TALLINNA TEHNIKAÜLIKOOL
Automaatikainstituut

Nikolai Vidjajev
ISS70LT

Pöördpendli juhtimine NXT 1.0 komplektiga

Magistritöö

Juhendaja**: Andres Rähni**

Tallinn 2014

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjuandusallikatest ja mujalt pärinevad andmed on töös viidatud.

04.06.2014 [illegible]

(kuupäev) (allkiri)

Pöördpendli juhtimine NXT 1.0 komplektiga

Annotatsioon

Magistritöö on koostatud traditsioonilise struktuurigaja koosneb sissejuhatusest, põhiosast, mis omakorda koosneb kuuest peatükist, kokkuvõtvatest järeldustest ning kasutatud kirjanduse loetelust.

Töö peamine eesmärk on pühendatud nägemusele, et pöördpendli suhteliselt soodne ja sobiv mudel on võimalik koostada kasutades NXT1.0-i peamise platvormina. Eeldatakse, et loodava mudeli saab programmeerida töötama autonoomselt eelistatult tundmatus, reaalses keskkonnas ja kommunikeeruma ning analüüsima oma ümbrust erinevate andurite ja kaasaegse andmeside kaudu.

Selle lahendamiseks uuriti kogu komplekti võimalusi ja pöördpendli füüsikalist mudelit ning koostati töötav mudel. Esialgsetes testides ilmnenud probleemidele leiti lahendused ja realiseeritud mudelis tehti tõsiseid muudatusi kontrolleri püsivaras, andurites ja programmeerimisvahendistes.

Töö tulemusena on valminud ja testitud sobiv pöördpendli töötav mudel. Testide tulemused kinnitasid, et pöördpendel suudab säilitada tasakaalu ja reageerib häiringutele ning mõjutustele reaalajas, vältides takistusi ja suhtleb teiste seadmetega ning nende kaudu inimestega.

Lõputöö on kirjutatud inglise keeles ning koosneb kuuest peatükist, mis moodustab 92 lehekülge teksti, kus on toodud 40 joonist ning 12 tabelit.

Inverted Pendulum Control on NXT 1.0 Set

Annotation

This graduation thesis was written in a traditional structure and contains: an introduction, a main part that consists of six chapters and a conclusion. Also bibliography and appendixes are presented.

Main aim of this thesis is devoted to proving that a suitable model of an inverted pendulum could be build at a relatively low price by using NXT 1.0 as the main platform. It is assumed that the model in question can be programmed to work autonomously in a real world and preferably unknown environment with the means of communicating (to) and analyzing its surroundings by using various sensors and modern communicational protocols.

To resolve this problem a conclusive analysis of the entire set and physical model of an inverted pendulum ware made and a test model was build. Thanks to initial tests some problems were found and the model underwent heavy modifications that included modification of firmware, sensors and programming environments.

As a result, of work presented in this thesis, a suitable model of an inverted pendulum was build and tested. Test had shown that the model in question could maintain balance and react to disturbances and applied forces in real time, while avoiding obstacles and communicating with other devices and thru them to people.

The thesis is written in English and contains 92 pages of text, 6 chapters, 40 figures, 12 tables.

Table of content:

Tables of content

List of Figures:

List of Tables

List of Appendixes

List of Abbreviations

Table 0.1 Physical parameters of the model

Symbol	Number	Units	Meaning
θ	0-360	Degrees	Average angle of both wheels
Ψ	0-360	Degrees	Angle of body pitch
ϕ	0-360	Degrees	Angle of body yaw
θ_m	0-360	Degrees	Motor angle
J		kgm^2	inertia moment
$J\omega$	$0.1*10^{-4}$	kgm^2	Wheel inertia moment
$J\phi$		kgm^2	Body yaw inertial moment
$J\psi$		kgm^2	Body pitch inertial moment
J_m		kgm^2	Motor inertial moment
f_m			Friction 5oefficient between body and DC motor
f_ω			Friction coeffiecient between body and DC motor
g	9.81	m/sec^2	Gravity acceleration
m	0.030	kg	Wheel / base weight
r	0.056	Meters	Wheel radius
M	0.628 0.715	kg	Body weight
W	0.15 0.145	Meters	Body width
D	0.05	Meters	Body depth
H	0.29 – 0.32	Meters	Body height
L	0.0725 - 0.08	Meters	Distance of the center of mass
R_m	6.7	Ohm	DC motor resistance
K_b	0.46	V sec/ rad	DC motor back EMF constant
K_t	0.31	Nm/A	DC motor torque constant
F(u)		N	Applied force
X		Units	Wheels place

Glossary

Table 0.2 – Glossary

Abbreviations	Meaning
PC	Personal Computer
IDE	Integrated Development Environment
PID	Proportional–Integral–Derivative controller
P	Proportional gain
I	Integral gain
D	Derivative gain
Robot / K-way / Light Segway / model	An inverted pendulum model name
LDD	Lego Digital Designer
FTC	First Tech Challenge
LEGO	LEGO (Danish for Leg Godt – play Good) Company name
NXT	Lego controller
NXT-G	Programming environment developed by LEGO
RobotC	Programming environment on C language
LabVIEW	Programming environment for a visual programming language
App Inventor	Programming environment on java language for android
VLC	VideoLAN is a project that develops software for playing video across a local area network

Required Products List

Table 0.3 – Products List

Required Products Product	Version	Release
LabVIEW for Lego Mindstorm	2010	2010 SP1
LabVIEW for FTC	2011	2010 SP1
leJOS NXJ	0.9	2011
RobotC VW (RVW)	3.0	2011
NXT-G	1.0 2.0	2006 – 2010
App Inventor	1.2	2012
NXT firmware	Note that most programs can run on 1.31 version of the firmware	2011
Android	2.3.7 – 4.1.2	2011– 2012

Required Products List

File List

Table 0.4 – File list

File	Description
K-way.LDD and Segway.ldd	Ldd designs
Robot Segway / K/ –light .rxe / nxt-G	NXT-G programs of robots
Test (number).avi	Recorded tests
Test (number).data	Recorded data via LabVIEW
Segway_additional_test.rbt	Segway program with Bluetooth
K-Way v104576,VI	K-way program LabVIEW
K-WAY v104576	NXT-G programmed K-WAY
K-WAY.C	RobotC program for K-WAY
Segway V10.C	Segway with robotC
grapher4all.vi	Graph drawer for K-way
Motorcheck.vi	LabVIEW program for motion checks
Mobile applications	Programs on smart-phones
TTU 104576	Remote control and socialize adaptation program
Delivery	Stuff caring authorization algorithm
Questionersy	Social localization algorithm
Parading bot	Flag recognition and photo recognition option

Introduction

Stabilization and multitasking nowadays represent most demandable aspects of both automation and robotics. Almost every modern device should be able to do several tasks simultaneously and remain within a corridor of parameters. The main aim of this thesis is to build a model of an Inverted pendulum using an NXT 1.0 set that will be able to balance while performing other tasks in an unknown environment. The model in question is presumed to be the base for future modifications and therefore must be cheap, re-constructible and programmable in several modern programming languages. To achieve autonomy the model must be equipped with several sensors to avoid obstacles and dangerous situations. Additionally the model in question should poses an ability to communicate via common network protocols to send and receive actual data from other devices and analyze it in a real time. Some parts of the code should be reprogrammable in a real time to make the model more adaptable to changing environment.

First chapter is devoted to describing the problem and deriving needed equations.

Second chapter gives a thorough overview of the NXT 1.0 set

Third chapter describes tests done on an inverted pendulum model with a light sensor. Results are analyzed, and notable problems were found and corrected. As a result, limitations of the model in question are listed.

Forth chapter describes modifications done to all aspects of the model.

Fifth chapter describes tests of multiple tasking for the final model.

Sixed chapter is devoted to remote control, data transfer, communications and analysis to / of a surrounding environment.

A conclusion of the entire thesis is presented at the final part of this thesis.

Chapter 1 - Introduction and Physical model

1.1 Inverted pendulum

An inverted pendulum is a pendulum which has its mass above its pivot point. [1]
An inverted pendulum is inherently unstable, and must be actively balanced in order to remain upright; this can be done by:

- By applying torque to the pivot point.
- By moving the pivot point horizontally in this case the whole model.

A model reviewed in this thesis is a two wheeled Inverted pendulum.

Inverted pendulum is a set of wheels and a body - pendulum, fixed to the wheels, so that the pendulum can move along the same plane on the 180^o. The pendulum can move in a straight line in both directions and turn in any direction.

The inverted pendulum system should also be resistant to external forces and noises. Schematic representation of this system is shown on Figure 1.1.

Motivation for choosing the thesis topic.

The system reviewed in this thesis:

- Is associated with one of the most complicated problems of modern robotics - the problem of stability and balance (balancing) of self-propelled (walking) vehicles. The inverted pendulum is a classical problem of dynamics and control theory that is widely used for testing control algorithms all over the world both for simplest and advanced algorithms.
- This system is a simplified algorithm of human walking (motion /ambulation): Inverted pendulum resembles a motion of one leg. [2] [3] One of the built models resembles a standing man or balancing acrobat.
  Research showed that Inverted pendulum model was used in variety of useful systems.[4]

<u>Historical In early models of:</u>

- Seismographs. [5]
- Chronometers.
- Polygraphs.
- Sail holders.
- Various stabilizing parts in military vehicles.
- Targeting systems aids.

<u>Modern:</u>

- Level Control in a Surge Tank.
- Carousels and various attractions.
- Rockets. [6]
- As a testing model for many algorithms.
- Is a part of mobile crane system.
- Segway.
- Base model for animations of locomotion.
- Plane stabilization (wheels)

In author bachelor's thesis it has been proved that a similar system can be designed and tested in MatLab / simulink environments. As a result of that work [1] it was proved that best ways of building and controlling “Inverted pendulum on a cart” control systems is by using PID controllers or closed loop feedback.

1.2 Stated aims

The aim of this thesis is to create Inverted Pendulum model(s) using NXT 1.0 set**.** Code a control program for the model in question to maintain balance at all times. Then build applications and/or programs, so that the model could be controlled remotely via modern communication network and communicate with the surrounding environment (including other terminals) in real time and understand it.

An Ideal model should be able to work autonomously in various unknown environments, with options of remote control and monitoring from another terminal(s).

The main credentials of the model are:

- The model in question should be portable (mobile)
- The cost of the model in question should as cheap as possible.
- The model in question should be able to understand surrounding environment and communicate with it.
- The model in question could be used for academic researches and teaching.
- The model in question could be easily modified in a short period of time.
- The model in question could be programmed in various languages and program environments.
- Pats and program code should be replaceable / reprogrammable in live time (meaning that some parts of the programmed code could be recorded and implemented on the model in question, that is already doing a preset program.)

1.3 Concept and physical model of the inverted pendulum

The task of an Inverted pendulum system is to maintain the pendulum in the upright position and maximally fast counteract all external influences on the system as soon as possible, i.e. counteract the disturbance exerted on the system and return the pendulum to the upright position. Theoretically, the system should not allow the pendulum fall at all times. But, to maintain mobility credential, those rules are slightly bended.

The main aim in at this point is to make the design of an inverted pendulum model and code it to maintain balance.

Main credential for this part of work is to find a suitable compromised state between

1. Maximal and minimal errors / disturbances, that may be corrected / compensated.
2. Speed of the control system response.
3. Controllability. (meaning both mobility and remote control functions)

Solution:

The most suitable compromise would be to code the system in a way that it could withstand some applied forces at a high but not maximal response time, because in the eyes of the control system: inverted pendulum is in an unstable state when it is mobile / moving and therefore, if the response time would be set very low, the inverted pendulum would be highly stable, but uncontrollable and close to an immobile system.

To build and code this system in is vital to describe this model by mathematical equations. In authors' previous thesis similar equations were calculated. [1] The following equations are a derived by modifying equations in [1] [7]

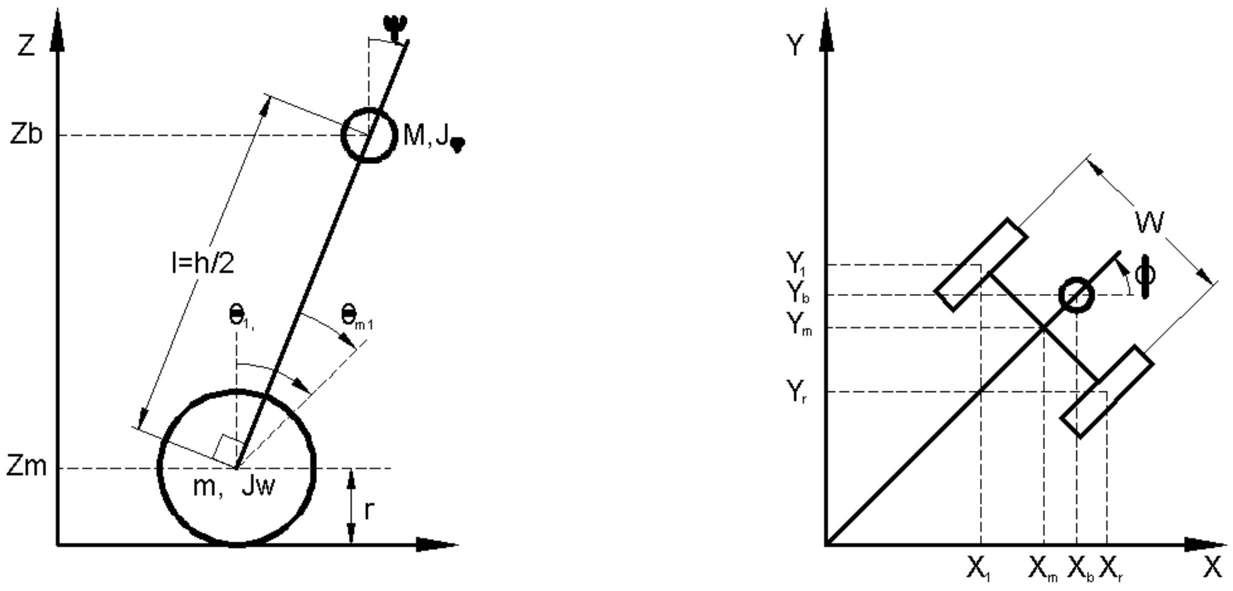


Figure 1.1 Coordinate system for equations of motion for the inverted pendulum (in Autocad).

Using the coordinate system presented (on Figure 1.1) and the Lagrange method the motion equations for a two-wheeled inverted pendulum can be derived:

If it assumed, that the direction of the wheels for the inverted pendulum is in the positive x - axis and time t=0, the following coordinates are given. The following variables are used in the proceeding equations:

θ - Average angle of both wheels
Ψ - Angle of body pitch
ϕ - Angle of body yaw Equation 1.1

$$(\theta,\phi)=\left(\frac{1}{2}(\theta_r+\theta_L),\frac{1}{2}(\theta_L-\theta_r)\right)$$ Equation 1.2

$$\dot{x}_m = r\dot{\theta}\cos\phi$$
$$\dot{y}_m = r\dot{\theta}\sin\phi$$ Equation 1.3

$$x_r = x_m + \frac{W}{2}\sin\phi$$
$$y_r = y_m - \frac{W}{2}\cos\phi$$
$$z_r = z_m$$ Equation 1.4

$$x_l = x_m - \frac{W}{2}\sin\phi$$
$$y_l = y_m + \frac{W}{2}\cos\phi$$
$$z_l = z_m$$ Equation 1.5

$$x_b = x_m + l\sin\psi\cos\phi$$
$$y_b = y_m + l\sin\psi\sin\phi$$
$$z_b = z_m + l\cos\psi$$ Equation 1.6

T_1 - Translational kinetic energy, T_2 - the rotational kinetic energy and U- the potential energy can be described by those equations:

$$T_1 = \frac{1}{2}m(\dot{x}_1^2 + \dot{y}_1^2 + \dot{z}_1^2) + \frac{1}{2}m(\dot{x}_r^2 + \dot{y}_r^2 + \dot{z}_r^2) + \frac{1}{2}M(\dot{x}_b^2 + \dot{y}_b^2 + \dot{z}_b^2) \quad \text{Equation 2.1}$$

$$T_2 = \frac{1}{2}J_\omega \dot{\theta}_l^2 + \frac{1}{2}J_\omega \dot{\theta}_r^2 \frac{1}{2}J_\psi \dot{\psi}^2 + \frac{1}{2}J_\phi \dot{\phi}^2 + \frac{1}{2}n^2 J_m \left(\dot{\theta}_l - \dot{\psi}\right)^2 \quad \text{Equation 2.2}$$

$$U = mgz_l + mgz_r + Mgz_b$$

Using the Lagrange method that determents in this case:

$$L = T_1 + T_2 - U\,, \quad \text{Equation 3}$$

Where those forces are equal to:

$$F_\theta = \frac{d}{dt}\left(\frac{\partial l}{\partial \dot{\theta}}\right) - \frac{\partial l}{\partial \theta} \quad \text{Equation 4.1}$$

$$F_\phi = \frac{d}{dt}\left(\frac{\partial l}{\partial \dot{\phi}}\right) - \frac{\partial l}{\partial \phi} \quad \text{Equation 4.2}$$

$$F_\psi = \frac{d}{dt}\left(\frac{\partial l}{\partial \dot{\psi}}\right) - \frac{\partial l}{\partial \psi} \quad \text{Equation 4.3}$$

By deriving the following equations the forces would be:

$$\begin{aligned}
F_\theta &= \left((2m + M)r^2 + 2J_\omega + 2n^2 J_m\right)\ddot{\theta} + \left(Mlr\cos\psi - 2n^2 J_m\right)\ddot{\psi} - Mlr\dot{\psi}^2 \sin\psi \\
F_\phi &= \left(\frac{1}{2}mW^2 + J_\phi + \frac{W^2}{2r^2}\left(J_\omega + n^2 J_m\right) + Ml^2 \sin^2\psi\right)\ddot{\phi} + 2Ml^2 \dot{\psi}\dot{\phi}\sin\psi\cos\psi \quad \text{Equation 5} \\
F_\psi &= \left(Mlr\cos\psi - 2n^2 J_m\right)\ddot{\theta} + \left(Ml^2 + J_\psi + 2n^2 J_m\right)\ddot{\psi} - Mgl\sin\Psi - Ml^2\dot{\phi}^2 \sin\Psi\cos\Psi
\end{aligned}$$

Looking forward, it is vital to mention, that although in most used IDEs and coding programs the voltages of the chosen motors will be presented and calculated by the firmware and/or IDE. It is imperative to write down those equations to get a proper mathematical model of the inverted pendulum. Using the DC motor torque and viscous friction, the generalized forces can be described as:

$$F_\theta = F_l + F_r\ ,\ \ where\ F_{l,r} - Force\ of\ the\ left\ \ ,right\ wheel. \quad \text{Equation 6.1}$$

$$F_l = nK_t i_l + f_m(\dot{\psi} - \dot{\theta}_l) - f_\omega \dot{\theta}_l \quad \text{Equation 6.2}$$

$$F_r = nK_t i_r + f_m(\dot{\psi} - \dot{\theta}_r) - f_\omega \dot{\theta}_r \quad \text{Equation 6.3}$$

$$F_\psi = -nK_t i_l - nK_t i_r - f_m(\dot{\psi} - \dot{\theta}_r) - - f_m(\dot{\psi} - \dot{\theta}_l) \quad \text{Equation 6.4}$$

$$F_\phi = \frac{W}{2r}\left(F_r - F_l\right) \quad \text{Equation 6.5}$$

Due to the fact that, the chosen motors use PWM to read motors voltages. The DC motor equation is:

$$L_m i_{l,r} = \upsilon_{l,r} + K_b(\dot{\psi} - \dot{\theta}_{l,r}) - R_m i_{l,r} \quad \text{Equation 7}$$

Using this equation, the force expressions can be found in terms of the motor voltage:

$$\alpha = \frac{nK_t}{R_m}$$ Equation 8.1

$$\beta = \frac{nK_t K_b}{R_m} + f_m$$ Equation 8.2

$$F_\theta = \alpha(v_l + v_r) - 2(\beta + f_\omega)\dot{\theta} + 2\beta\dot{\psi}$$ Equation 8.3

$$F_\psi = -\alpha(v_l + v_r) - 2\dot{\theta} - 2\beta\dot{\psi}$$ Equation 8.4

$$F_\phi = \frac{W}{2r}\alpha(v_r - v_l) - \frac{W^2}{2r^2}(\beta + f_\omega)\dot{\phi}$$ Equation 8.5

To simplify the motion equations they can be linearized about the Inverted pendulum balance point thus making $\Psi \to 0$ meaning that $\sin\psi = \psi$, $\cos\psi = 1$ and second order terms ($\dot{\psi}^2$, $\dot{\psi}\dot{\phi}$, $\dot{\phi}^2$) – may be neglected due to the fact that they are considerably smaller compared to other terms. So the equations will be:

$$F_\theta = \left((2m + M)r^2 + 2J_\omega + 2n^2 J_m\right)\ddot{\theta} + \left(Mlr\cos\psi - 2n^2 J_m\right)\ddot{\psi}$$ Equation 9.1

$$F_\phi = \left(\frac{1}{2}mW^2 + J_\phi + \frac{W^2}{2r^2}\left(J_\omega + n^2 J_m\right)\right)\ddot{\phi}$$ Equation 9.2

$$F_\psi = \left(Mlr - 2n^2 J_m\right)\ddot{\theta} + \left(Ml^2 + J_\psi + 2n^2 J_m\right)\ddot{\psi} - Mgl\Psi$$ Equation 9.3

F_θ and F_ψ can be grouped together as they both contain angles θ and ψ . So they can be written as:

$$S = \begin{bmatrix} (2m + M)r^2 + 2J_\omega + 2n^2 J_m & Mlr\cos\psi - 2n^2 J_m \\ Mlr\cos\psi - 2n^2 J_m & Ml^2 + J_\psi + 2n^2 J_m \end{bmatrix}$$ Equation 10.1

$$T = 2\begin{bmatrix} \beta + f_\omega - \beta & -\beta \\ -\beta & \beta \end{bmatrix}$$ Equation 10.2

$$U = 2\begin{bmatrix} 0 & 0 \\ 0 & -Mgl \end{bmatrix}$$ Equation 10.3

$$V = 2\begin{bmatrix} \alpha & \alpha \\ -\alpha & -\alpha \end{bmatrix}$$ Equation 10.4

$$V\begin{bmatrix} v_l \\ v_r \end{bmatrix} = \begin{bmatrix} \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} + T\begin{bmatrix} \dot{\theta} \\ \dot{\psi} \end{bmatrix} + U\begin{bmatrix} \theta \\ \psi \end{bmatrix}$$ Equation 10.5

Furthermore F_ϕ can be expressed as:

$$I = \frac{mW^2}{2} + J_\phi + \frac{W^2}{2r^2}\left(J_\omega + n^2 + J_m\right)$$ Equation 11.1

$$J = \frac{W^2}{2r^2}(\beta + f_\omega)$$ Equation 11.2

$$K = \frac{W}{2r}\alpha$$ Equation 11.3

$$I\ddot{\phi} + J\dot{\phi} = K(v_r - v_l)$$ Equation 11.4

Using all of the above equations a State – space equation can be written as:

$$\dot{x}_2 = A_2 x_2 + B_2 u$$ Equation 12.1

$$x_1 = \begin{bmatrix} \theta \\ \psi \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \; x_2 = \begin{bmatrix} \phi \\ \dot{\phi} \end{bmatrix} \; u = \begin{bmatrix} v_l \\ v_r \end{bmatrix}$$ Equation 12.2

$$\dot{x}_1 = A_1 x_1 + B_1 u \quad \dot{x}_2 = A_2 x_2 + B_2 u$$ Equation 12.3

$$A_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & A_1(3,\ 2) & A_1(3,\ 3) & A_1(3,\ 4) \\ 0 & A_1(4,\ 2) & A_1(4,\ 3) & A_1(4,\ 4) \end{bmatrix},\ B_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ B_1(3,\ 1) & B_1(3,\ 2) \\ B_1(4,\ 1) & B_1(4,\ 2) \end{bmatrix}$$ Equation. 12.4

$$A_2 = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{j}{l} \end{bmatrix}, \quad B_2 = \begin{bmatrix} 0 & 0 \\ -\frac{K}{l} & -\frac{K}{l} \end{bmatrix},$$ Equation 12.5

$$A_1(3,\ 2) = \frac{-gMle(1,\ 2)}{\det(E)}$$ Equation 12.6

$$A_1(4,\ 2) = \frac{gMle(1,\ 1)}{\det(E)}$$ Equation 12.7

$$A_1(3,\ 3) = \frac{-2[(\beta + f_\omega)E(2,\ 2) + \beta E(1,\ 2)]}{\det(E)}$$ Equation 12.8

$$A_1(4,\ 3) = \frac{2[(\beta + f_\omega)E(1\ 2) + \beta E(1,\ 1)]}{\det(E)}$$ Equation 12.9

$$A_1(3,\ 4) = \frac{2\beta[E(2,\ 2) + E(1,\ 2)]}{\det(E)}$$ Equation 12.10

$$A_1(4,\ 4) = \frac{-2\beta[E(1,\ 1) + E(1,\ 2)]}{\det(E)}$$ Equation 12.11

$$B_1(3,\ 1) = B_1(3,\ 2) = \frac{a[E(2,\ 2) + E(1,\ 2)]}{\det(E)}$$ Equation 12.12

$$B_1(4,\ 1) = B_1(4,\ 2) = \frac{-a[E(2,\ 2) + E(1,\ 2)]}{\det(E)}$$ Equation 12.13

$$\det(E) = E(1,\ 1)E(2,\ 2) - E(1,\ 2)^2$$ Equation 12.14

1.4 PID regulator [8]

PID regulator is the main regulator used in this thesis and it is logical to describe it early by following equations:

K_p – Proportional gain, a tuning parameter
K_i – Integral gain.
K_d – Derivative gain.
e – Error = SP – PV
t – Time or instantaneous time (the present).
τ – Variable of integration; takes on values from time 0 to the present t.

$$P = K_p e(t) \qquad \text{Equation 13.1}$$

$$I = K_i \int_0^t e(\tau) d\tau \qquad \text{Equation 13.2}$$

$$D = K_d \frac{d}{dt} e(t) \qquad \text{Equation 13.3}$$

So the PID is:

$$u(t) = MV(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t) \qquad \text{Equation 13.4}$$

A PID controller calculates an error value as the difference between a measured process variable and a desired setpoint. The controller minimizes the error by adjusting the process through use of three separate constant parameters: the proportional, the integral and derivative values. These values can be interpreted in terms of time: P depends on the present error, I on the accumulation of past errors, and D is a prediction of future errors, based on current rate of change. The weighted sum of these three actions is used to adjust the process via a control element.

