be seen on the enclosed plot).

VI. Generation and upload .rx file which controls and stabilizes robot

Generation made by Matlab, upload done automatically by Matlab using NeXTTool application.

Source: “.../nxtway_gs/nxtway_gs_controller.mdl”

EFFECT: Robot isn't stabilized and falls down

VII. Correction of parameters

File: “.../nxtway_gs/param_plant.m”

1. Wheel weight [kg]:
m=0.03 -> m=2*0.0185
2. Wheel radius [m]:
R=0.04 -> R=0.03
3. Body weight [kg]:
M=0.6 -> M=0.641
4. Body width [m]
W=0.14 -> W=0.068
5. Body height [m]
H=0.144 -> H=0.244

File: “.../nxtway_gs/param_controller.m”

1. PWM offset
pwm_offset=0 -> pwm_offset=2
2. Sound frequency [Hz]
sound_freq=440 -> sound_freq=240
3. Distance threshold for obstacle avoidance [cm]
dst_thr=20 -> dst_thr=10

EFFECT: Robot is stabilized quite well.

VIII. LQ regulator modification

Extra anti-windup feature added, which aims to improve controller performance. It decreases integral value when output is saturated (when disturbance is received).

1. LQ variables:
 - Theta – wheels angle difference
 - Psi – (integral of psi) angle of vertical robot position
 - Thetadot – (derivative of theta) angular velocity of wheels
 - Psidot – angular velocity of vertical robot position
2. Variables used to turn robot (not used in LQ regulator)
 - Phi – horizontal angle of robot
 - Phidot – horizontal velocity of robot

IX. Correction of autonomous drive mode

Not included in this model.

X. Connecting robot to computer via Bluetooth and controlling it by analog game pad

Robots were presented in open Day in AGH – University of Science of Technology. There were great conditions to test our Bluetooth among another Bluetooth devices.

IMPORTANT: When there are a lot of Bluetooth devices in area, it is difficult to connect to a robot.

XI. Rebuild of NXTway-GS Controller

1. Shared data

- Removed

- **flag_mode** – autonomous/remote control drive
- **flag_auto** – autonomous drive is/isn't ready
- **flag_avoid** – robot is/isn't avoiding obstacle

- Added

- **head_initial_pos** – initial position for head – servo
- **head_pos** – target head angle (range: -90 <-> 90 degrees - because of robot construction)
- **initial_theta_difference** – initial wheels angle difference
- **turn_angle** – horizontal body angle difference (range: int 32)
- **forward_cmd** – robot speed (range: -100 <-> 100 which is scaled to -0.3 <-> 0.3 m/s)

2. Task_init

- Removed autonomous/remote control drive mode switch (this software version doesn't need to use remote control drive mode)
- Data storage head_initial_pos is read from head – servo motor to store initial angle of head (servo gives absolute angle value)
- Data storage initial_theta_difference is read from two wheel - servo motors as a difference of their angles (for the same reason as head_initial_pos)

3. Task_ts1

- Removed
 - In “Balance & Drive Control -> Balance Controller -> Cal x1 – current state – PV” subsystem: Removed doubled theta output (making LQ regulator subsystem makes this variable unneeded)
 - In “Balance & Drive Control -> Balance Controller -> Cal Reference State SP” subsystem:
Removed doubled theta output (making LQ regulator subsystem makes this variable unneeded)
Deleted block responsible for scalling phidot_cmd variable that is used when using game pad (we use P regulator to position phi angle now)
 - “Balance & Drive Control -> Controller -> Data logging” subsystem removed. It was responsible for logging body angle by NXT GamePad ADC Data Logger.
 - “Balance & Drive Control -> Controller -> Make command” subsystem removed. It was responsible for receiving command via bluetooth and setting values of robot vertical and horizontal velocities (thetadot and phidot) or for autonomous drive mode (autonomous drive is controlled now by separate task)
 - “Balance & Drive Control -> Balance Controller -> Calculate PWM” subsystem: Robot wheel synchronizing blocks removed (P regulator used for turning keeps now a difference constant when driving straightforward)
- Created/Inserted
 - “Balance & Drive Control -> Balance Controller -> Head Positioning” subsystem inserted.
This subsystem is a control loop feedback mechanism, which is realized as a proportional regulator. Input signal is set point added to initial angle of head. Error is received by deducting actual head angle and this signal is multiplied by gain. Actually gain is equal 2 and is determined by a

compromise: head moves quite fast but not enough to destabilize robot. Control variable goes to saturation subsystem and to PWM output.

- “Balance & Drive Control -> Balance Controller -> Calculate PWM -> left/right anti backlash” subsystem created. It is responsible for reducing wheel backlash. It reduces swinging our robot while not moving.
 - In “Balance & Drive Control -> Balance Controller -> Calculate PWM” subsystem anti-windup feature added to integral part of regulator. Anti-windup decreases integral when output is saturated on motors.
 - “Balance & Drive Control -> Balance Controller -> Servo-Motor Controller” subsystem created. It contains P regulator which positions phi angle (difference wheel angle) what makes previous model synchronization redundant.
 - “Balance & Drive Control -> Balance Controller -> LQ Regulator” subsystem. Blocks from “Balance & Drive Control -> Controller ” were grouped to make model more readable. It was necessary after adding anti-windup feature to model (which complicated model a lot).
4. Task_ts2 removed (responsible for obstacle check and autonomous drive)
 5. Task_ts3 – nothing changed
 6. Task_ts4 – created by us.
 - During compilation, model loads a file with robot way in matrix form. Matrix is split up into vectors of control, which are loaded into look-up tables and at runtime used as turn_angle, head_pos and forward_cmd variables. They are used to generating proper robot motion.
 - Block responsible for speaking when an obstacle is seen or introducing and thanking by a robot (it depends of used software version).

7. Task sampling
ts1 = 0.004 [s]
ts2 = 0.02 [s]
ts3 = 0.1 [s]
ts4 = 0.01 [s]

Complete model with detailed implementations is enclosed to this document.

XII. Driving specified way

To drive a specified way, model needs a look-up table, where should be specified control variable of three servos, which is provided every 100ms. To do this we specified way by creating vectors. After that points located in these vectors were specified.

Every two points create a tiny vector. In fact, our robot moves straight and curves are driven as lots of tiny lines.

Every three points enable us to count angle between two neighbouring lines. When robots drives to end of one, before driving another line has to turn that counted angle.

We created small C++/QT application which having two points, specifies vector of control variable. Control variable is provided every 100ms to PWM so number of elements of control variable vector is counted by our application.

This reduces specifying way to painting vectors.

XIII. Stabilizing spectrum of masses

In the end we re-build a robot to check ability of stabilizing random masses. We removed head servo and build a basket at front. We put there random masses up to doubled robot mass which is about 0.6 kg.

EFFECT: Robot leant backward and stabilized well but was less able to move – they moved slower and heavier.

XIV. Accomplished targets:

1. Identification of brick masses and robot moment of inertia.
2. Written and tested software for data acquisition.
3. Analysis and identification of collected data.
4. Diagnosis of Bluetooth communication for remote control tasks, data logging.
5. New and individual robot construction with variable moment of inertia property, enhanced control possibilities.

6. Improvement of the control quality by the controller tuning.
7. Programming of the robots movement cycle for demonstration purposes.
8. Get used to:
 - Matlab/Simulink, Real Time Workshop
 - NXT Osek Real Time Operation System
 - Embedded Coder
 - Erobot NXT