To set a PID regulator in question mainly so-called manual method was used, but Ziegler–Nichols tuning method was also taking in the consideration.

1.5 Ziegler–Nichols tuning method [8,9]

Ziegler–Nichols tuning method is a heuristic method of tuning a PID controller.
It is performed by first setting Ki and Kd values to zero. Kp is then increased until it reaches the ultimate gain Ku, at which the output of the control loop oscillates with a constant amplitude. Ku and the oscillation period Tu are used to set the Kp, Ki and Kd gains depending on the type of controller used:

Table 1.1 - Ziegler–Nichols tuning method

Control type	K_p	K_i	K_d
p	0.5 K_u		
PI	0.45 K_u	1.2K_p/T_u	
PD	0.8 K_u		$K_pT_u/8$
PID			
Classical	0.6 K_u	2 K_p/T_u	$K_pT_u/8$
Pessen Integral Rule	0.7 K_u	0.4 K_p/T_u	0.15 K_pT_u
With overshoot	0.33 K_u	2 K_p/T_u	$K_pT_u/3$
No overshoot	0.2 K_u	2K_p/T_u	$K_pT_u/3$

1.6 Manual tuning method [8,9]

If the system must remain online, one tuning method is to first set Ki and Kd values to zero. Increase the Kp until the output of the loop oscillates, then the Kp should be set to approximately half of that value for a "quarter amplitude decay" type response. Then increase Ki until any offset is corrected in sufficient time for the process. However, too much Ki will cause instability. Finally, increase Kd, if required, until the loop is acceptably quick to reach its reference after a load disturbance. However, too much Kd will cause excessive response and overshoot.

A fast PID loop tuning usually overshoots slightly to reach the setpoint more quickly; however, some systems cannot accept overshoot, in which case an over-damped closed-loop system is required, which will require a Kp setting significantly less than half that of the Kp setting that was causing oscillation.

Table 1.2 - Manual tuning method

Parameter	K_p	K_i	K_d
Rise time	Decrease	Decrease	Minor change
Overshoot	Increase	Decrease	Decrease
Settling time	Small change	Increase	Decrease
Steady-state error	Decrease	Eliminate	No effect in theory
Stability	Degrade	Degrade	Improve if K_d is small

Now that all mathematical equations are stated next phases of construction could be done.

Chapter 2 – NXT 1.0 set

2.1 Materials Requirements

To design and build an inverted pendulum model (later referred to as simply “model) proper materials needed to be acquired. The materials for constructing the model were chosen based on their:

1. Low complexity – Because one of the thesis goals is to build a model that could be modified (including future modifications) with relevant ease. It is also essential if the model is presumed to be used in academic researches and teaching.
2. Cost – Because initially author set a budget of 400-500 Euros for an entire project.
3. Weight – because the model should be portable. (also this is author preference due to the fact that author is not used to carry heavy objects)
4. Safety and durability – final model, including all parts: joints, sensors, motors and control blocks should be able to withstand some applied forces. And in case of an accidents that may, and most likely will occur, during the testing phase, be easily repaired with no damage to vital (and preferably all) parts. Also safety and durability is essential in case of work in an unknown environment.
5. Programmability / controllability – the controller part of the model should be reprogrammable, preferably in different programming languages and in a real time.
6. Availability – meaning that the chosen parts could be obtained within 2-3 weeks.

To find suitable materials many options were analyzed. Solutions that met most requirements were considered to be potential platforms for an “inverted pendulum” model. Among those solutions were:

- Arduino.
- PICAXE.
- Gumstix.
- Mindstorms NXT 1.0. set.

First thee solutions were excluded, because of various reasons – mainly because of the fact that the model in all those cases would be presumably build using metal materials and may include many screws soldering and maybe even welding (and in case of plastic alternative glue and rubber).

Based on a brief market research and the fact that the NXT 1.0 set was loaned to the author for research. The NXT 1.0 set was chosen as the most suitable platform for building an “inverted pendulum” model, because it meets most conditions:

- Relatively low cost: NB! Its cost will be included based on its market value of that time.
- No extra tools (bolts, nuts, screws, screwdrivers, etc.) are required to build this model
- All sensors are protected by plastic boxes.
- A suitable controller.
- A lot of parts to create different models.
- Contained several different sensors and motors. Namely 3 motors / rotation sensors, light, sound, ultra-sonic and touch sensors.
- There is build–in Bluetooth controller to connect it with other devices.
- Had a disk with an Integrated development environment designed for this set.
- The voltage level control is already programmed in the NXT controller brick meaning that at most cases there is no need to program voltage level separately. Only if the designed system requires other vice.

2.2 Specifications of the NXT 1.0 set. (code of the set 8527)

[10,11,12,13]

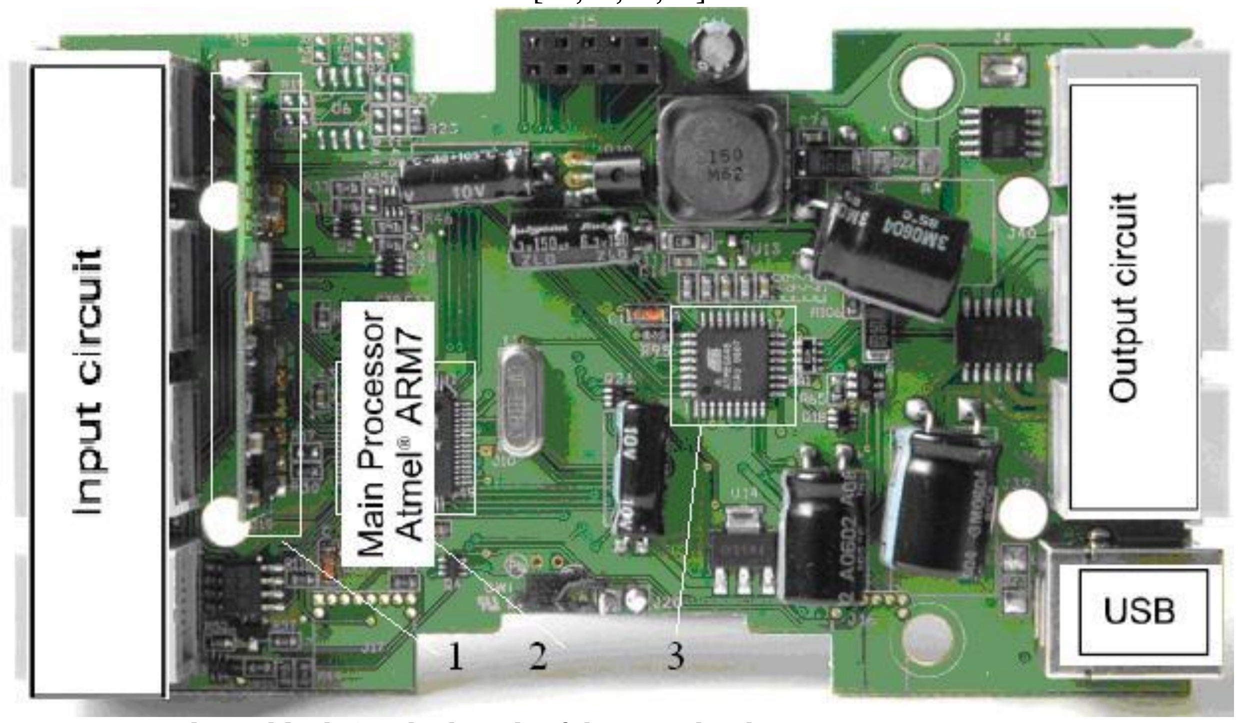


Figure 2.1 Hardware block (motherboard) of the NXT brick

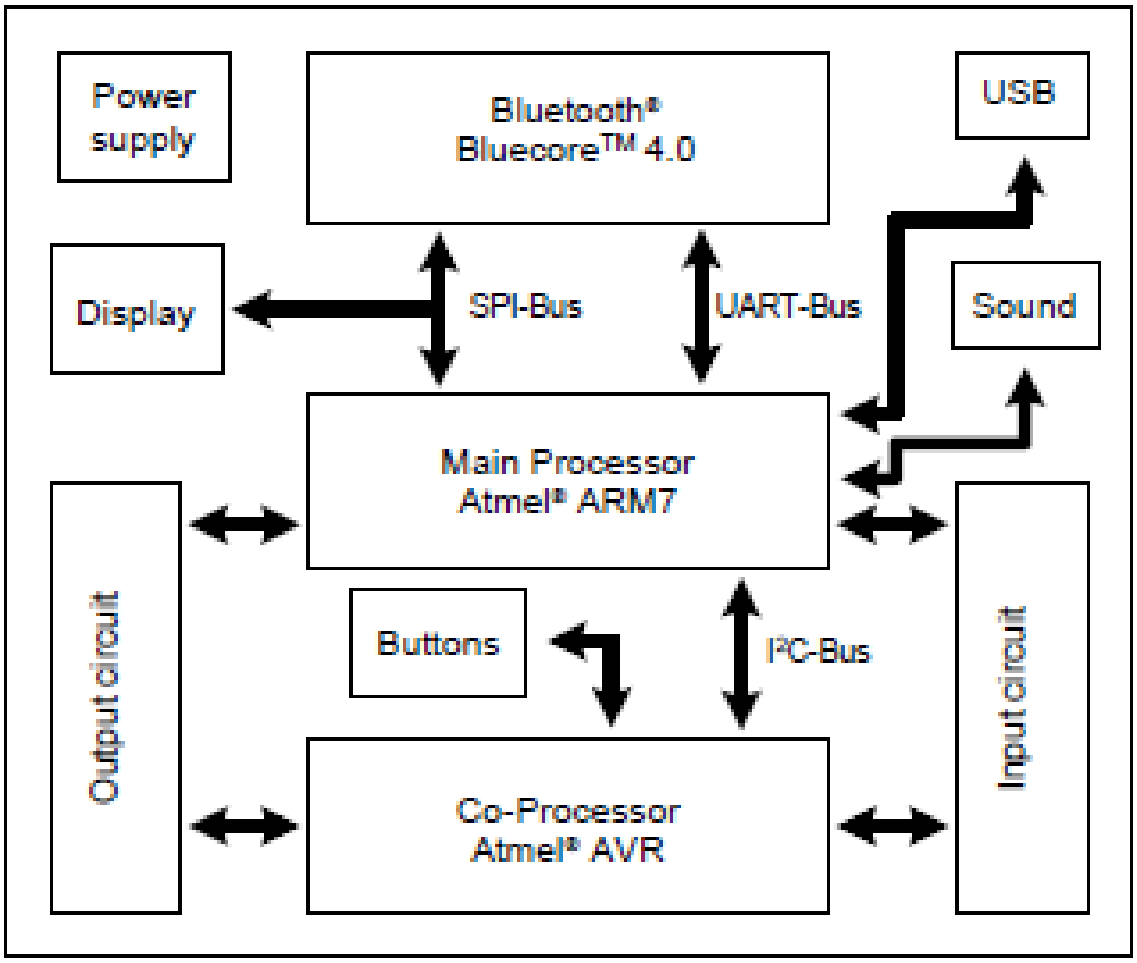


Figure 2.2 Hardware block diagram of the NXT brick

A summary list of hardware specifications for the NXT brick:

1. Bluetooth wireless communication CSR BlueCoreTM 4 v2.0 +EDR System (fig .(1))
    - Supporting the Serial Port Profile (SPP)
    - Internal 47 KByte RAM
    - External 8 MBit FLASH
    - 26 MHz
    - Arm 7 and Bluecore communication speed 460.8 K bit/s; data bits: 8 bits; 1 stop bit
2. Main processor: Atmel® 32-bit ARM® processor, AT91SAM7S256 (fig .(2))
    - 256 KB FLASH.
    - 64 KB RAM.
    - 48 MHz.
3. Co-processor: Atmel® 8-bit AVR processor, ATmega48(fig .(3))
    - 4 KB FLASH
    - 512 Byte RAM
    - 8 MHz.
4. USB 2.0 communication Full speed port (12 Mbit/s)
5. 4 input ports:
    - 6-wire interface supporting both digital and analog interface.
    - 1 high speed port, IEC 61158 Type 4/EN 50170 compliant
6. 3 output ports:
    - 6-wire interface supporting input from encoders

    6.1 Display:
    - 100 x 64 pixel LCD black & white graphical display
    - View area: 26 X 40.6 mm.
7. Loudspeaker Sound output channel with 8-bit resolution.
    - Supporting a sample rate of 2-16 KHz.
8. Button user-interface (Rubber buttons).
9. Power source 6 AA batteries.
    - Rechargeable Lithium-Ion battery 1400 mAH is available.
10. Connector 6-wire industry-standard connector, RJ12 Right side adjustment.

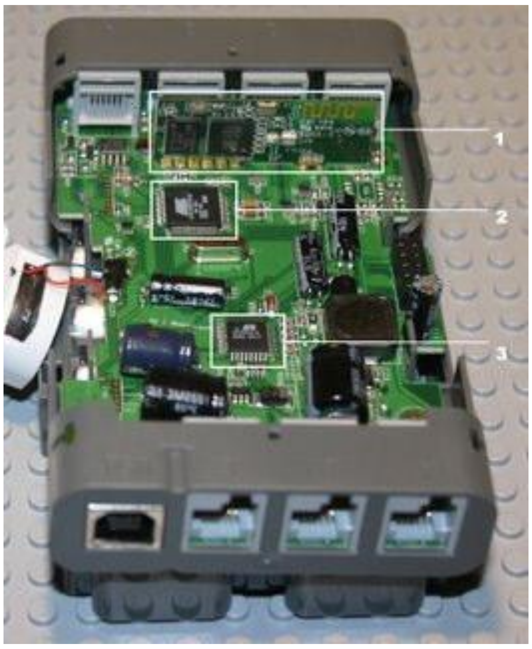


Figure 2.3 NXT controller

Now that the controller specifications are analyzed it is vital to analyze sensors, since they are main means of communication between the inverted pendulum model and real-time environment.

2.3 Light sensors specifications

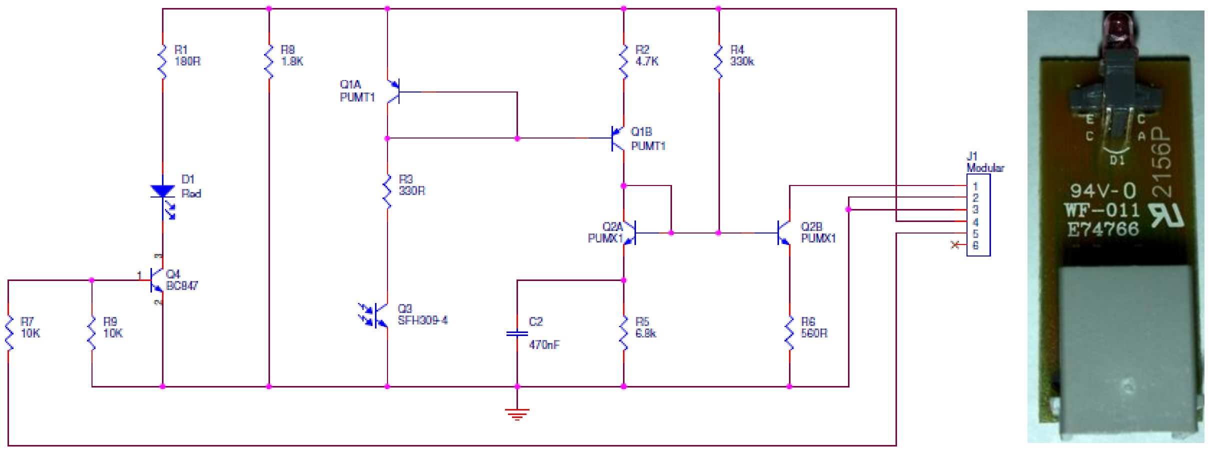


Figure 2.4 On the left: Electronic schematic of the NXT light sensor; on the right: Light sensor

The principle of this sensor is simple: The sensor reads the reflected light raw value and scales in on 0-100% scale. The led on a sensor may be lighted or not.

The phototransistor in the Light Sensor is far more sensitive to the infrared colors of light than the relatively narrow visible spectrum of an eye. Figure (number) shows how the transistor's spectral response overlaps the human eye.

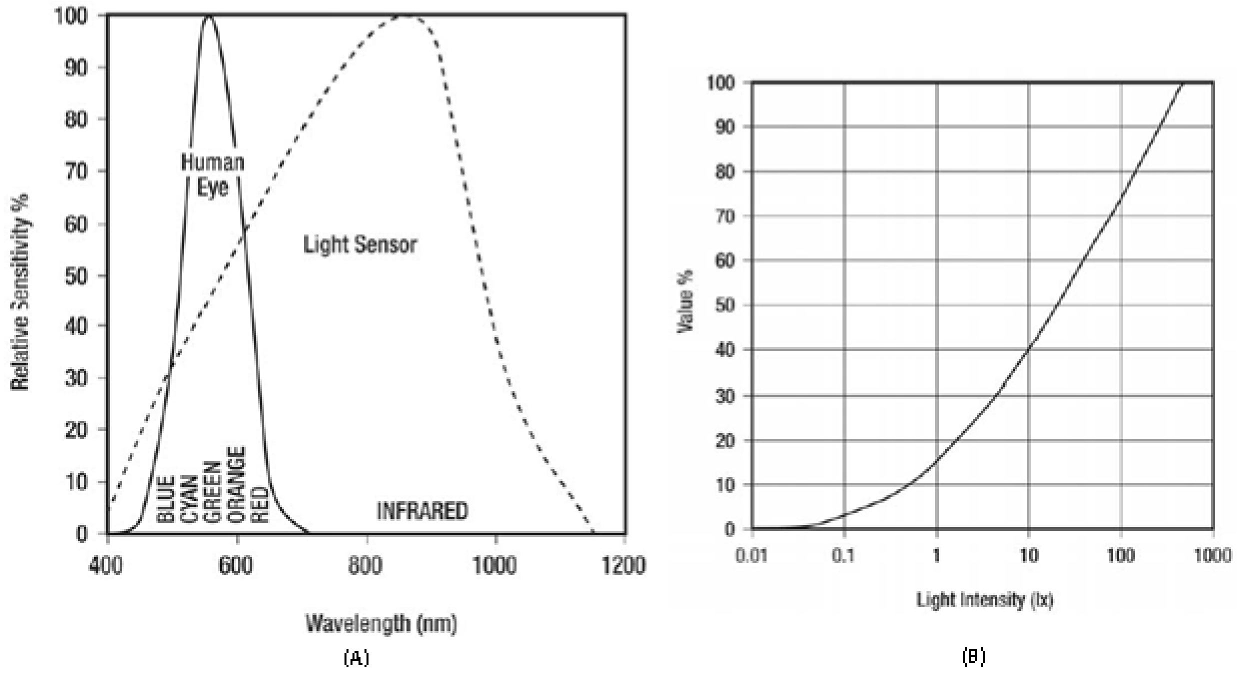
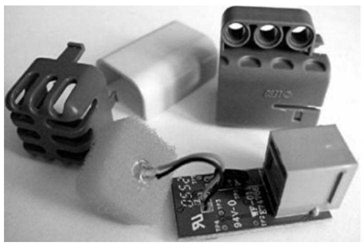


Figure 2.5 (A) Light Sensors spectral response. (B) Light Sensor sensitivity to light intensity.



2.4 Sound sensor specifications

Sensor measures sound pressure level based on two range settings:

- It can measure in dB
- It can measure in dBA

Figure 2.6 Disassembled sound sensor

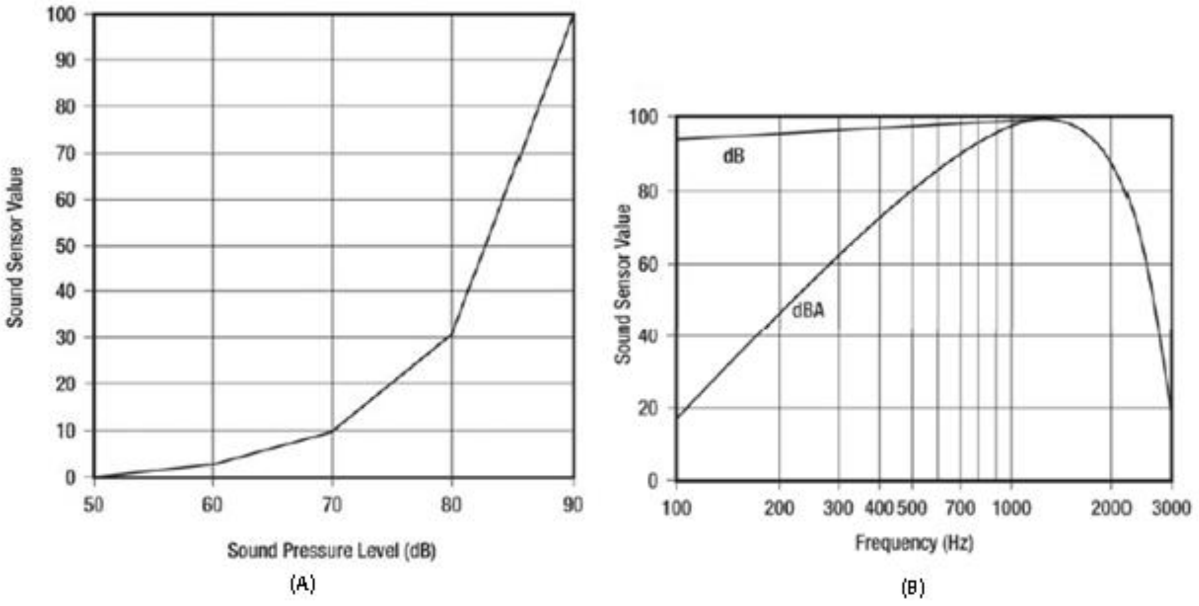
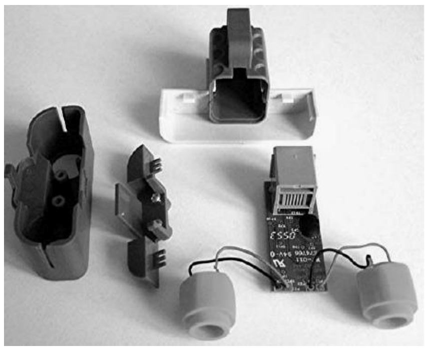


Figure 2.7 (A) Sound sensor value versus sound pressure level
Figure 2.7 (B) Approximate sound level value versus frequency

According to figures (number) – The Sound Sensor can measure sound pressure levels Up to 90 dB

2.5 Ultrasonic sensor specifications



- Measuring rate: 0 to 255 cm.
- Precision: +/- 3 cm.
- Ultrasonic frequency: 40kHz

Figure 2.8 Disassembled ultrasonic sensor

The sensor works like sonar by sending out a short burst of ultrasonic sound at 40 kHz then measures the time it takes for the sound to travel out to an object, reflect, and travel back. If there’s only one large object, the measurements are very reliable. However, if the scene becomes complicated, such as with many small objects, it isn’t as reliable. Also noises and low magnetic fields can disrupt and confuse the sensors measurements.

2.6 Motor / Rotation sensor specifications.

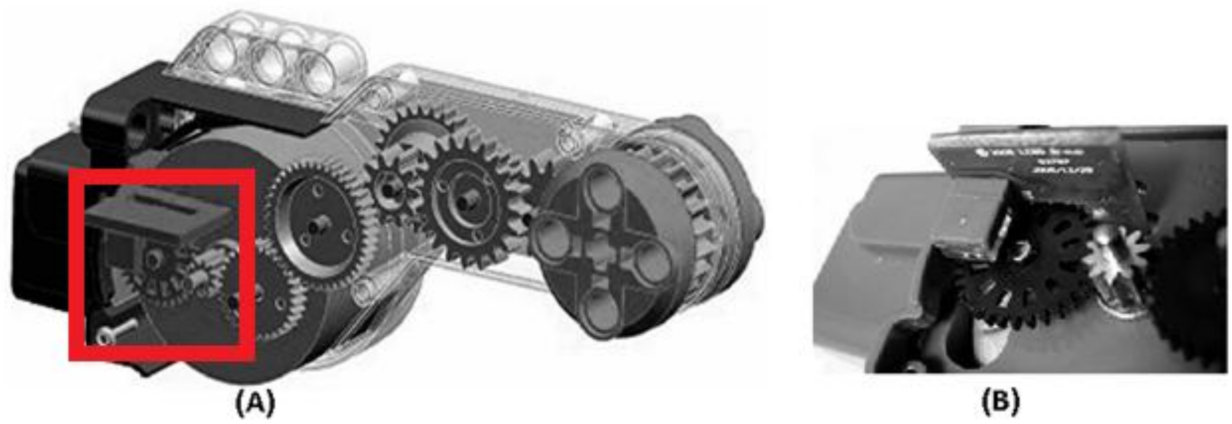


Figure 2.9 (A) Motor disassembly (B) Rotation Sensor

Direct Current (DC) motors almost always rotate too quickly to be connected directly to wheels and other loads. Some sort of gear train is necessary to lower the speed, which conveniently increases the torque. The NXT motors have built-in gear reduction The NXT motor is an impressive combination of gear reduction and feedback sensing (see Figure (number)). The gear reduction alone involves eight different gears, and because the Rotation Sensor is way back by the motor end, it has one degree of revolution on the output shaft. The speed of the motor is controlled by pulse width modulation (PWM). With the standard firmware, the length of the whole control cycle is 128μs. For the NXT, the relationship between the motor speed and the applied voltage is linear.

2.7 Ports and voltages.

Sensor inputs / outputs					
Pin	Color	Name	Usage	Voltage / Current	Sample time
1	White	AN	as an analog input	0 to 5V / 14 mA per input port.	3 - 4 milliseconds
			as a 9V power supply	0 to 9V	
2	Black	GND	ground – protection from short circuits: Usually pins 2 and 3 are connected together in sensors, If a sensor is accidentally connected to an output port, the driver will be partially short-circuited. Fortunately, the driver is well protected.		
3	Red	GND			
4	Green	4.3V Power	main power supply	0 to 4.3V / 25 mA to 180 mA	3 - 10 milliseconds
5	Yellow	DIGI0	I^2C Clock (SCL)	0 to 3.3V	Starting from 0.1 milliseconds
6	Blue	DIGI1	I^2C Data (SDA)		

Table 2.1 Sensor inputs and outputs

The sensor needs a capacitor to store power during the read interval. There is a current limit of approximately 14 mA per input port.

Motor inputs / outputs					
Pin	Color	Name	Usage	Valtages / Amps	Sample time
1	White	M1	provide power to the motor	0 to 9V Port A – 800 mA Port B and C – 500 mA	128μs
2	Black	M2			
3	Red	GND	As ground – indicates port as a motor		
4	Green	4.3V Power	connected to the 4.3V power supply that's shared between all the ports	0 to 4.3V / 25 mA to 180 mA	3 - 10 milliseconds
5	Yellow	TACHO0	inputs are used for the optical encoder built into the NXT motors, allowing the controller to determine the direction and rotations of the motor	0 to 3.3V	Starting from 0.1 milliseconds
6	Blue	TACHO1			

Table 2.2 Motor inputs / outputs

Capacitors and resistors		
Name	Used in	Capacity / Parts
pull-up resistor	AN pin	10 KΩ
4.7kΩ resistor	DIGI0 and DIGI1pins	4.7 KΩ
H-bridge	Motors	IC drivers LB1836M and LB1930M).

Table 2.3 Capacitors and resistors

2.8 List of possible problems and their possible solutions

As a result of this detailed overview some problems and malfunctions were discovered and presented in an organized list describing both the problem (number with a dot); source (dot) and possible solutions (number and brackets).

List of limitations and malfunctions that influenced the constructions and / or program changes to the "Inverted pendulum" model:

1. The display on the controller brick does not work.
- Source of the problem: "Unfortunately, some of the NXT Intelligent Bricks manufactured in-between 2006 and early 2008, may have a faulty display, resulting in a flickering or non-working display.

  Solution:

  1) To overcome (neglect) this malfunction it was decided to use NXT monitor emulators, because vast variety of IDEs has them

Other possible solutions:

  2) It was said, that "LEGO Corporate Management has decided that any NXT Intelligent Brick manufactured with this malfunction of the display will have an extended guarantee beyond the standard guarantee. The LEGO Group offers to replace NXT Intelligent Bricks with the above-mentioned faulty display. So it is possible to replace the faulty part via LEGO support.[14]
  3) It is also possible to fix this problem by correcting the malfunctioned part using the instructions in reference link [15]

2. The working powers of motors are different, which means that this could limit models movement speed and turning speed.

Solutions:

  1) Find a most suitable pare of motors to build the model with. Appendix 1
  2) Connect both motors with gears and connectors. But in this case the model will lose its ability to turn.
  3) Find a way to minimize this difference via programming. This can be done by using filters and regulators.

3. The efficiency of the motors strongly depends on the battery level.
- Source: all power to the sensors and motors is distributed via controller brick, which is powered by 6 AA batteries. Unfortunately, they tend to dry out.

  Solutions:

  1) Try to keep battery level higher than 6.5 volts, which should be enough.
  2) Limit tests to several minutes to save the battery (only if this has no influence on the test results).

4. The flash memory on the controller is 8 MBit and firmware and general programs take a large part of it.

  Solution

  1) Test one or two programs at the time and store tested files else ware.
  2) Find a way to expand the capacity of the initial storage device.

5. The latency (ping) of the Bluetooth connection. Transported data may be outdated (by a second or two)

Solution:

  1) Keep time sensitive algorithms directly on the controller.

Chapter 3 - Design and analysis of first Inverted pendulum model

3.1 General description

In this chapter, concepts of both the design and program code are given to better understand the modifications that are/ will be done to improve different aspects of the inverted pendulum model.

Components:

It was decided to use only sensors and parts included in the NXT 1.0 kit

- 3 motors / rotation sensors
- Light sensor
- Ultrasonic sensor (later was dimed impractical to use at this point)
- Sound sensor (later was dimed impractical to use at this point)

Programming:

Initially it was plant to use only MINDSTORM's program NXT-G that was included in the set. However, thanks to research done in Chapter 4, similar programs were also written in other IDEs.

IDE list:

- NXT-G – Programming the control (balancing) system and testing the motors.
- LabVIEW – Programming the control (balancing) system, testing the motors gathering of the data and remote control.
- RobotC – Programming the control (balancing) system.
- LEGO Digital Designer – for designing the inverted pendulum model.

First concept (real working model is not presented in this thesis)

The concept:

- 2 motor(s) control and prove moving force for the model
- Ultrasonic sensor acquires real time data.

Program: gets (writes) values of an ultrasonic sensor and calculates and corrects power of motors to maintain balance.

Stand up pendulum has a fixed distance to the ground (N) if the distance is more than the fixed one (F+ number), then the pendulum is falling backwards, if the distance is less than the fixed (F - number) then the pendulum is falling forwards and the motors should react accordingly to those changes and the pendulum should maintain balance.

This concept was not realized due to low accuracy of the ultrasonic sensor. It is +-3 centimeters. But the concept of the programming is partly used in the "Segway" concept model.

"Segway" concept model (light senor)

Design (shown on figure 3.1:

The design of this model is actually a modification of "segway with a rider" model. [14] The main modifications are:

- Sound sensor was included in the model.
- Sound sensor was placed instead of an ultrasonic sensor.
- Ultrasonic sensor was placed on the middle part of the model.

As a result of this the model in question is slightly (about 14 grams) heavier than the original model. But thanks to those changes the model is theoretically could be remotely controlled by voice commands and ultrasonic sensor has beater vision.

Chapter 3 - Design and analysis of first Inverted pendulum model

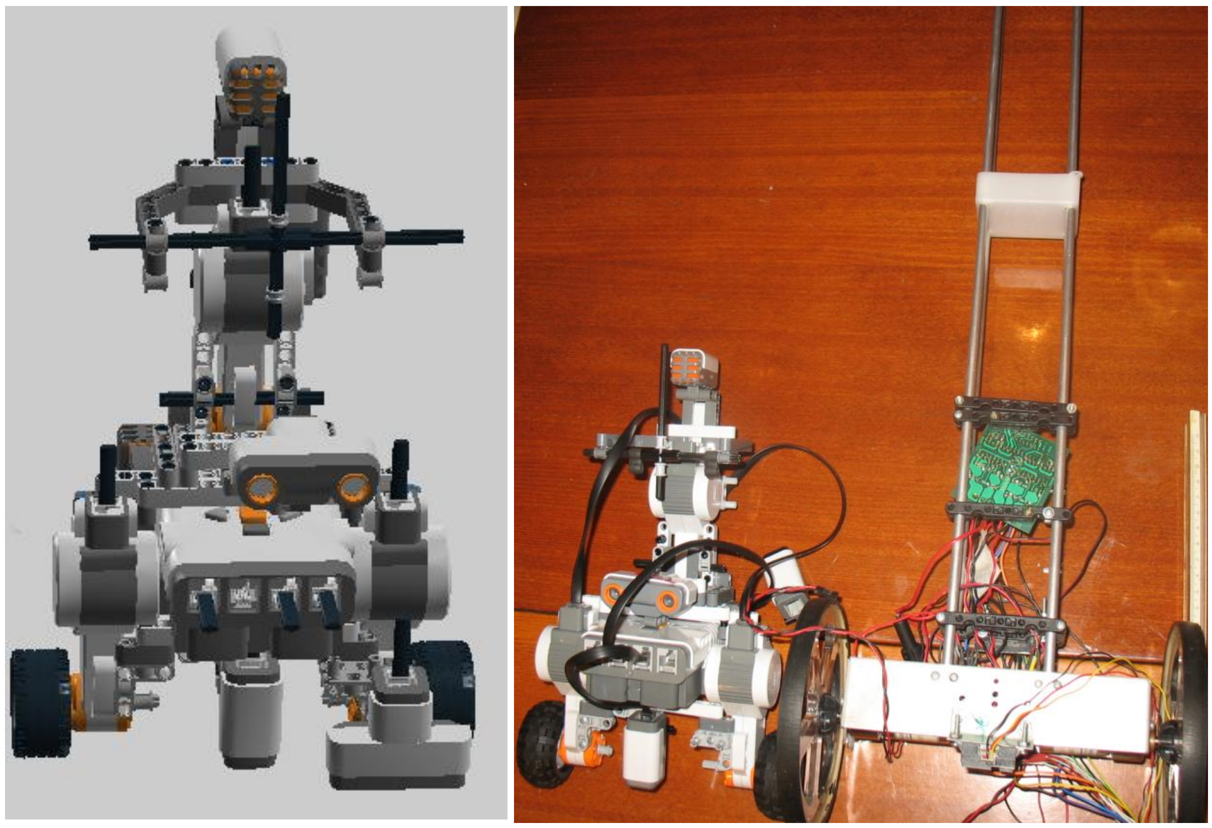


Figure 3.1 On the left: LDD design of Inverted pendulum; on the right of Inverted pendulum compared with other model.

Programming:

The principle of this program is similar to the first concept:
Initially there were only 4 steps and 2 conditions but some steps were added to provide some additional functions (problems and solutions are listed below).

1. The program starts.
2. You have 5 second to put the robot straight up. (was added to correct a problem)
3. At the start the light sensor measures the value of light reflected from the surface and saves it a control point.
4. Then the light sensor will measure values of the light sensor and compare them to the control value.

- If the light value is bigger than the control value – that means that the “Inverted pendulum” leaned forward.
- If the light value is lower than the control value – that means that the “Inverted pendulum” leaned backward.
- Emergency stop block will stop the motors if it is presumed that “Inverted pendulum” had fallen down or the angle can’t be corrected. (was added to correct a problem)

5. Program will calculate appropriate power and steering (vector) for motors by using PID controller.

Additionally sounds and display pictures were added to the code, to provide information and attract attention.

List of (faults) problems and solutions that were corrected by editing the program's code:
Problem

- The "Inverted pendulum" must stand perfectly still on its own before the start of the program and user has to push the button, which is situated on the NXT block, to start the program and that often destabilized the "Inverted pendulum". (i.e. outside force is applied to the "Inverted pendulum", before the controlling program is started.)

Solution

- The best solution is to upload the program directly from PC to the NXT for every test. That way the program will start automatically and users do not have to touch the buttons on NXT block.
- Give some additional time after the start of the program, but before measuring a set (control) point for the sensor. That will give time for users to ser the "Inverted pendulum" straight.

Problem

- If "Inverted pendulum" falls down motors start working at their maximum power and a fallen "Inverted pendulum" can still move relatively fast

Solution

- Emergency stop block – that stops motors if it is presumed that Inverted pendulum had fallen down or the angle can't be corrected.

List of problems that could not be corrected by editing the program's code:
Problem:

The surface, on which the "Inverted pendulum" is tested on, must have a uniform solid color and very uniform brightness with no pattern, because the light reflection will vary as the "Inverted pendulum" moves.

Solution

Do following tests on suitable surface (in this case light grey colored plank)

Problem :

The light sensor is very sensitive. Therefore, external light(s) source(s) or shadow(s) will confuse the light sensor, because lights and shadows will appear as the "Inverted pendulum" moves around.

Solutions:

- Use a location where light sensor will be in shadow from any lights, even as it moves.
- User(s) should try not to create shadows or always shield "Inverted pendulum" from any light source(s).
- Use a plaice with one light source that is directly above the Inverted pendulum or does not interfere with the light sensor.
- Perform experiments in a dark room with no lights.

Chapter 3 - Design and analysis of first Inverted pendulum model

The NXT-G Program

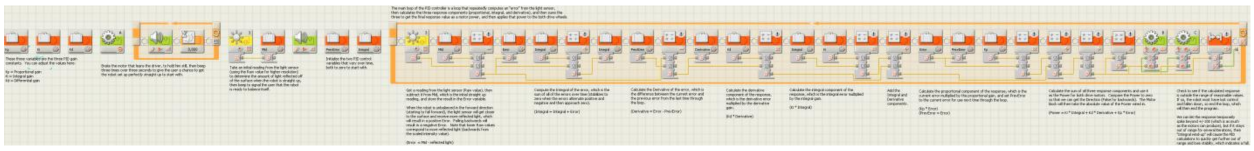


Figure 3.2 The entire NXT-G KL-WAY (light-Inverted pendulum program)

The program is described in Appendix 2.

3.2 The first test of the Inverted pendulum with a light sensor

<u>The main aim of this test</u> is to prove that a PID controller could control the designed Inverted pendulum model.

<u>Secondary aim</u>: is to find some possible problems

Conditions:

1. Program is done on NXT-G and recorded via text files / file access function.
2. Graphs are done in Excel.
3. Tests are done on a narrow grey board.
4. With no lights in a totally dark room.
5. Time of the test is 20 seconds. Due to the memory shortage. (see Chapter 2)

<u>Settings</u>:

Program KL-way.xre is used (appendix 2)

Sample time is set to 0.5 seconds

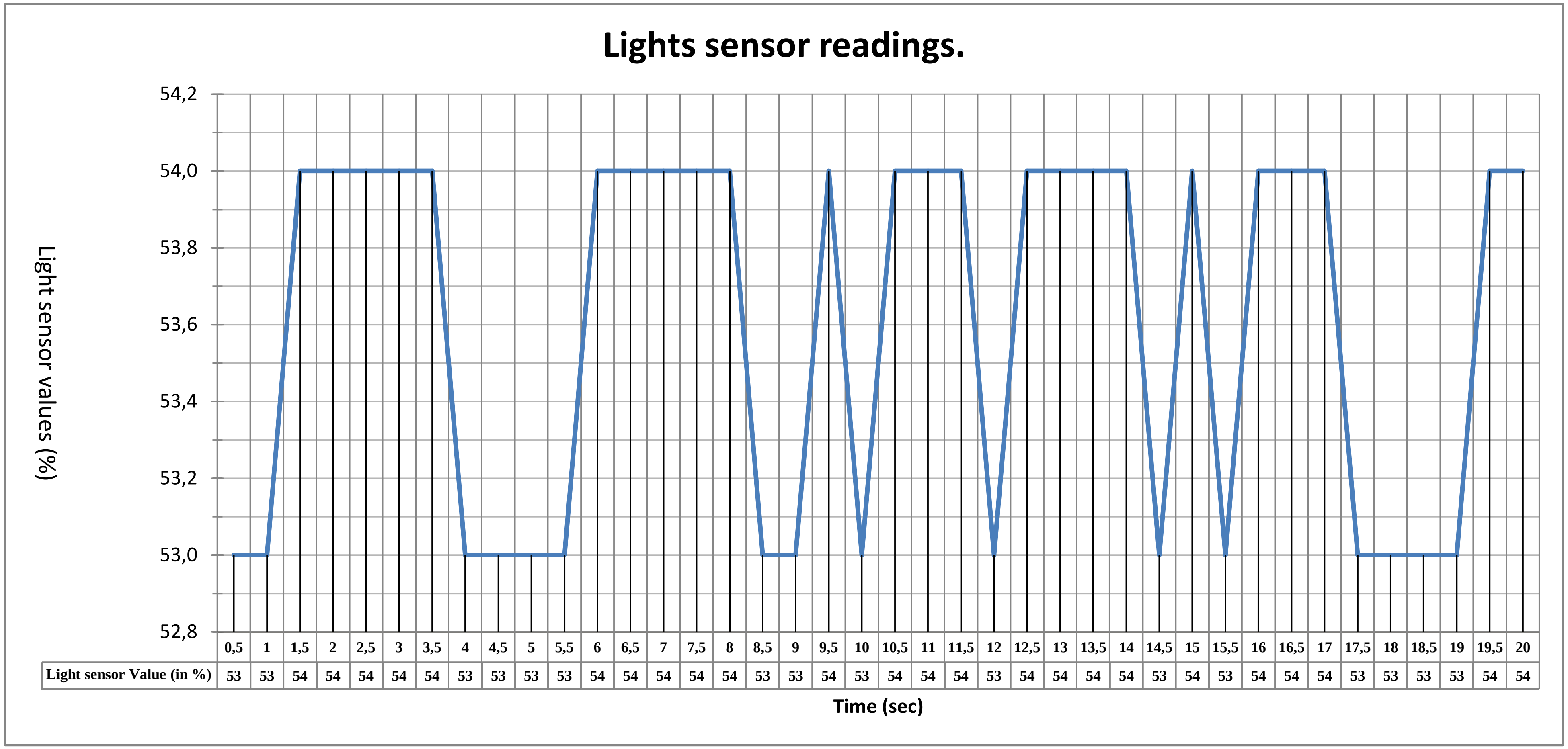
PID controller configurations:

- P = 40
- I=2
- D=10

Expected results:

The model will try to remain balanced and minimize its movements during the duration of the test.

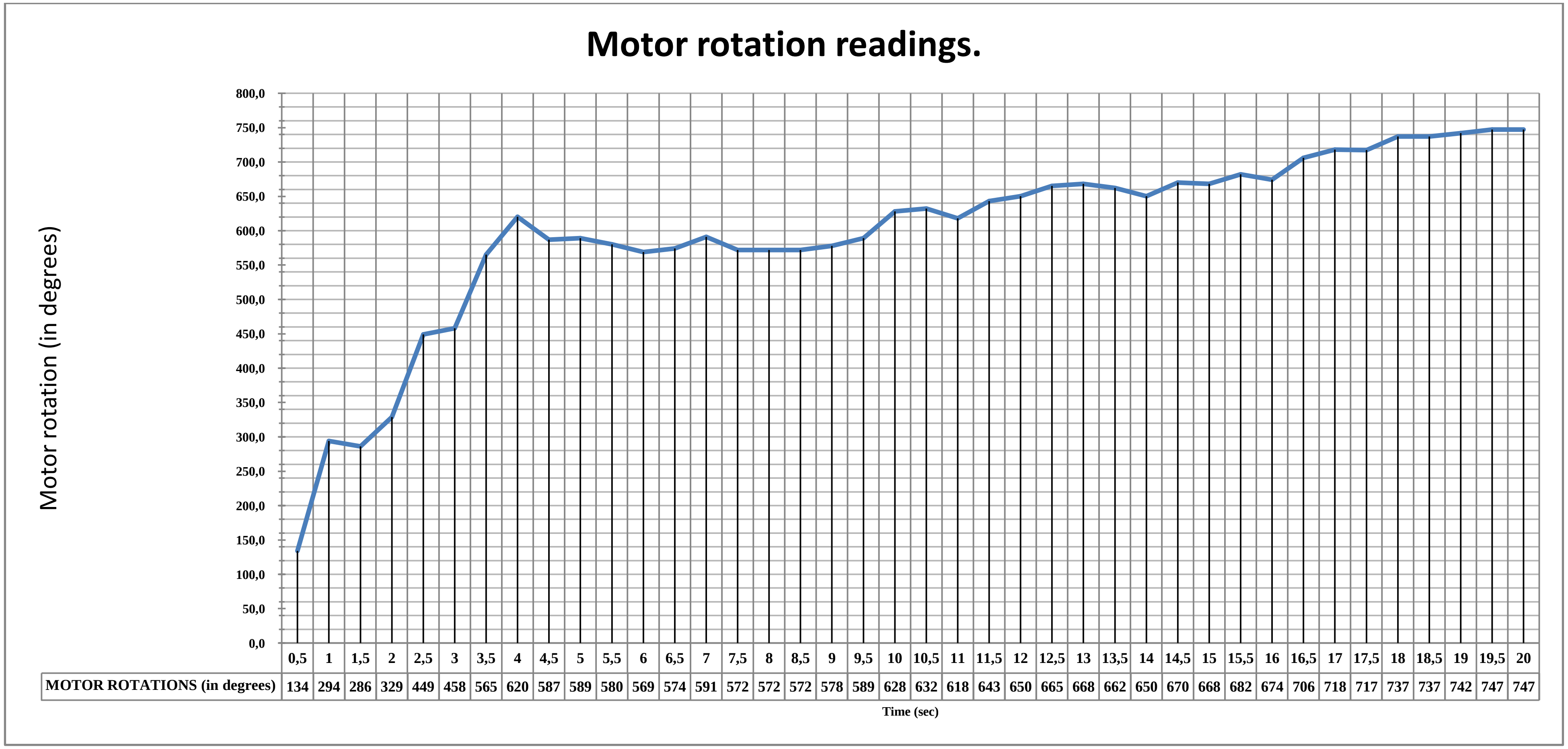
The results are shown on three graphs (figure 3.4 – 3.7)



Time (sec)	0,5	1	1,5	2	2,5	3	3,5	4	4,5	5	5,5	6	6,5	7	7,5	8	8,5	9	9,5	10	10,5	11	11,5	12	12,5	13	13,5	14	14,5	15	15,5	16	16,5	17	17,5	18	18,5	19	19,5	20
Light sensor Value (in %)	53	53	54	54	54	54	54	53	53	53	53	54	54	54	54	54	53	53	54	53	54	54	54	53	54	54	54	54	53	54	53	54	54	54	53	53	53	53	54	54

Figure 3.3 Lights sensor readings in live time

This graph shows Light sensor values over time in even presents but the reading to calculate PID controllers are coded in bits from 0 to 1023

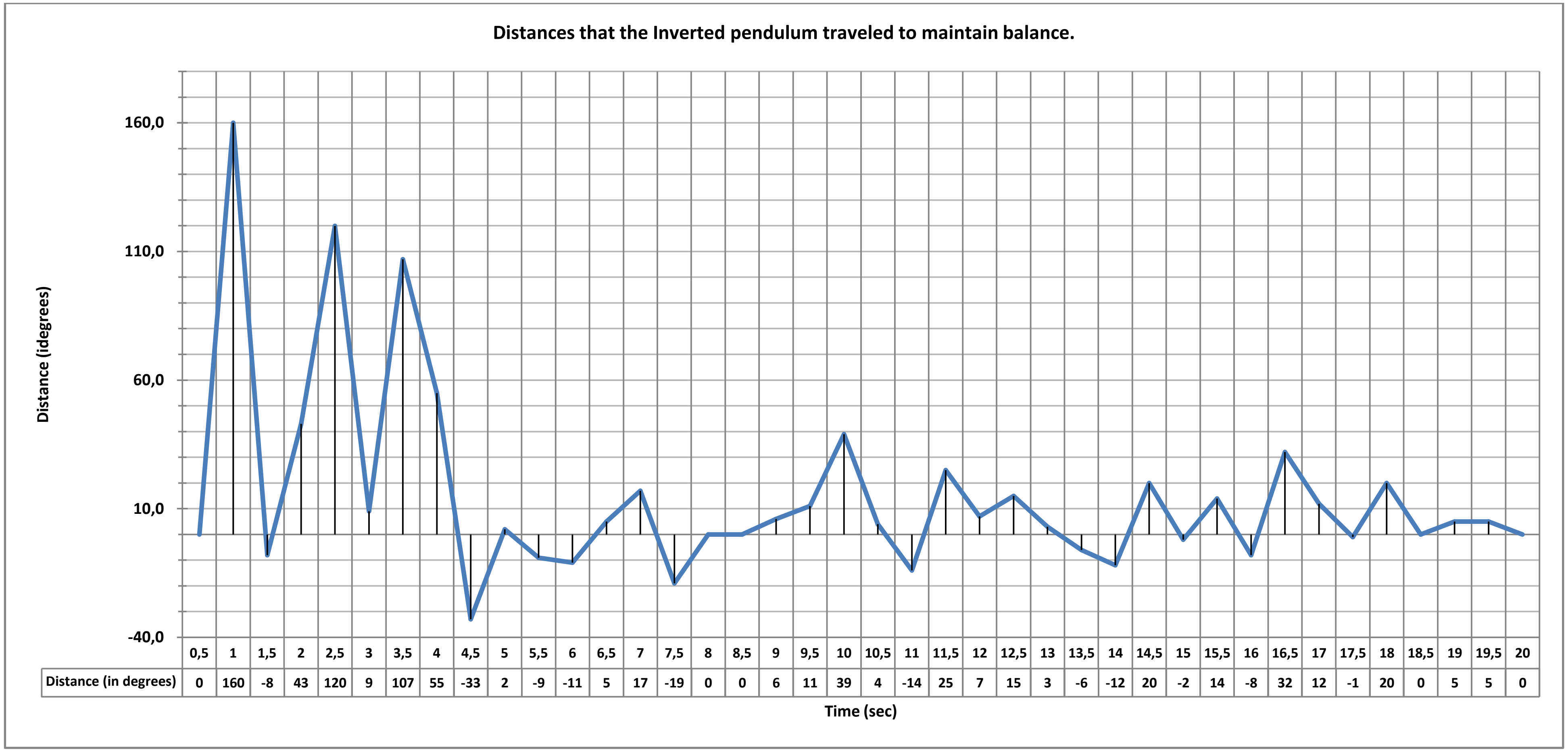


	0,5	1	1,5	2	2,5	3	3,5	4	4,5	5	5,5	6	6,5	7	7,5	8	8,5	9	9,5	10	10,5	11	11,5	12	12,5	13	13,5	14	14,5	15	15,5	16	16,5	17	17,5	18	18,5	19	19,5	20
MOTOR ROTATIONS (in degrees)	134	294	286	329	449	458	565	620	587	589	580	569	574	591	572	572	572	578	589	628	632	618	643	650	665	668	662	650	670	668	682	674	706	718	717	737	737	742	747	747

Figure 3.4 Motor rotation readings graph

This graph shows motors response to changes of light values recorded by light sensor.

Unfortunately, it is hard to see vectors of movement and distances it took to balance the inverted pendulum, but we can use acquired data to calculate distances (rotations) between measurements.



	0,5	1	1,5	2	2,5	3	3,5	4	4,5	5	5,5	6	6,5	7	7,5	8	8,5	9	9,5	10	10,5	11	11,5	12	12,5	13	13,5	14	14,5	15	15,5	16	16,5	17	17,5	18	18,5	19	19,5	20
Distance (in degrees)	0	160	-8	43	120	9	107	55	-33	2	-9	-11	5	17	-19	0	0	6	11	39	4	-14	25	7	15	3	-6	-12	20	-2	14	-8	32	12	-1	20	0	5	5	0

Figure**.** 3.5 Calculated distance graph

This graph illustrates distances that the inverted pendulum had to travel to maintain balance. Although far from perfect but this graph illustrates harmonic motion and damping.

Overview of the result (Conclusion)

As it can be seen from the graphs, the inverted model behaved as expected:

At first, the set value of the light sensor was recorded. Then the "Inverted pendulum" started balancing. PID controller and light sensor readings controlled power of the motors. The balance point was maintained after 20 seconds. Unfortunately, the light set point was set some ware between 53 – 54 percents. However, it could not be seen clearly on the graph, because it, for now, could not be recorded more specifically, sue to the fact that the controller could read raw value, but the sensor can only measure values in percentages.

Suggestions:

Find better setting of PID regulator, so that the "Inverted pendulum" could react to disturbances and balance more quickly. In addition, it is vital to find a way to obtain more accurate values from the light sensor.

To find better setting of PID regulator settings methods described in chapter 1. Data for Ziegler–Nichols tuning method is approximated

3.3 The second test of the light-Inverted pendulum

The main aim of this test is to prove that a PID controller could control the designed Inverted pendulum model.

Secondary aim: is to find some possible problems

Conditions:

1. Program is done on NXT-G and recorded via text files / file access function.
2. Graphs are done in Excel.
3. Tests are done on a narrow grey board.
4. With no lights in a totally dark room.
5. Time of the test is 20 seconds .Due to the memory shortage 3

Settings:

Program KL-way.xre is used (appendix 2)

Sample time is set to 0.01 seconds

PID controller configurations:

- P = 60
- I=3
- D=8

To compare all data (meaning all graphs will be presented in one figure). Rules to read those graphs are:

- Blue and green lines are measured in units, since program was now designed to use bit code to read raw values of the light sensor.
- Blue line is actual light sensor values (in units).
- Green line is a set point that is recorded before the balancing program is started and the pendulum is presumed to be in a stable state.
- Red line is actual rotation sensor values (in degrees)

The results of this test are presented on a figure 3.7

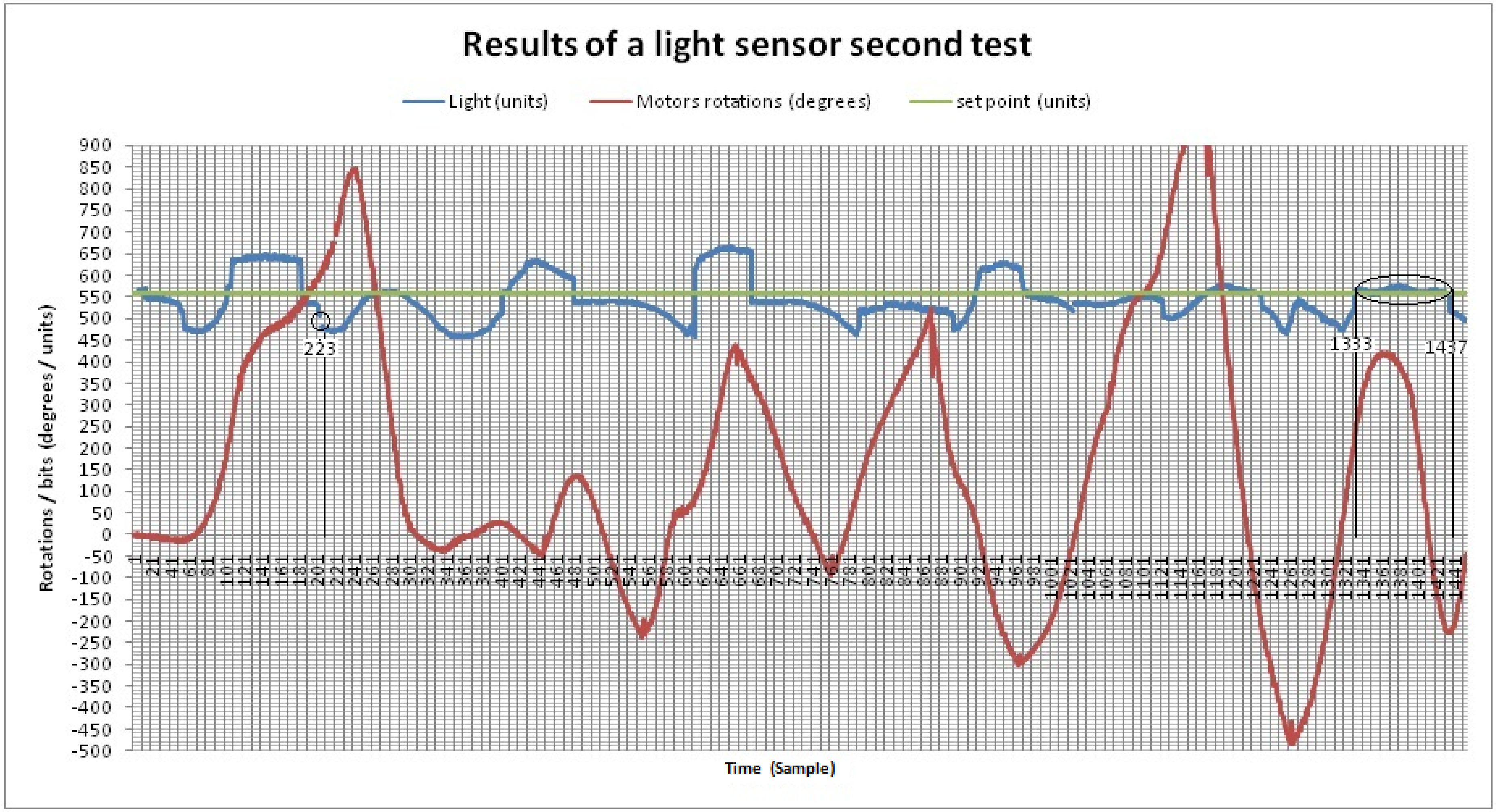


Figure 3.6 Combined results of the second test of the inverted pendulum with a light sensor.

Overview of the results

Main results of the tests are:

- It took the pendulum 13.33 second to balance.
- The pendulum could balance for 1.04 seconds.
- Some data points are lost (example at step 223)
- The recording of data stopped after 14.55 seconds.

Description:

As the inverted pendulum model started balancing it almost immediately lost balance (due to the fact that it was not set in a perfectly stable position, it took the pendulum 1333 steps = 13.33 seconds to balance itself and the pendulum could remain almost perfectly still for 104 steps = 1.04 seconds. The data recording stops almost immediately afterwards.

Additionaly, it could be perfectly seen from the graphs that the motors and light sensor are not synchronized, because the motors take some time to react to the light changes.

One other defect that could be seen from the graph (mainly during sample 223):
At some points data was not recorded to the text file. This is probably because it takes some time for the controller to open a file, then write a data to it and finally save it. And there are a total of three files to work with. At some point the data may become lost and erased on the next step before it could be recorded. This is probably due to the fact that the sensors value signal was recorded in a raw value instead of percentage, which may took more memory then in the previous test.

Problems:

Although the pendulum could balance itself at a relatively suitable time the program could not record the essential data shortly after the initial stabilization.

Solutions:

Find a more efficient way to record data from the controller and sensors to prolong testing time and accuracy.

Record the data in percentages but make the program read it as a raw value to ensure accuracy of the light sensor while balancing the inverted pendulum.

3.4 The third test of the Inverted pendulum coded in the NXT-G

To do this test it was designed to record data via Bluetooth and a program done in LabVIEW IDE. (described in chapter 4) and a balancing program done in NXT-G.

<u>The main aim of this test</u> is to improve PID controller so that the inverted pendulum could be more stable in wider corridor of disturbances, by this the entire model will be more mobile which is imperative for one of the goals of this thesis. Since it has already been proven that the light sensor is sensitive to the light of room's lamps, including the table lamp. It is logical to test if it is possible to control the movements of the inverted pendulum by adjusting the lights in the room during the test.

Conditions:

1. The control program is done on NXT-G KL-Way program
2. Data is recorded via program done in LabVIEW via live connection between NXT and a PC terminal.
3. Graphs are edited in Excel.
4. Tests are done on a narrow grey board.
5. With a table lamp lights in a totally dark room, but at the end of the test main room lamp will be turned on to see how the inverted pendulum will react to those changes.
6. Time of the test is virtually unlimited (unless the batterys dry out).

<u>Settings</u>:

1. Program KL-way.xre is used appendix 2
2. Sample time is set to 7 samples per second
3. PID controller configurations:

- P = 103
- I=2
- D=5

To review this test the entire test graph will be divided to main 3 parts to describe those situations.

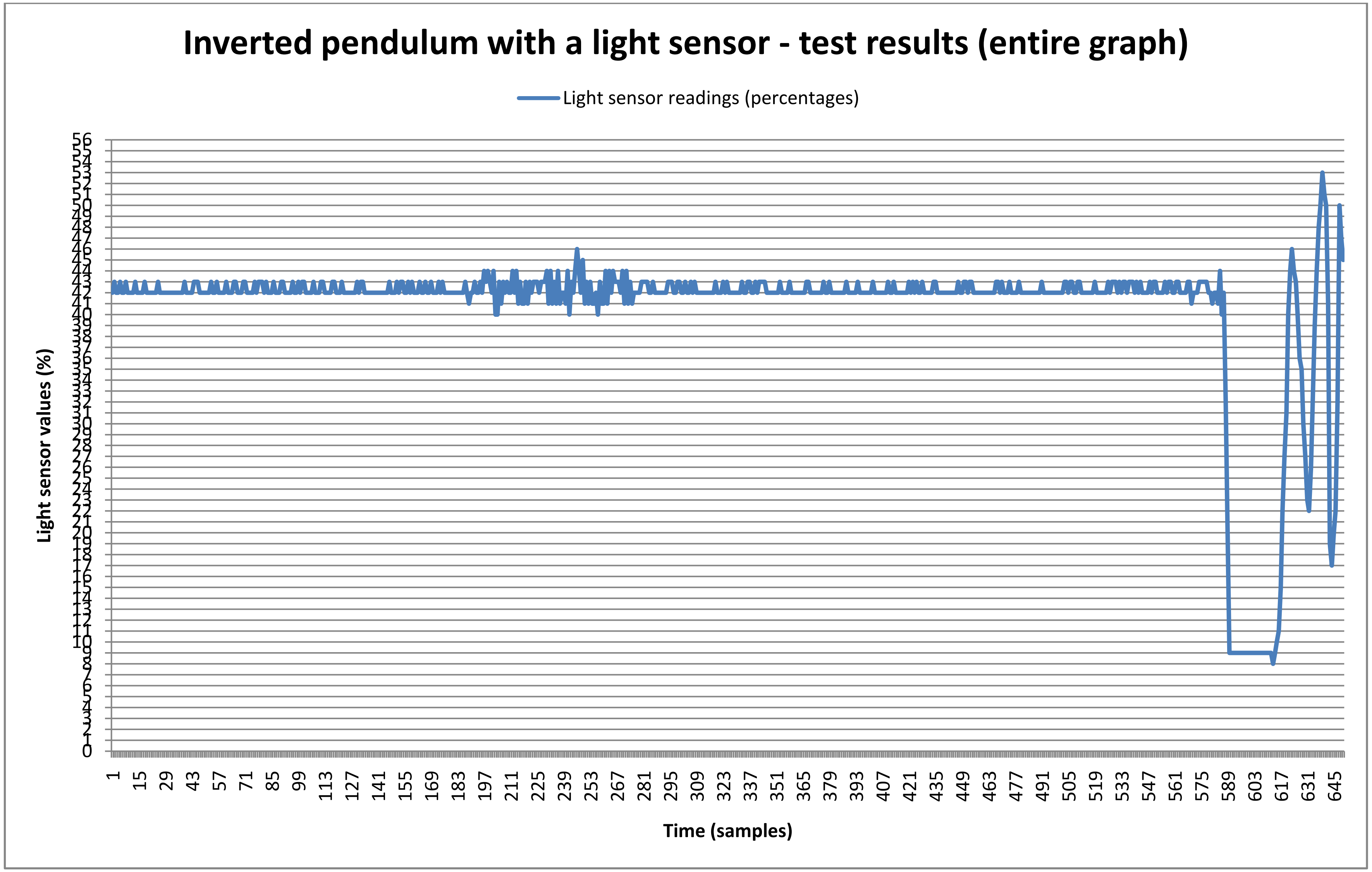


Figure 3.7 Light graph of the final inverted pendulum with a light sensor test.

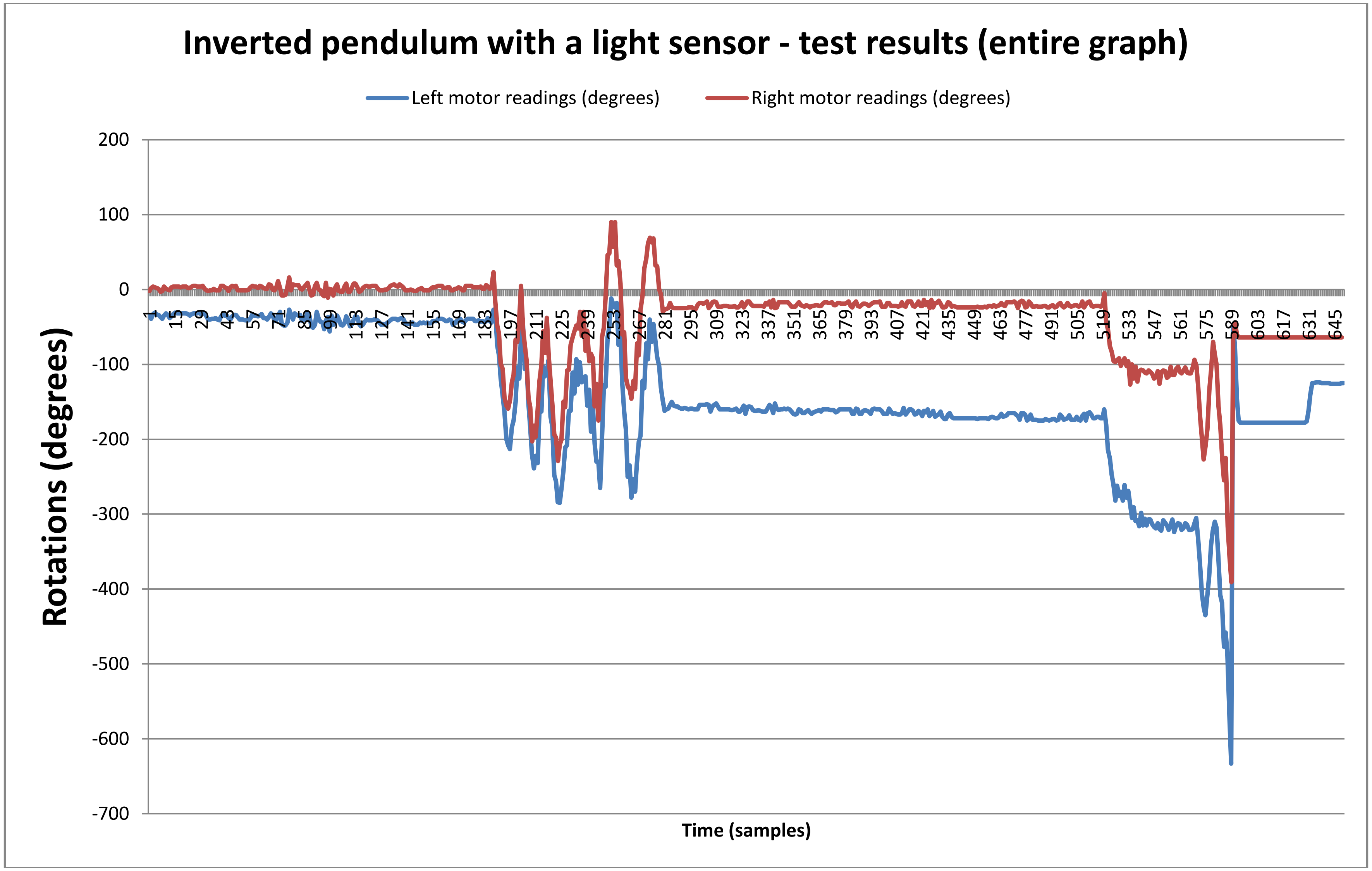
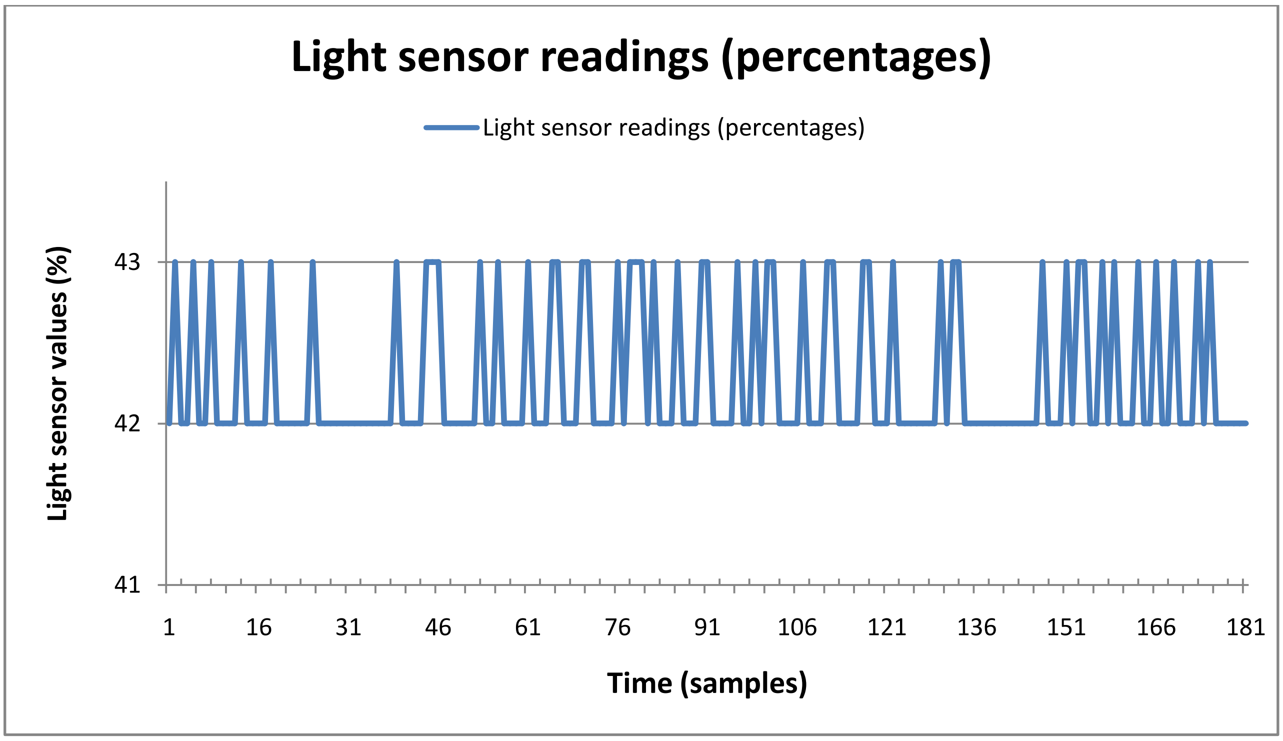


Figure 3.8 Motors readings graph of the final test of the inverted pendulum with a light sensor.



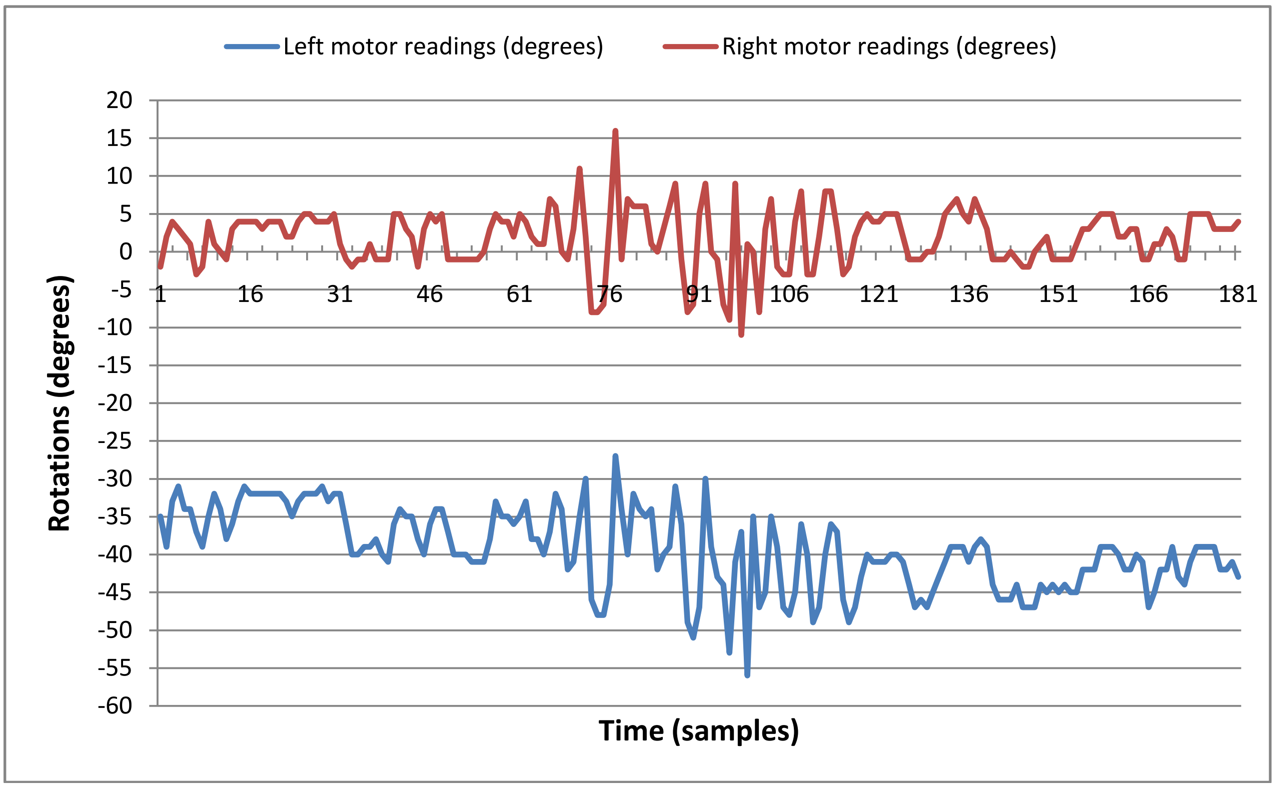
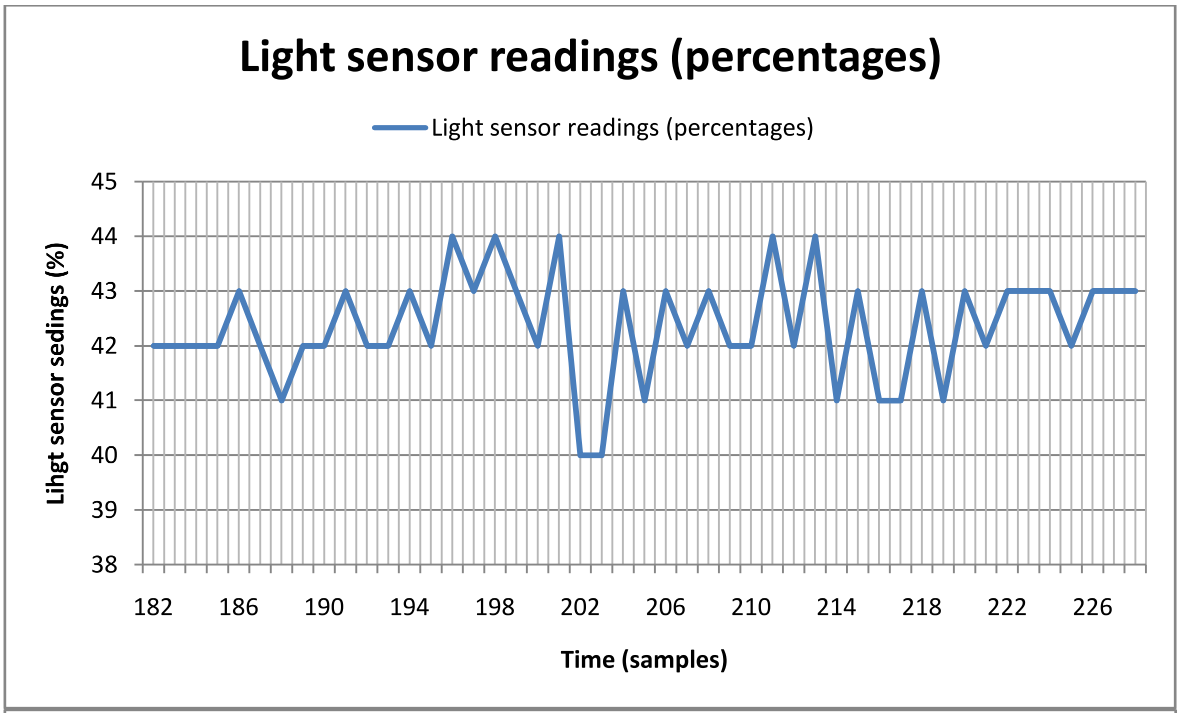


Figure 3.9 First part of the final test.

As it could be seen from those graphs a set point was set between 42 and43 %. The inverted pendulum could balance for 181 steps = 181/7 = 26.86 seconds. And the movements were minimal. $\dfrac{|-11| \ (\deg rees) \ + |16| \ (\deg rees)}{360 \ (\deg rees)} * 56 \ (mm) \ * \pi = 13{,}18 \, (\text{mm}) = 1{,}318 \, (\text{cm})$



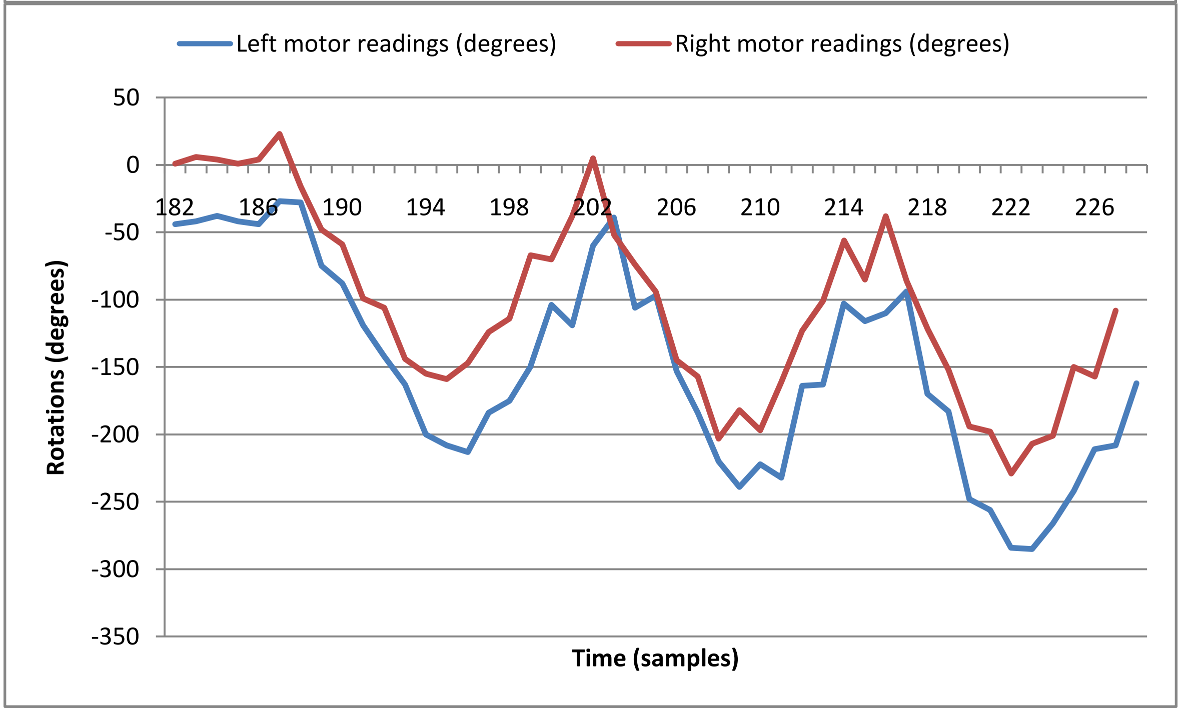


Figure 3.10 Second part of the final test.

At the 185 step some disturbance was given to the light sensor (a desk lamp was shaded for 1-3 seconds.

The inverted pendulum model could compensate this disturbance and return to its "corridor" in about 43 steps = 6.14 seconds. The maximal distance it took to balance was:

$$\frac{|-229|\ \ (\deg rees)\ \ +|23|\ \ (\deg rees)}{360\ \ (\deg rees)} * 56\ \ (mm)\ \ * \pi = 123{,}09\,(\mathrm{mm}) = 12{,}309\,(\mathrm{cm})$$

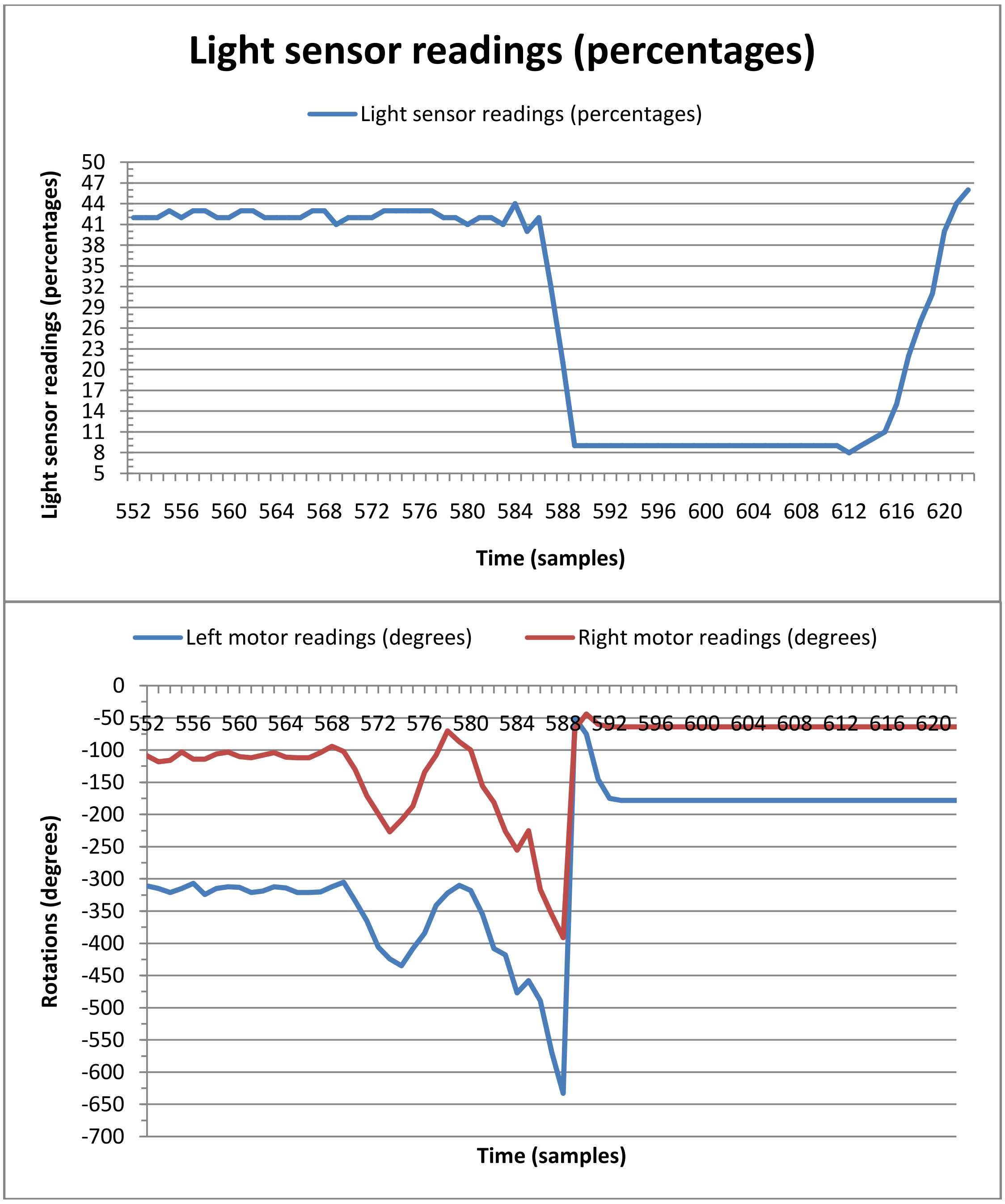


Figure 3.11 Third part of the final test.

After the inverted pendulum regain balance all lights were turned off at 586 step = 83.7 second after the start of the test. And the inverted pendulum could not handle such disturbance and feel down the lights were turned back on at the 628 samples = after 6 seconds.

Overview of the test

Main results of the tests are:

- It has been proven that the Inverted pendulum could be control via light disturbances.
- The pendulum could balance for more then 83 seconds.
- It took approximately 6.14 second for the pendulum to counteract the disturbance
- The recording of data stopped after 14.55 seconds.
- The balance "corridor" is about 1.5 – 2 percents (at those diference inverted pendulum can maintain a suitable balance.
- The duration of the test recording excited 92 seconds

Description:

As the inverted pendulum model started balancing it could maintain a suitable balance; and maintain it until light disturbances were applied to it The control algorithm could counteract some disturbances and used about 6 seconds to return the inverted pendulum in to desired value "corridor". The control algorithm could not counteract most drastic illumination changes. (disturbance). The starting positions of the motors are different, but they are controlled via PID algorithm inside the NXT-G firmware.

Problems:

The designed inverted pendulum model still cannot be used as a mobile multifunctional platform mainly because of the fact that this model cannot counteract light and surface changes in a real time environment.

Solutions:

Find a more reliable concept of the inverted pendulum model – this may require modification of not only programming but the hardware (materials) as well.

Chapter 4 – Modifications
4.1 Short description

To solve problems that were found in the previous chapter and improve functionality of the model it was decided to find alternative IDE in which the NXT controller could be coded and analyzed.

There are more than 60 IDEs that can compile a code for NXT and supports most functions. Although there were total of 10 IDEs that were tested during the research phase. Only few of them will be presented in this thesis.

One of the initial goals of this thesis is to build a multi functional platform that could be used in academic studies and teaching. That goal in the perspective of coding means that this platform should have an ability to be coded by different IDEs in different coding languages.

The main problem of most IDEs is that each IDE requires its own firmware, and using different IDE usually means formatting of the controller's internal memory. To bypass this problem a custom firmware was set on the NXT 1.0 controller and PC.

Firmware.

Generally, every IDE checks the version of the NXT controller firmware before connecting to it and some even would not compile a code of a program before without a proper firmware.

To customize firmware different files of several firmwares were merged in to one set of files. Because most IDEs check so called "check sum" / "hash". The firmware was customized in a way that it shows different version markers to different IDEs this way there is no need to reinstall firmware every time that different IDE is used to control the NXT controller. (Note: That it is beater to use default firmware(s) if you want to edit or replicate any code presented in this thesis.) Other positive thing of manipulations with firmware is the fact that now several simultaneously working IDEs may send and collect information to/from the NXT controller in a real time, and generally there is no need to pre-code the communication algorithm on the NXT. There are certain rules and limitations (for example: NXT-G needs to be connected before LabVIEW if they are working on the same port USB or Bluetooth at the same time).

Changes in IDEs

Used IDEs were chosen based on several factors:

1. Operating system should be only so called "cross-platform" that will include most popular platforms used in Academic researches and studies.
2. IDE should have an remote display option to neglect the faulty display (chapter 2)
3. Different IDEs should have different abilities that could be used with one another.
4. Preferably, each IDE should present a coding / compile option on different coding language than others.

List of chosen IDEs:

1. Lego NXT-G 1.0
2. Lego NXT-G 2.0
3. LabVIEW for Lego Mindstorms toolbox (LVLM) and (FTC toolbox)
4. LabVIEW FIRST Tech Challenge toolbox (FTC toolbox)
5. RobotC
6. RobotC for Virtual Worlds
7. Lejos
8. App inventor

Chapter 4 – Modifications

To improve performance of every IDE and their positional features the IDEs were modified the following way:

- Lego NXT-G 1.0 and NXT-G 2.0 were combined via file manipulations to get better performance.
- The two toolboxes LVLM and FTC for LabVIEW were combined: to save time on switching between the two environments.
- RobotC and RobotC for Virtual World were combined. The version of the program is slightly different from the standard one (although the number is the same); reasons for this change are due to the fact that the author was a participant in testing this IDE for stability (to receive this IDE and some benefits.
- Lejos was integrated with app inventor and some other programs.
- Smart-phone was set to communicate with a NXT controller via a Bluetooth connection and simultaneously connect to other terminals (mainly PC) via internet connection if this option is available.

4.2 Descriptions of IDEs

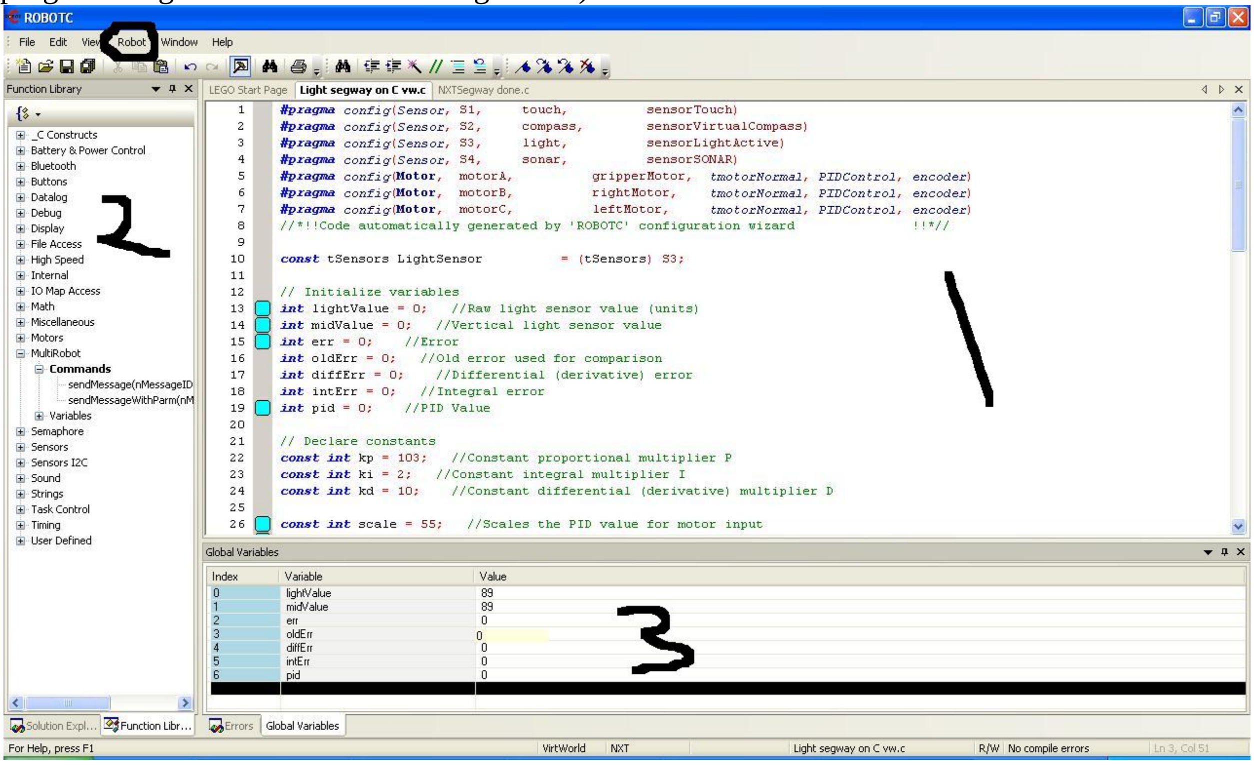
RobotC [17]

RobotC is an Integrated development environment targeted towards students, that is used to program and control LEGO NXT, VEX, and RCX robots using a programming language based on the C programming language.

It aims to allow code to be ported from one robotics platform to another with little or no changes in the code.

1. Has a debugging feature. (figure (3))
2. Has a big command library (figure (2))
3. Can target many robots figure (0-highlighted)
4. Uses text-based commands, commonly contrasted from NXT-G, a graphical language also for the NXT.
5. Has several tutorials designed to help the user learn the programming language.
6. Is intended for simple to advanced programs.
7. Has a "natural language" fetcher
8. Requires specific firmware in order to upload the program to NXT.

The IDE itself resembles a DEV C++ environment and is very user-friendly to both those who are un-experienced users and those who have some experience in programming. (The programming window is shown on figure 4.1)



```
#pragma config(Sensor, S1,     touch,          sensorTouch)
#pragma config(Sensor, S2,     compass,        sensorVirtualCompass)
#pragma config(Sensor, S3,     light,          sensorLightActive)
#pragma config(Sensor, S4,     sonar,          sensorSONAR)
#pragma config(Motor,  motorA,          gripperMotor,  tmotorNormal, PIDControl, encoder)
#pragma config(Motor,  motorB,          rightMotor,    tmotorNormal, PIDControl, encoder)
#pragma config(Motor,  motorC,          leftMotor,     tmotorNormal, PIDControl, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

const tSensors LightSensor         = (tSensors) S3;

// Initialize variables
int lightValue = 0;   //Raw light sensor value (units)
int midValue = 0;   //Vertical light sensor value
int err = 0;    //Error
int oldErr = 0;   //Old error used for comparison
int diffErr = 0;    //Differential (derivative) error
int intErr = 0;   //Integral error
int pid = 0;    //PID Value

// Declare constants
const int kp = 103;   //Constant proportional multiplier P
const int ki = 2;   //Constant integral multiplier I
const int kd = 10;    //Constant differential (derivative) multiplier D

const int scale = 55;   //Scales the PID value for motor input
```

Index	Variable	Value
0	lightValue	89
1	midValue	89
2	err	0
3	oldErr	0
4	diffErr	0
5	intErr	0
6	pid	0

Figure 4.1 – RobotC / RobotC for virtual worlds programming window with library and debugger.

To prove that a same program could be written in different languages a program was written to do the same algorithm as the program written in NXT-G. (shown in Appendix 3)

Chapter 4 – Modifications

RobotC for Virtual Worlds [18]

The Robot Virtual World software was developed using unity, a “Game Development” software and ROBOTC robot programming software. The Robot Virtual World software has two large tasks: drawing the 3D graphics in real time and simulating the physics of the virtual environment.

The IDE for coding is the same as in RobotC the only difference is that The RobotC for Virtual World allows to compile program to a virtual robot instead of a real one. The advantages of this environment are:

- There is no need to build real world robots to test some (simple) programs.
- It is possible to see how the sensors could work, without the need of having a real sensor in a set. It is very useful especially if the price of sensors is high and there is no assurance that the sensor in question is suitable for the desired model.
- Virtual Worlds contain simulation of a real world table that is used for FTC competitions.

The reason for not using RobotC for Virtual Worlds to simulate the inverted pendulum model is because the limitation of the IDEs engine. The RVW does not have and does not support models for testing balancing algorithms. Furthermore, RVW does not allow the user to build own (custom) models only modify existing ones. Testing had shown that performances of the initiated program in RVW for several times were worst then the performance of the same program on a real – world model. (Example: anti-collision wall detecting program made the robot hit the wall several times in situations where that should not happen).

NI LabVIEW for LEGO® MINDSTORMS® [19]

LabVIEW for LEGO MINDSTORMS was designed specifically for use with the LEGO Education robotics platform, providing a sophisticated teaching tool that helps students program the LEGO MINDSTORMS NXT brick.

Specifications:

- Works with LabVIEW *LabVIEW 7.1.x, 8.0, 8.2, 8.5.x, 8.6.x*
- Works better if used with LabVIEW 2010 SP1 *or older)
- Program and control NXT devices with the full power of LabVIEW
- Get real-time updates from the NXT device during program operation with LabVIEW front panels
- Create native blocks for LEGO MINDSTORMS NXT software
- Direct connection to the NXT terminal
- Shows battery level in volts.
- Has many applications like piano player, sensor viewer, and remote NXT display.

NI LabVIEW FTC 2011-2012 Toolkit [20]

The LabVIEW for FTC Toolkit contains VIs that are specifically designed to help program a robot for an FTC competition. Use the LabVIEW for FTC Toolkit with LabVIEW for LEGO® MINDSTORMS® provided in the FTC 2011-2012 Kit to program the NXT brick to control FTC-specific sensors and motors.

The LabVIEW for FTC Toolkit includes the following features:

- The LabVIEW for FTC Toolkit supports the Samantha Wi-Fi Module, which the FTC 2011-2012 competition requires all users to connect. Refer to the FIRST website to install and setup a wireless connection with the Samantha Wi-Fi Module.
- The FTC Tools VIs palette contains nine VIs that are either new or altered from previous versions of the LabVIEW for FTC Toolkit. In the LabVIEW NXT Module.

- The Remote Control Editor contains a Generate Code button, which creates LabVIEW code based on the configurations you set for one or two game controllers. You can modify this base LabVIEW code as necessary for your specific robot.
- The Run Program page of the Remote Control Editor contains two modes for testing the game controller configurations: Prototype and FTC Game. Prototype mode allows you to test the configurations using the game controller. FTC Game mode allows you to test the configurations with programs that you can download to the NXT brick or programs that already exist on the NXT brick so you can use the game controller as you would in an FTC match. The Enable and Disable buttons allow you to switch between Teleop mode and Autonomous mode respectively.
- Contains more advanced data analyzer.

Usage and tools:

- The usage process of those programs are basically the same as a regular LabView.
- The tools to connect interact with NXT are very comfortable in use.
- The tools and additional applications of those toolboxes enable usage of many potential benefits of the NXT .

App Inventor [21]

App inventor is an open-source web application originally provided by Google, and now maintained by the Massachusetts Institute of Technology.
App Inventor includes:

- A designer, in which a program's components are specified. This includes visible components, such as buttons and images, which are placed on a simulated screen, and non-visible components, such as sensors and web connections.
- A blocks editor, in which the program's logic is created.
- A compiler based on the Kawa language framework and Kawa's dialect of the Scheme programming language, developed by Per Bothner and distributed as part of the GNU operating system by the Free Software Foundation.
- An app for real-time debugging on a connected Android device.

4.3 Multi-programming, recoding

To improve capability of the standard NXT controller, it was modified. List of both standard functions and limitations and a list of modifications is presented.

<u>Limitations of a standard NXT 1.0 block:</u>

- Only one program can run on the NXT controller at the same time.
- The NXT controller could be connected to one terminal via USB connector and no more than tree terminals via a Bluetooth connector.
- Initially the NXT controller can connect to several devices, but work with only one device at the time (read / write data)
- The USB port could be connected only to one device or IDE (program) at the same time.
- The NXT controller can not be connected to the Ethernet or other networks that do not use Bluetooth connection protocol.
- Potential multitasking needs to be coded in one program file.
- Every IDE need its own firmware to communicate with the NXT controller.
- Lack of memory that can be used with an NXT controller.
- The robot could use four sensors and three motors / sensors at the time.

<u>Properties and modifications that were achieved to ensure and expand multi tasking and multiprogramming in a real time environment:</u>

- Several (potentially more than ten) programs can run simultaneously on different IDEs.
- The NXT controller could be connected to one terminal via USB connector.

Chapter 4 – Modifications

- The NXT controller could be connected to six devices via Bluetooth direct conection
- One of the devices connected to the NXT can ask other devices to serve as a router or a hub to connect to more then six devices.
- The NXT can read and write to several devices: one device at a time and store vital information in a backup file before its transmission.
- The NXT controller could use Ethernet if one of the connected devices has a direct access to the network.
- The NXT can recognize a variety of sensors and devices by asking and receiving their name.
- The NXT could use sensors of the connected devices.
- Several tasks can be coded as separate and sometimes independent programs / applications and can be executed via different terminal in a real time. (example: balancing is done via a program executed directly on the NXT controller, but collision avoidance and data recordings are done via program executed on PC and / or smart-phone.
- One or several terminals can serve as a storage unit for NXT controller files (excluding the programs that are executed at this moment) so that the memory capacity is now only depends on the capacity of the connected terminals, which can be over 100 (hundreds) of Gigabytes.

Design options

It was clear, from the tests done in chapter 3, that the Inverted pendulum model needed a better sensor to use as a main sensor for balancing. To solve this problem additional research was done to find most suitable set of sensors.

It was possible to use two sharp Infra-Red (IR) range sensor sensors (to measure distance from two different points of the robot to the surface. Then compare the readings to set points and use them to calculate power of the robot’s motors.

The main problem was to find a way to connect “Sharp” sensors to the NXT controller. The NXT and sharp sensors have different connection ports (reference 19). Then find a way to code them in the same IDE as other parts of the model.

Possible solutions:

- Connect the sensor directly to the NXT controller using data in chapter 2.
- That method requires to use information written in a report of the similar attempt done in 2008-2010 years. [10]
- Use a connector (coding chip) to transfer one connector to another.

Problems:

- That method requires additional (hardware) part and specific changes in a NXT controller structure.
- Changing electrical schemas requires steady hands, knowledge of electrical laws, some extra parts like conductors, resistors, etc.
- Voltages need to be manually coded both for the NXT and sensors.
- Sensor needs to be coded and decode f that the NXT controller could understand them.
- Any mistake during hardware connection and/or coding may resolve in critical damage both to the sensor and NXT controller.

Better version of the previous construction can be constructed by using High Precision Medium Range Infrared distance sensor for NXT (DIST-Nx-Medium-v3) [NXT sharp] instead of “Sharp” sensors. Those sensors are specially designed and coded for NXT set, so they have [22]

- Same connectors as the NXT controller,
- Voltage and signals are calibrated for the NXT controller.
- Detect obstacles between 10cm to 80cm

- Provides reading directly in millimeters
- Uses Sharp GP2Y0A21YK sensor
- Uses I2C bus digital interface to connect to NXT
- Maximum current consumption: 38mA at 4.7V
- This sensor has built-in calibrations to provide high resolution readings. When not in use, the sensor can be powered down to conserve battery.

It was also possible to buy another light sensor, connect two sensors simultaneously, and use the sum value of the light sensors to correct one another. (However, it could also cause more problems: i.e. two sensors are out of synchronization: which reading is correct.
That also meant that at least two sensors are used to balance; the initial goal was to build a multi functional robot and there are only 4 sensors (totally) that could be use simultaneously. So It was decided to find a way to control the balancing of a robot with only one sensor

After the, above mentioned, research it was confirmed that an angular or gyroscopic sensor could be used on the NXT 1.0. controller. There were total of three sensors that could be coded in IDEs used in this thesis.

1. Gyro sensors used in most modern smart-phones.
2. dIMU(Inertial Motion Sensor). [23]

- The Dexter Industries dIMU Sensor reads acceleration, tilt, and speed of rotation. This sensor combines a gyroscope and accelerometer into one sensor. Both the accelerometer and the gyroscope read on all three axes (x, y, and z). Build robots that know which way is up, can measure tilt, balance themselves, and measure acceleration and rotation on all axes all at once.
- The accelerometer works on three different levels and can measure ±2g, ±4g, and ±8g. The accelerometer can be calibrated for static acceleration of gravity.
- The gyroscope works on three different levels of precision as well, and can measure ±250, ±500, and ±2,000 degrees per second. The gyroscope has zero offset and very low drift, making it very suitable for inexperienced users.
- The sensor is supported in NXT-G, RobotC, NXC, and LabVIEW for LEGO MINDSTORMS (LVLM)

3. HiTechnic NXT Gyro Sensor for LEGO Mindstorms NXT. [24]

- The NXT Gyro Sensor contains a single axis gyroscopic sensor that detects rotation and returns a value representing the number of degrees per second of rotation.
- The Gyro Sensor can measure up to +/- 360° per second of rotation.
- The Gyro Sensor connects to an NXT sensor port using a standard NXT wire and utilizes the analog sensor interface. The rotation rate can be read up to approximately 300 times per second.
- The Gyro Sensor is housed in a standard Mindstorms sensor housing to match the other Mindstorms elements.

Conclusion:

It was decided to use a HiTechnic Gyro Sensor for constructing a mobile inverted pendulum. Among other advantages the Gyro sensor weights same as standard NXT sensor (12g) and could be used instead of any other sensor in the construction without the need of recalculating the weight of an entire robot. This sensor is lighter and smaller then dIMU or a smart phone. There is no need to charge two batteries like in case of using gyroscope in a smart-phone.

One of the reasons that this sensor was chosen a good price offer and bonuses that were given for purchasing this sensor (also two more sensors were bought to save money).

4.4 Optimal Design

To maximize Inverted pendulum model functionality the design of the model was also reconfigured. The design was heavily influenced by models constructed earlier and especially by the model published by Laurens [24] and HiTechnic [25] programming code was also partly used. The inverted pendulum model is presenter on figure 4.2

Capabilities of the model:

1. Compact.
2. Easy to modify and rebuild.
3. Suitable payload: 6 sensors and place to carry a smart-phone or some baggage.
4. Has an adjustable ultrasonic sensor that could have multiple purposes.
5. In case of an accident can be rebuild (fixed) in a matter of minutes
6. Uses three sensor ports for gyro, ultrasonic and light sensors leaving one port to be used for other sensors, that may be needed for solving different problems.

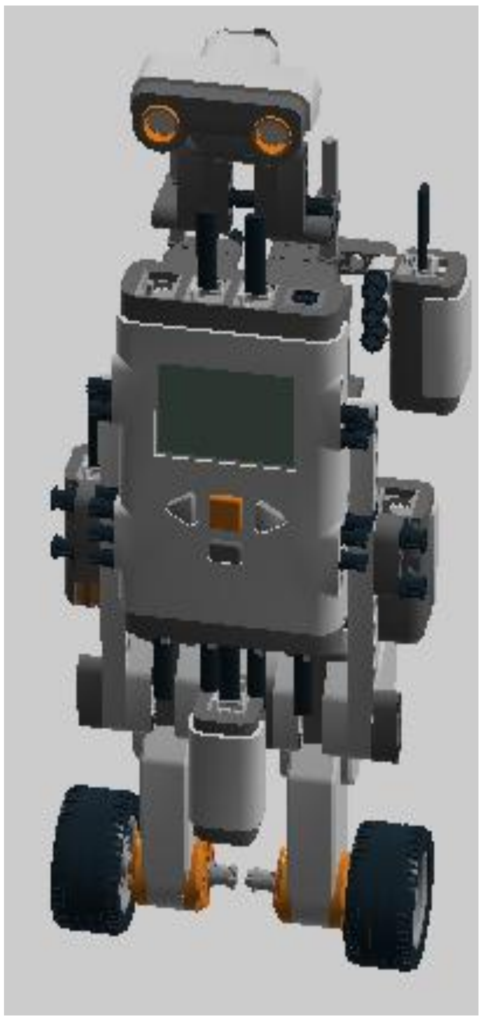


Figure 4.2 design of an inverted pendulum in LDD.

Chapter 5 – Tests of a multifunctional inverted pendulum model

5.1 Description

In this chapter various test are done to prove that the inverted pendulum model (later referred to as "robot") can handle multiple tasks at the same time.

Components:

- 2 motors / rotation sensors
- Light sensor
- Ultrasonic sensor (later was dimed impractical to use at this point)
- Gyro sensor (later was dimed impractical to use at this point)
- Compass, angular, sound sensors may be used.

Programming:

The program controls the inverted pendulum with gyroscope sensor than uses ultrasonic sensor for object detection and collision avoidance.

The same program was done in 3 IDEs:

1. NXT-G 2.0
2. RobotC
3. LabVIEW

The balancing control is virtually the same in all three programs.

The code enables the robot to balance itself. The steps of the algorithm:

1. The program is uploaded to the NXT controller and starts.
2. Robot informs the user that it will get an offset of the gyroscope and recommends the user to put the robot on a flat surface.
3. the gyro offset is displayed on the screen.
4. Enter button is pushed.
5. The user is asked to select the variations of parts used in robot.
6. The robot plays a sound for 5 seconds to give time to user for putting it in a upright position.
7. The robot reads the value of the gyroscopic sensor and controls the output of the motors to balance itself.
8. Robot moves forward at a fixed speed.
9. If the ultrasonic sensor detects an obstacle in close proximity (30 cm) the robot will backup and turn away from the obstacle to avoid it

Note: First 5 steps of this algorithm may be preprogrammed and skipped. Although this may limit the robots reliability and make it harder for the user to correct errors and noises (debug).

The code for this program is shown on figure 5.1.

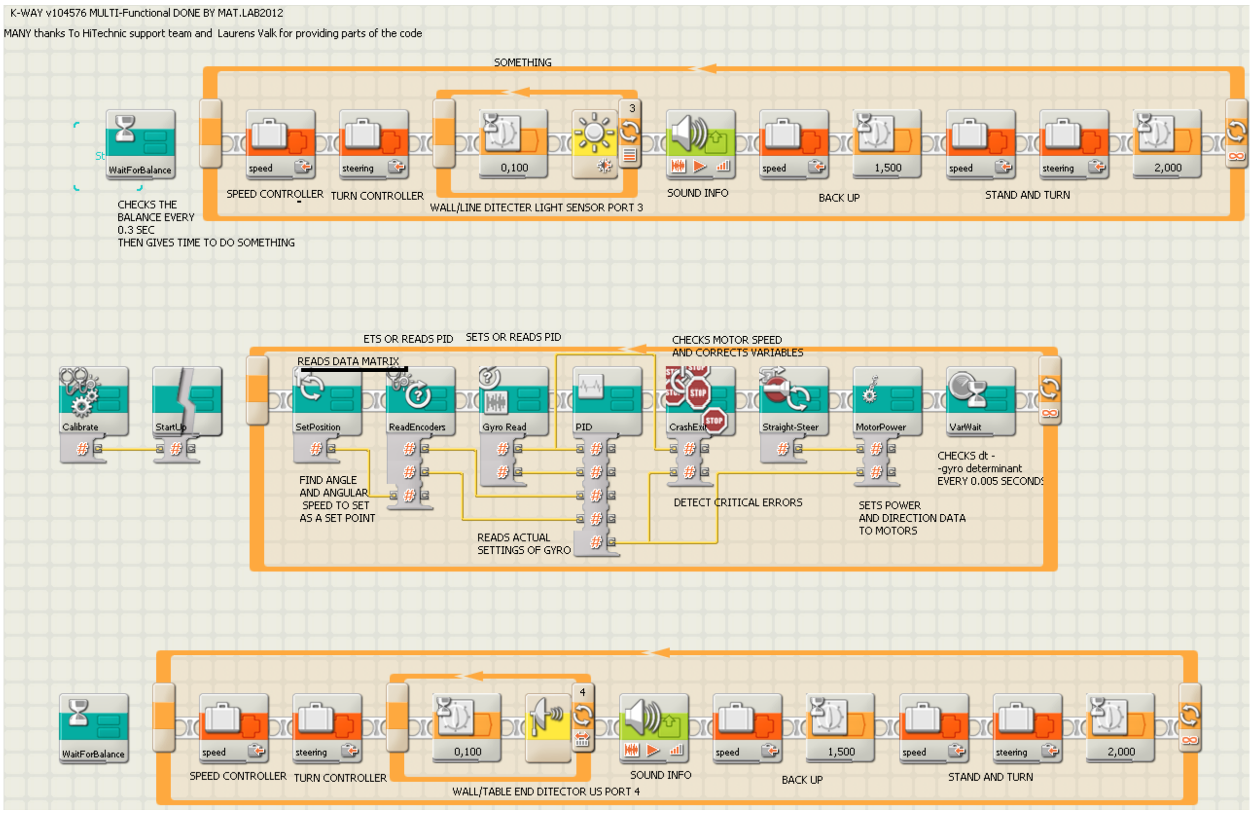

K-WAY v104576 MULTI-Functional DONE BY MAT.LAB2012
MANY thanks To HiTechnic support team and Laurens Valk for providing parts of the code
SOMETHING
WaitForBalance
speed
steering
0,100
3
speed
1,500
speed
steering
2,000
SPEED CONTROLLER
TURN CONTROLLER
WALL/LINE DITECTER LIGHT SENSOR PORT 3
SOUND INFO
BACK UP
STAND AND TURN
CHECKS THE BALANCE EVERY 0.3 SEC THEN GIVES TIME TO DO SOMETHING
ETS OR READS PID
SETS OR READS PID
CHECKS MOTOR SPEED AND CORRECTS VARIABLES
READS DATA MATRIX
Calibrate
StartUp
SetPosition
ReadEncoders
Gyro Read
PID
CrashExit
Straight-Steer
MotorPower
VarWait
CHECKS dt - -gyro determinant EVERY 0.005 SECONDS
FIND ANGLE AND ANGULAR SPEED TO SET AS A SET POINT
READS ACTUAL SETTINGS OF GYRO
DETECT CRITICAL ERRORS
SETS POWER AND DIRECTION DATA TO MOTORS
WaitForBalance
speed
steering
0,100
4
speed
1,500
speed
steering
2,000
SPEED CONTROLLER
TURN CONTROLLER
SOUND INFO
BACK UP
STAND AND TURN
WALL/TABLE END DITECTOR US PORT 4

Figure 5.1 K-WAY V104576 program on NXT-G

The main program is in the middle and is constructed using “My-blocks”

Lower block was added for Multi tasking test: obstacle detection

Upper block was added for Multi tasking test: floor detection

5.2 The offset of the HiTechnic Gyroscope sensor.

To maintain balance of the inverted pendulum it was vital to celebrate the Gyroscope sensor.

Conditions:

To test its accuracy the values of the sensors were measured for 10 seconds.
The results are shown on figure 5.2

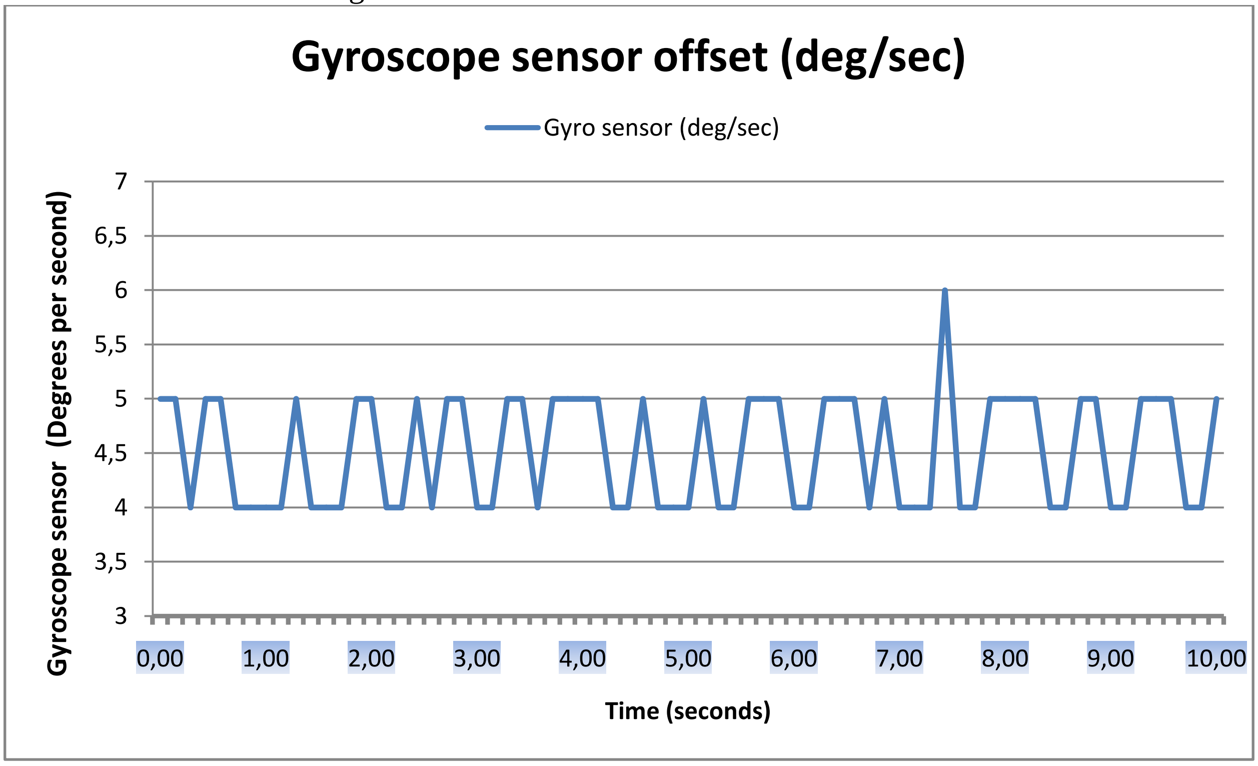


Figure 5.2 An offset of a gyroscopic sensor.

Overview of the result

As it seen from the graph the Gyroscope sensor maintain values between 4 and 5 degrees per seconds most of the time the noise was registered after 7,43 second for 0.14 second (that is equal to 1 sample)

The test was done for 10 seconds (700 samples) and error value was only registered for 0.14 seconds (1 sample). That means that an error value may accrue less than once every 10 seconds, which is sufficient

5.3 Balancing Test of the Inverted pendulum.

The main aim of this test is to prove that the inverted pendulum could use gyroscope sensor to balance for a significant amount of time and withstand (counteract) applied forces in a small distance.

The programs starts the balancing process and the inverted pendulum will try to balance itself. The pendulum does not know its starting position. But the balancing process was coded in a way that should allow the inverted pendulum quickly react to all disturbances and minimize the distance it will take to regain balance.

The program is described in appendix 6 7

The results are shown on figure 5.3

To make those graphs easier to explain, they will be combined and divided in to several parts. The rotations will be presented as distance traveled between samples. The falling sequence will be excluded and only readings between -310 and 110 will be presented.
Roles to read this graph:

- Red line represent gyroscope readings (degrees per second).
- Blue line represent distance traveled between samples (degrees).
- Yellow rectangles represent time gaps where Inverted pendulum was near it stable state.
- Green lines represent approximate time when forces were applied to the inverted pendulum (the pendulum was pushed)
- Green oval represents a state when an inverted pendulum was perfectly (as much as it possible in terms of precision)
- Sample time is 7 samples per second.
- P =0.51
- I =4.0
- D = 0.005

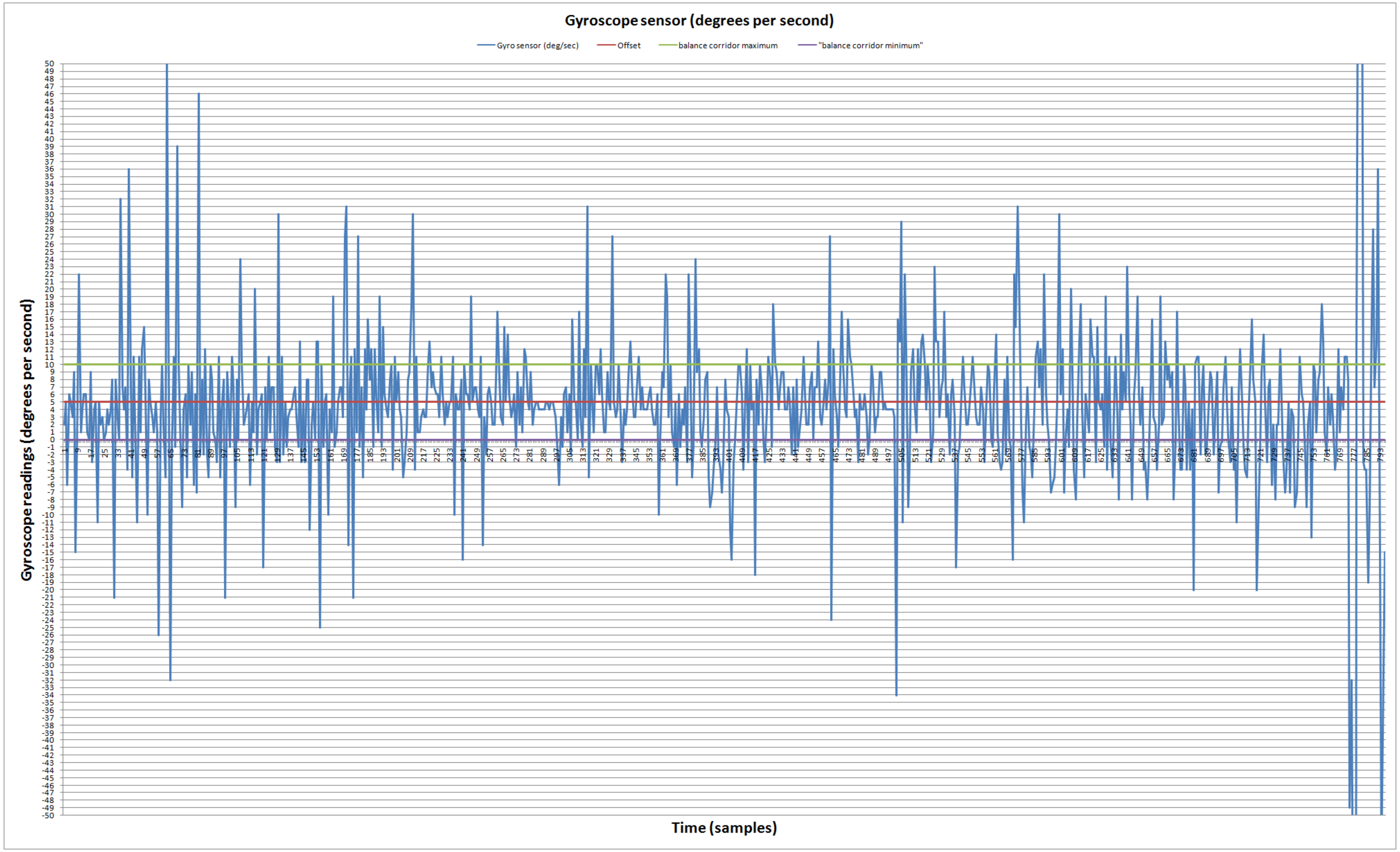


Figure 5.3 Gyroscopic sensor graph of the balancing test

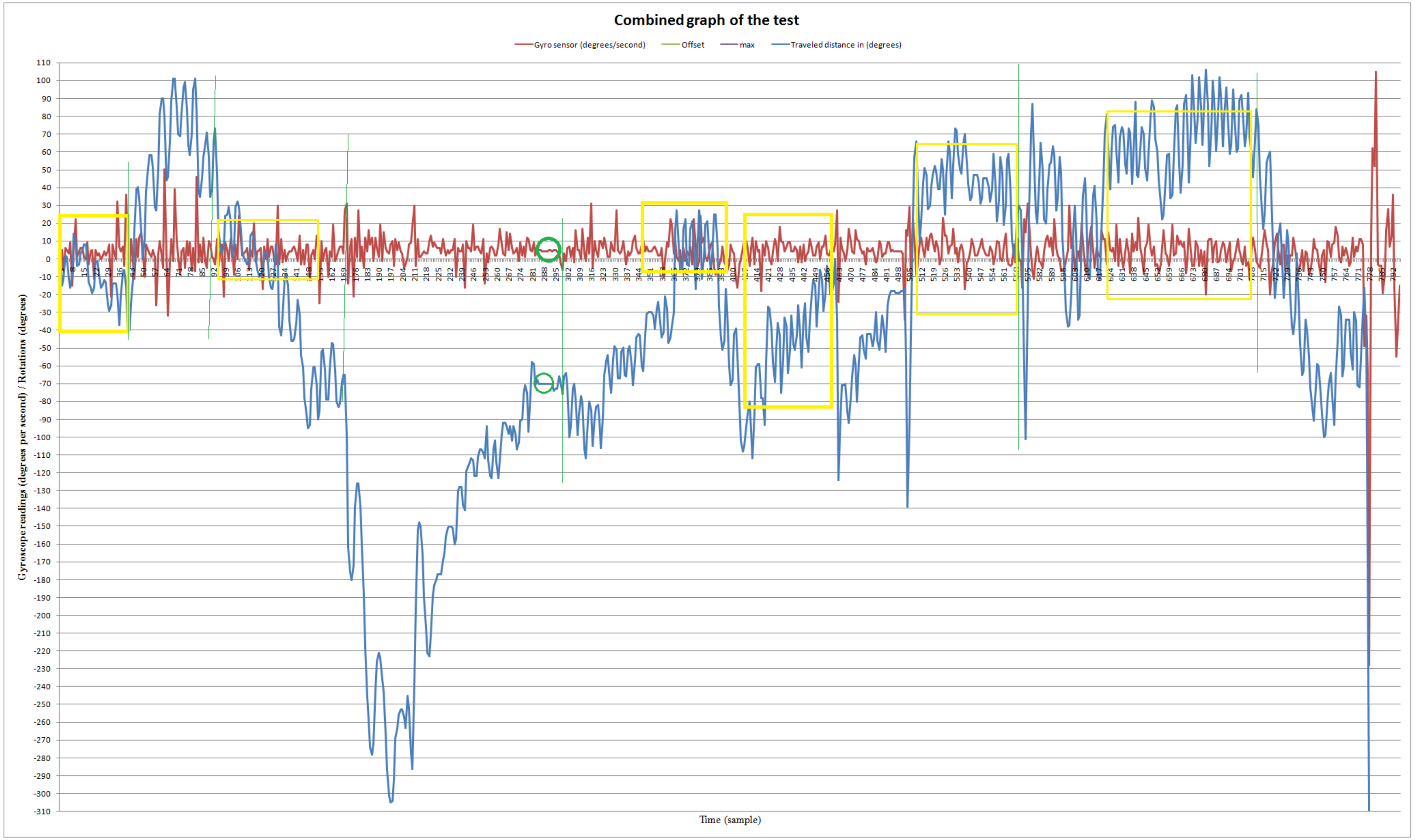


Figure 5.4 Combined graph of the balance test.

Chapter 5 – tests of a multifunctional inverted pendulum model

Overview of the results

The inverted pendulum could balance itself in a matter of seconds
The corridor of balancing is presumed to be between -5 and 10 degrees per second.
The inverted pendulum could remain a perfect balance for one second (between sample 289 and sample 296)
The inverted pendulum balanced for 773 steps = 110.4 seconds and counteracted several disturbances (applied forces). Traveled for approximately and operated in a zone of 20,134 centimeters
To calculate

$$\frac{|306|\ (\deg rees)\ +|106|\ (\deg rees)}{360\ (\deg rees)} * 56\ (mm)\ * \pi = 201{,}34\,(\text{mm}) = 20{,}134\,(\text{cm})$$

To calculate total distance traveled by the inverted pendulum a sum of all modules of distances traveled between samples was calculated and divided by 360 degrees

$\frac{10194\ (\deg rees)}{360\ (\deg rees)} * 56\ (mm)\ * \pi = 4981{,}72\,(\text{mm}) = 498{,}172\,(\text{cm})$ to compare it may be said that it took: $100\frac{201{,}34}{4981{,}72} = 4\%\ \text{of a total traveled distance to balance the inverted pendulum}$

5.4 Multi tasking test: obstacle detection

It is presumed that the designed model will be used in a real time environment and work autonomously (to some degree). For that, the model should be able to detect obstacle and avoid them to maintain stability.

To handle this task a collision detection and avoidance algorithm was encoded as a secondary objective in the main program. Meaning that the model will first try to maintain a suitable balance then check for obstacles. If an obstacle will be detected, the model will back up a little and turn to the side to avoid it. Due to the fact that the sensor (ultrasonic or light) cannot determine the dimensions of the obstacle. The model will simply turn away from the obstacle rather then come round it.

The Inverted pendulum turns in predetermined manor: one wheel rotates clockwise and the other counter-clockwise. Therefore the difference between the values of the rotation sensors will differ every time when the sensor (in this test Ultrasonic) detects an obstacle and that difference will increase at every obstacle.

Settings of the PID controller:

- P =0.51
- I =4.0
- D = 0.005

The results are shown on figure 5.5
The turning phase is shown on figure 5.6

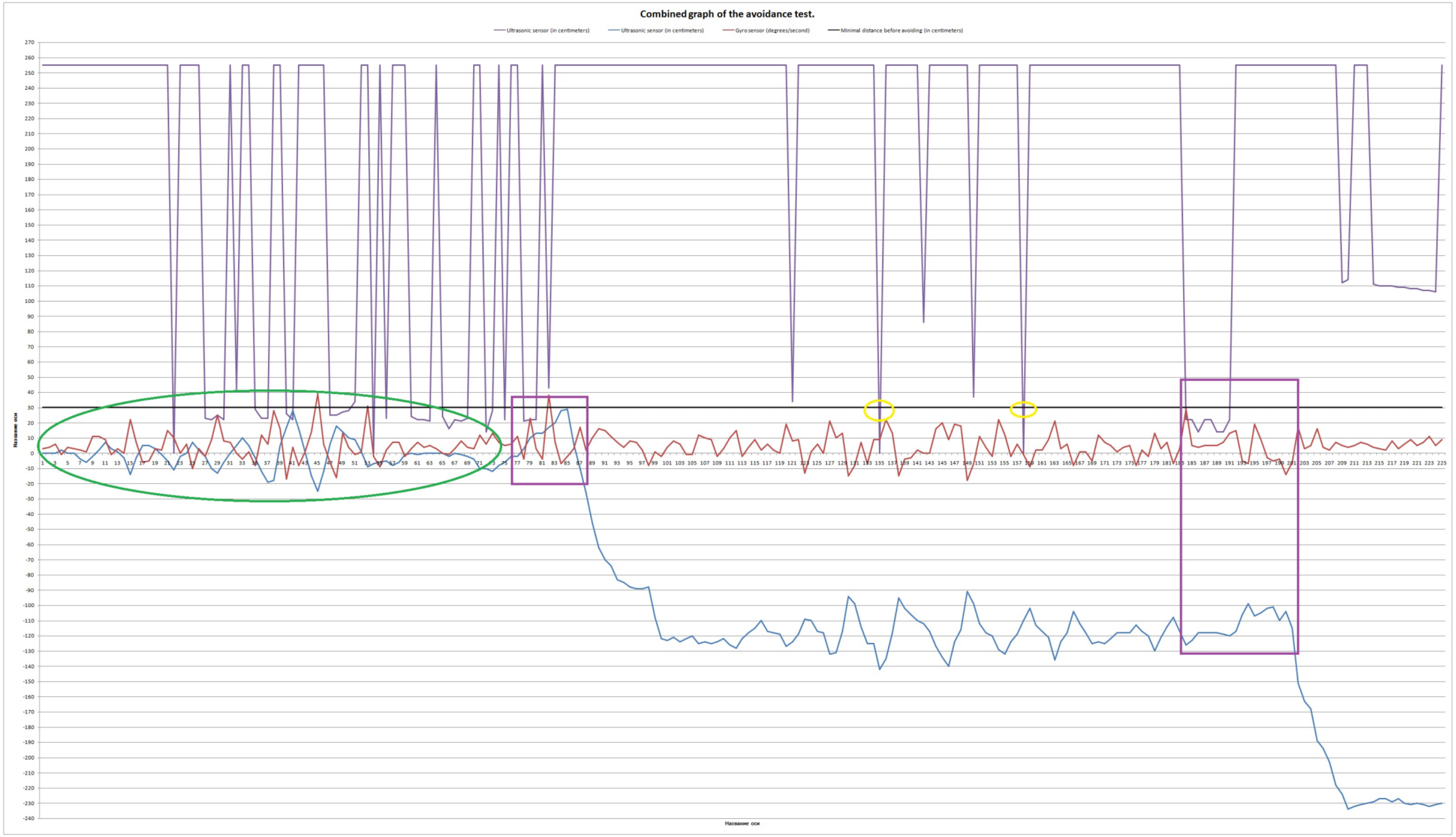


Figure 5.5 Entire graph of the Multi tasking test

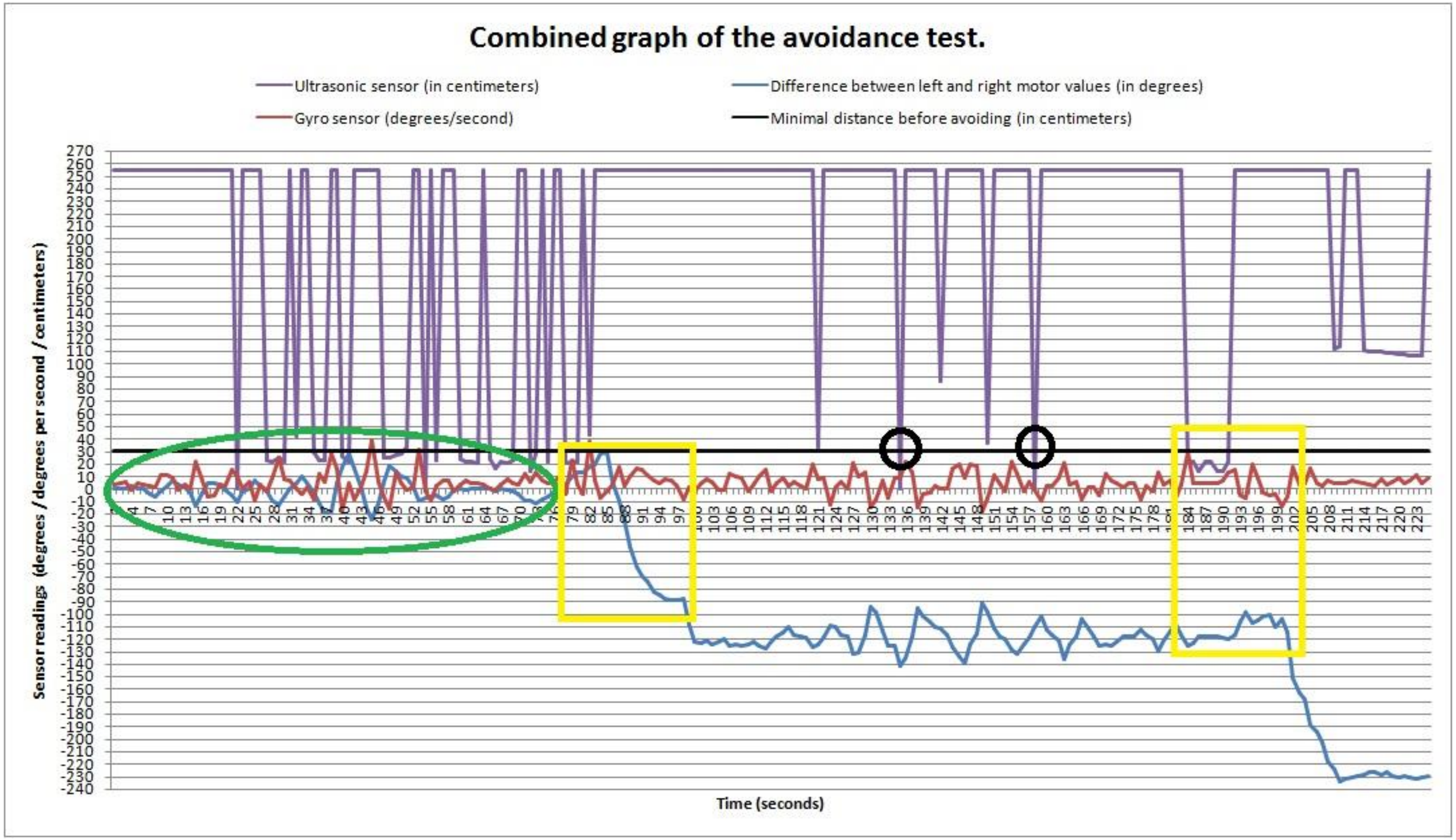


Figure 5.6 Cropped graph of the Multi tasking test

Overview of the results

The inverted pendulum started balancing and did not paid attention to the ultrasonic sensor readings until 79 sample (Dark green oval). After a suitable balance was gained, the inverted pendulum detected an obstacle then backed up and turned away from it (Yellow rectangles). After 184 sample another obstacle was detected and avoided in the same manner.

There were two noted errors (noises) that were recorded at samples 134 and 157(Dark circles), but Inverted pendulum did not react to them because there are a filter algorithm that is presented in the firmware.

Rules:

5.5 Multi tasking: floor detection

The other trouble that may happened is the fall of the inverted pendulum for a big distance (example table or stairs). So it was decided to code a similar program to detect (lowering) (a sort of elevation sensor or fall prediction sensor), although it is better to use an ultrasonic sensor to detect distance to the floor. It cannot be done in the desired model of the inverted pendulum, due to the fact that the ultrasonic sensor needs to be used in other applications and has to substitute sensors for those applications.
To solve this problem a light sensor was used to calculate approximate distance to the floor and prevent the model from falling a long distance.

Program

This program is coded as a part of the main balancing program:
The light sensor reads the reflected light from the floor and if it is too low it is presumed that the inverted the distance to the floor is too high for the inverted pendulum to travel in that direction (meaning that the pendulum will probably fall if it continues to move in that direction).
Settings of the PID controller:

- P =0.51
- I =4.0
- D = 0.005

The results are shown on figure 5.7

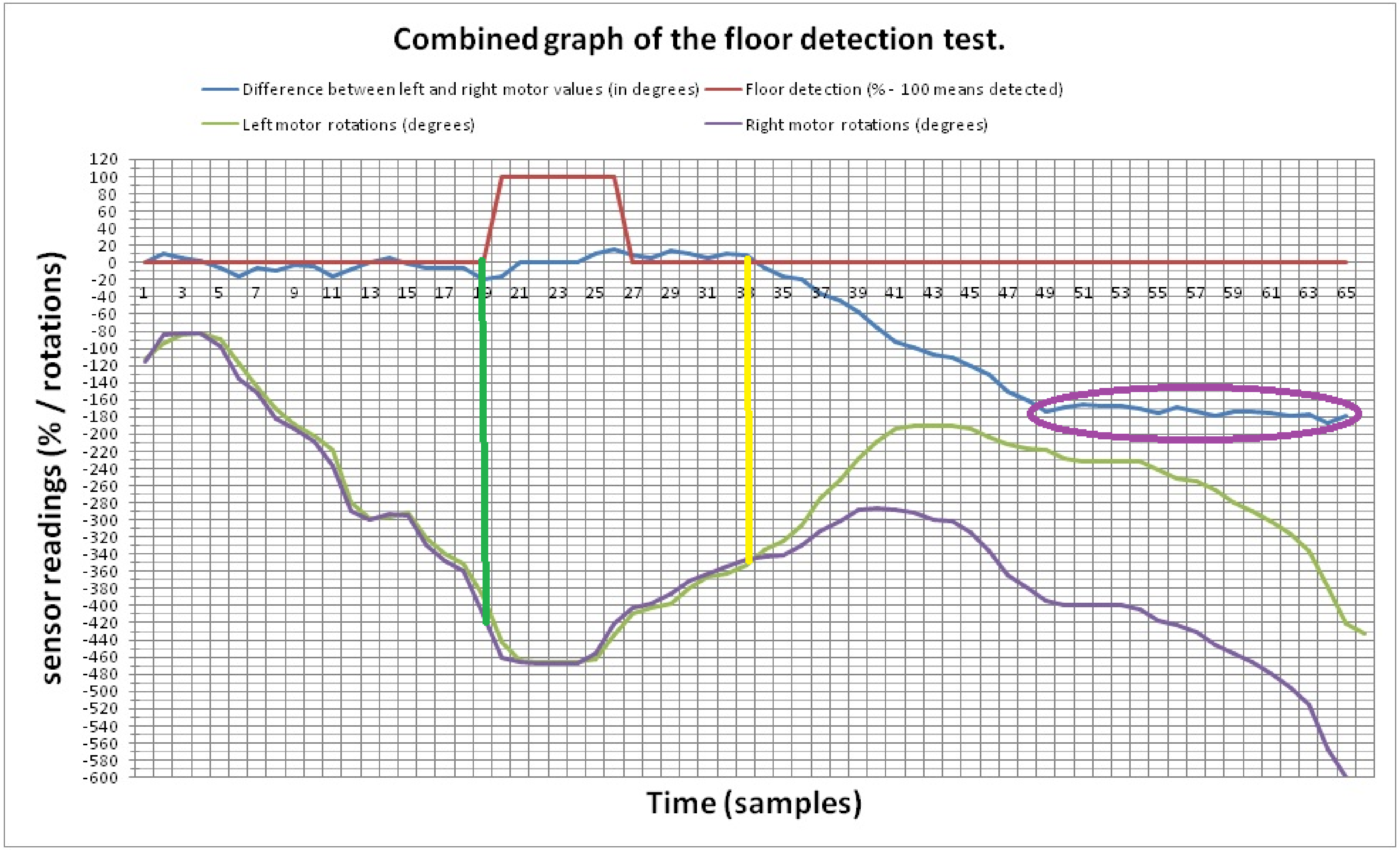


Figure 5.7 combined graph of the multi tasking: floor detection

Chapter 5 – tests of a multifunctional inverted pendulum model

Overview of the results

Rules:

- Red line indicates when floor was detected
- Dark green line shows the start of waiting and backing up phases.
- Yellow line indicates turning phase.
- Purple oval shows the continuation of forward moving.

The inverted pendulum could balance itself while moving forward (rotations are negative because the motors are designed to count rotations in other way and detect the end of the table, backup and turn away from it then continued balancing and moving forward. Note that it took about 4 seconds (29 samples) to complete the turning phase (this time is pre-coded and can be changed if the need arise). More important is the fact that it took 1 sample =0.14 seconds (maybe even less due to the fact that this was set as a lowest time interval between recordings) to react to floor detection – it is highly probable that this time is sufficient to avoid most elevation (lowering) problems.

Chapter 6 – Remote controlling and parallel processing

6.1 Remote controlling

Thanks to the modifications done in chapter 3 and tests done in chapter 4: the inverted pendulum model can detect and avoid obstacles while maintaining balance. Several types of programs were made to optimize and improve the models environmental interaction and analysis ability.

First type of interaction is the models ability to receive and follow commands from other terminal(s). In simple words a remote control function.

There are generally three possibilities to remotely control an NXT controller:

1. By coding a remote control function inside the algorithm that runs directly on the controller itself.
2. To send detect commands to the main and / or coprocessor processor of the NXT controller.
3. To use a firmware predetermined code to send signals to / from motors and sensors.

Theoretically a NXT block could be remotely controlled by:

- A remote controller
- By another NXT controller
- By another terminal, in this case word “terminal” means: PC, smart-phone and / or web application.

Every IDE, listed in chapter 3, has its own remote control over the NXT controller

RobotC

RobotC also allows customizations of the joystic to use it both on a real robot and in a virtual world. But unlike FTC, robotC requires a physical joystic to interact with a robot.

NXT-G

NXT-G only provides an option to remotely control connected NXT motors, but not sensors or sounds. An example of a remote control for the model of an inverted pendulum with a light sensor is shown in appendix 5.

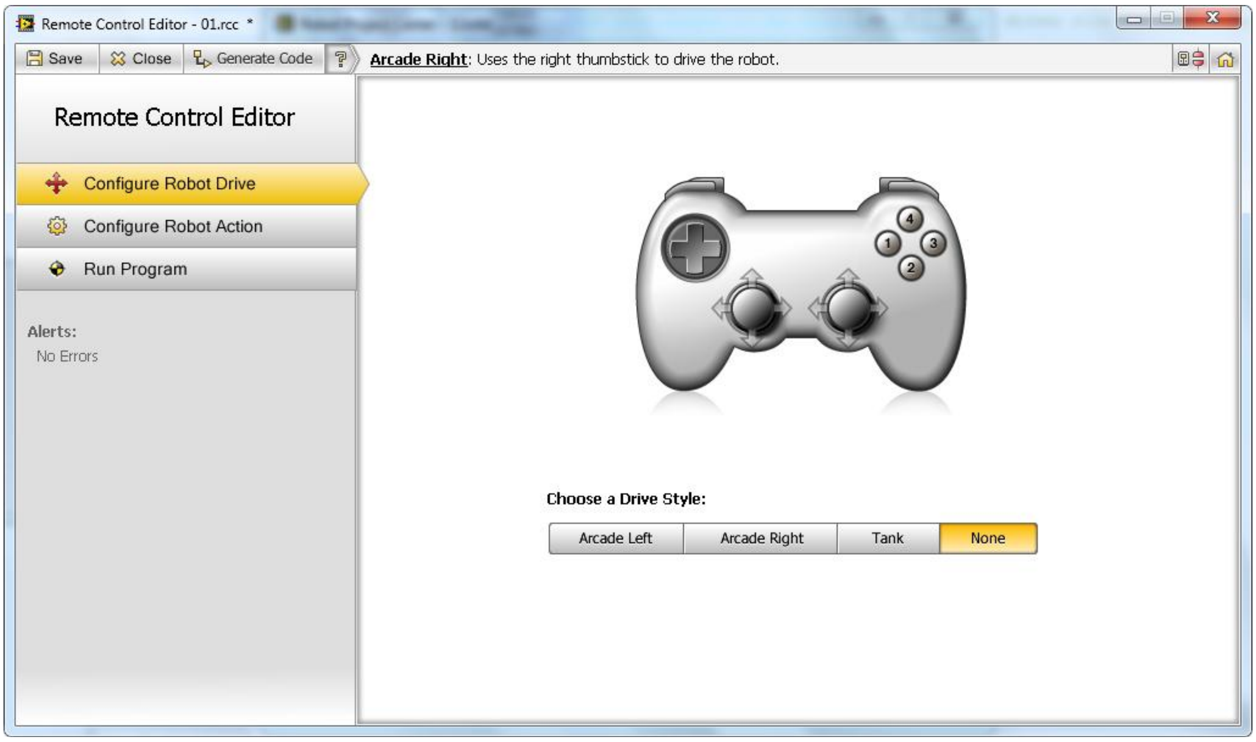
FTC

The FTC toolkit allows to code wii/PS controller and aider emulate it via Bluetooth or USB port or send store the code in the NXT controller to use with an actual wii/PS controller.

Due to the fact that an actual controller was not listed in the materials list. This function was chosen to emulate and customize a PS controller for the designed inverted pendulum model. This controller enables the user to:

- Control motors, when the inverted pendulum is presumed to be in a balanced state.
- Sand Bluetooth massages over the controller
- Control sensors if they are attached (example put the led on the light sensor on or off)

A customized FTC controller code is shown on figure 6.1.


Remote Control Editor - 01.rcc *

Arcade Right: Uses the right thumbstick to drive the robot.

Alerts:
No Errors

Choose a Drive Style:
Arcade Left | Arcade Right | Tank | [✓ None]

Figure 6.1 FTC virtual remote control.

Remote controller

The advantages of a controller type remote control are:

- Some controllers are battery powered, and therefore portable
- They are light and compact.

The limitations of a controller type remote control are:

- Limited functionality due to the maximal number of buttons and therefore commands that could be coded on them.
- Lack of communicational / interactional means (meaning no display sensors and so on)
- Most of them use IR ports or other communication ports, but do not use Bluetooth.
- No constant power supply.
- Most of them lack an ability to store files on them.

Price is not included due to the fact that it seems less practical to use this type of a controller as a main controller.

Control program on PC

For an alternative set of controllers: several programs were coded on Labview and NXT-G IDEs and robotC.

Those programs were designed to directly control and receive data from an NXT block. The graph recording program in appendix 4 can be considered one of them.

One example of a controlling program on PC is shown on figure 6.2

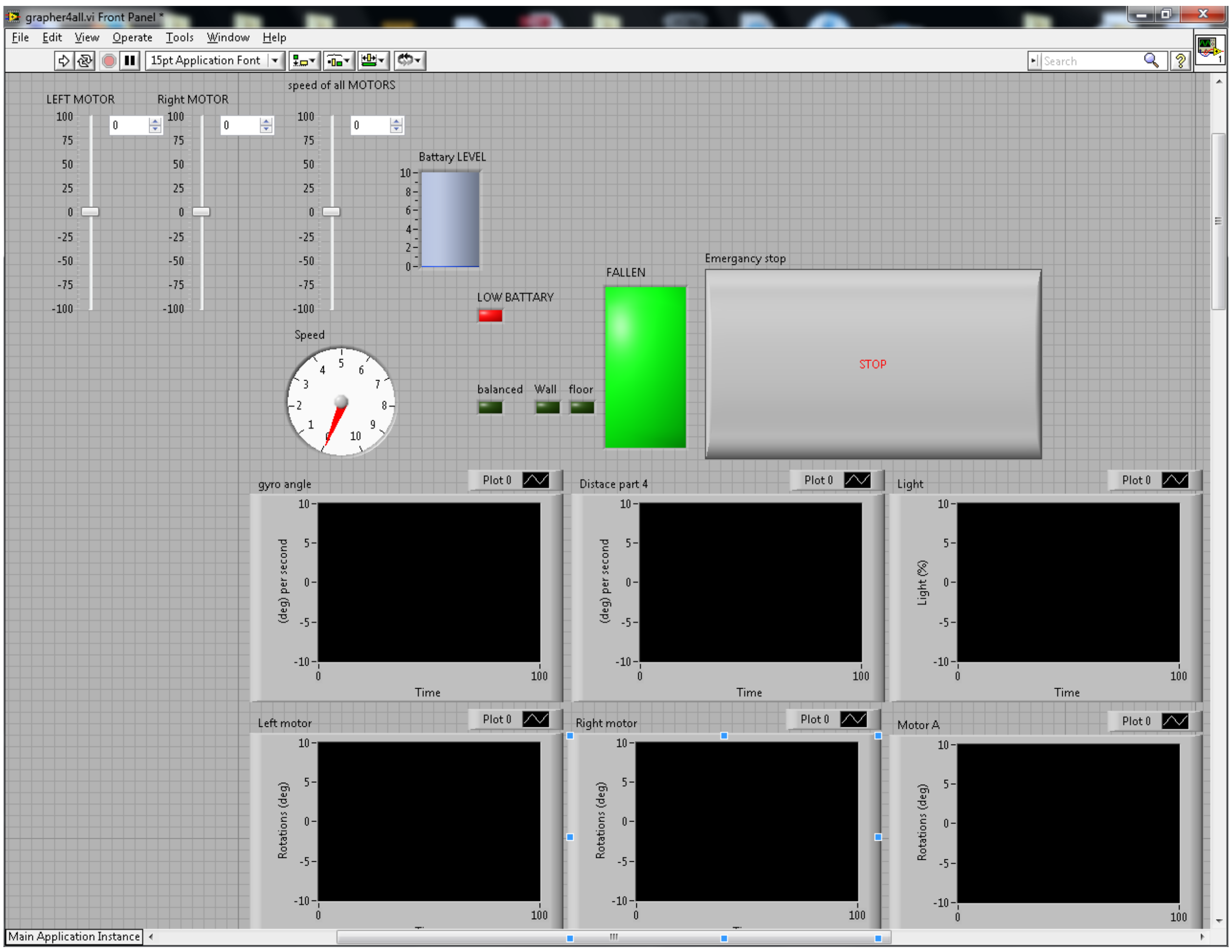


Figure 6.2 Control and recording program example in LabVIEW

PC

The advantages of a PC type remote control are:

- It can be portable (battery powered Notebooks), and stationed (connected to a constant power supply (Notebooks and/or stationed PCs).
- They have (a) powerful processor(s) that allows them to handle multiple complex tasks simultaneously.
- Generally suitable amount of memory.
- Possible connection to Ethernet and other networks including Bluetooth.
- A set of sensors that allows the model to gather additional information.
- A vide specter of possible interactions (display, keyboard, mouse/touchpad speakers, microphone and so on).
- Some PCs have build in cameras.
- They are very common in a modern society.

The limitations of a controller type remote control are:

- Stationary PCs are considered non-portable.
- Different PCs may use different OS (Operating Systems)
- Some PCs have a considerable weight
- Some PCs do not have a Bluetooth port or connection to the Ethernet.

Price:

Moderate price: They price greatly differ depending on the brand functionality and other aspects. Moderate price: A new notebook may be bought in author’s country of origin starting from 150-200 euro or bought in an online shop for 130 dollars and above.

Smart-phone

The advantages of a smart-phone type remote control are:

- Smart-phone is portable.
- Smart-phone has (a) powerful processor(s), that allows it to handle multiple complex tasks simultaneously.
- Generally suitable amount of memory.
- Possible connection to Ethernet and other networks including Bluetooth
- An option to make a phone call or text via common networks.
- A set of sensors that allows the model to gather additional information.
- A wide specter of possible interactions (sensor-display, speakers, microphone and so on)
- Almost every smart-phone contains one or even two build-in cameras and GPS sensors.
- They are very common (even more common then a notebook) in a modern society.

The limitations of a controller type remote control are:

- Functionality is theoretically still less then one of a PC terminal.
- Different OS (Operating Systems) on different brands of phones.
- Most of them use IR ports or other communication ports, but do not use Bluetooth.
- No constant power supply.
- Usually they contain too much personal and easily accessible information. That fact may discourage (scare) a person to connect to the model in question, so that he/she may help the model in question to complete its goal (mission).

Price:

Relatively cheap: They price greatly differ depending on the brand functionality and other aspects.

A new smart-phone may be bought in author’s country of origin starting from 39 euro or bought in an online shop for 50 dollars and above.

Considering many facts a smart phone was chosen to be a secondary controller (helper) for an NXT controller. Most notable facts were:

- Smartphone is light (generally) and therefore, could be placed on the model in question. This will mean that the limited range between Bluetooth connected devices will be neglected.
- Smart-phone contains many of the desired options.
- It is more probable for the model in question to meat a person with a smart-phone (smart device), and if a person does not have a required (most communications does not require them, but it is planned to wider applications for other devices that a meet person could use).
- Smart-phone may provide access to other terminals, sensors and Ethernet.

6.2 Parallel multitasking and analysis in a real time environment

To provide a wider range of possible environmental, social and assisting interactions; and to improve analyzing capabilities of the model in question, it was decided to design a network based environment that sends and gathers data to create databases and provide all the terminals with a most actual (in means of programming) information. The base concept of this network will look like this:

Components / members

1. NXT controller / Inverted pendulum.
2. Smart phone - is considered to be a secondary controller for the inverted pendulum.
3. PC based main terminal(s) – are considered to be main data gathering and analyzing centers. That means that most (and preferably all) of the data will be virtually send to this (those) terminal(s) to compile and analyze databases.
4. Guests – Other terminals that may connect to the model in question to interact with it; or those terminals to which the model in question connects to gather data.
5. Devices – all the devices that are found in any network (Bluetooth or Wi-Fi) in a proximity of the model in question.
6. Users / helpers – people with whom the model in question can socialize.

Locations

1. NXT controller / Inverted pendulum can be located any ware (in an unknown environment)
2. Secondary controller smart-phone is considered to be located in range of an NXT Bluetooth connection. (close proximity)
3. PC based main terminal(s) is (are) considered to be located any ware (even thousands of kilometers from the model in question)
4. Quests are considered to be located in range of an NXT / secondary controller Bluetooth connection. It is theoretically possible to build a bridge type connection to increase the possible connection range.
5. Devices are considered to be located in range of an NXT Bluetooth connection.
6. Users are considered to be in a visual (or similar) contact with the model in question, but there are cases when they may be away from it. To explain those situations two examples are presented.
   - The user was in direct contact with the model in question and helped it to find a way to its objective by making the model follow him to a certain point. The user in question did not had a portable device with Bluetooth and / or Ethernet or texting (SMS) capabilities. He was asked by a model in question to write a report and send it to the main analyzing terminal. After some time the user did as he was asked to.
   - The model in question could connect to a “Guest” and left a massage (Bluetooth massage) on it the user in this case ether did not saw the model in question, or was away from the terminal and the model in question has already traveled beyond the range of a Bluetooth connection of this terminal.

Data process

1. NXT controller only store programs and files that are currently in use and does not store any (almost any) information about the occurred events and environment. It can only store an initial goal and conditions for its completion. It can ask and store small amounts of data from any member of the network(usually it is vital information about its current objective).
2. Secondary controller smart-phone stores information from any network member.
   - Can send and receive any data from any connected member in a real time. Usually gathers live (actual) information from NXT controller and sensors. (Live data streaming from sensors on other terminals is not guarantied).
   - Stores all the files that may be needed for the NXT controller to communicate with the environment (for example sound files and alternative already compiled programs)

- Processes most of the data that is not vital for the current objective or is considered secondary (for example: photography detection algorithm)
- If there are no connection to the Ethernet files are stored on the device until a confirmation from the main analyzing terminal is received.
- If the Ethernet is available and it was recorded that a program was executed on the NXT controller after the date of previous confirmation from the main analyzing terminal: All data files are send to the main analyzing terminal and confirmation is awaited.
- After the confirmation from the main analyzing terminal is received the data files in question are erased after a certain time (it was set to 24 hours as default)

3. PC based main terminal.

- Can send and receive any data from any connected member.
  (a) If a Bluetooth connection to any member is established in a real time.
  (b) If there is only Ethernet connection with a considerable difference in time (Ping). May even be several days.
- Locally analyses and organizes all the gathered data in to data base (datasets)
- Sends the requested data if the identity of the receiver is confirmed by the main or secondary controller and required data is found in the database.

4. Quests can send and receive information to/from main and secondary controllers. A certain data can be send to the quest from the main PC terminal (for example a photo).
5. Devices usually enlisted and can not do anything until their status is changed to "Quests"
6. May interact with NXT and secondary controller by several means
   - If the secondary controller is connected voice may be recorded and converted into data or commands.
   - Usually "Users" can not send data to the model in question if there are no other terminals nearby.

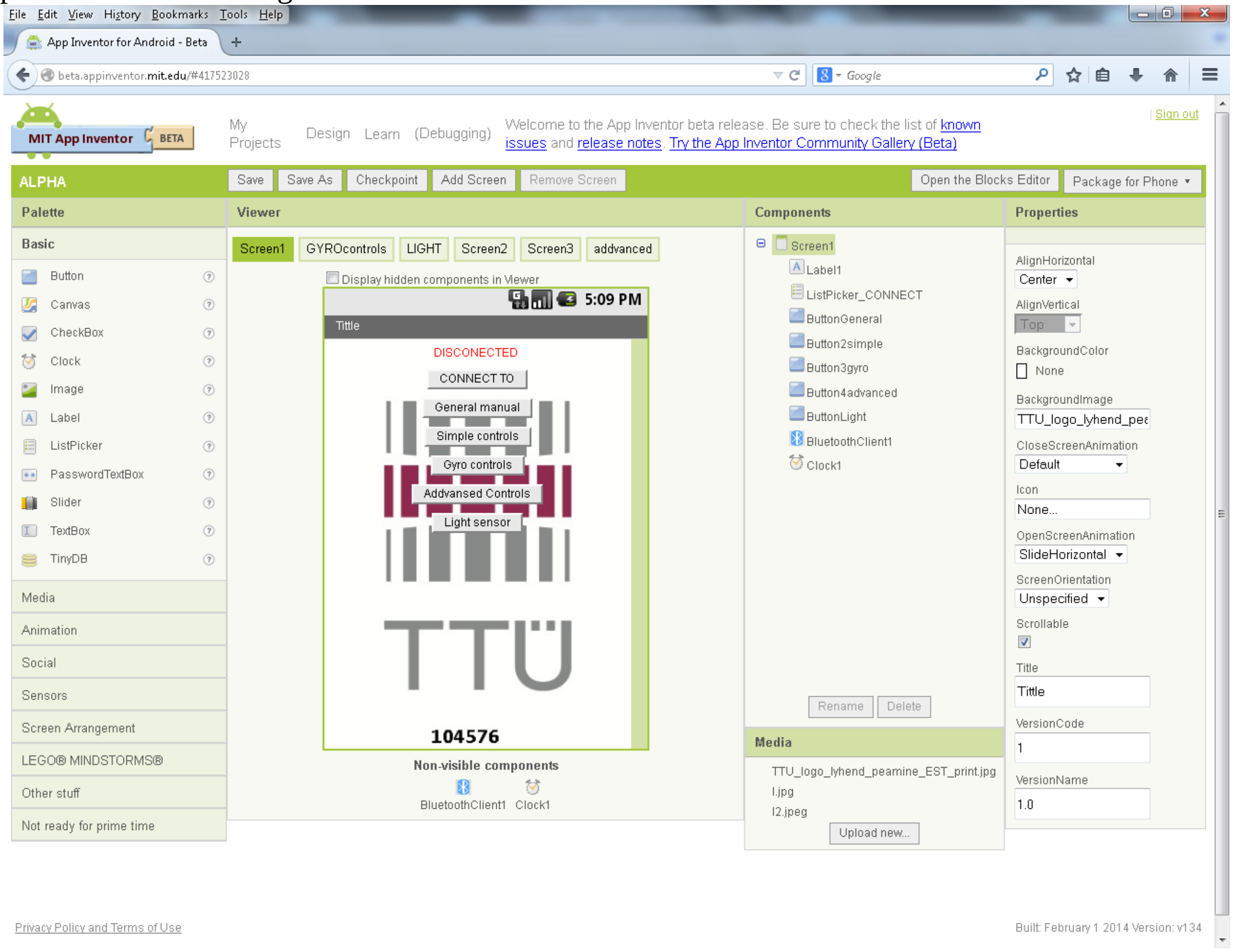
Note that all data is send either by a Bluetooth or Ethernet emails in text and seldom files via standard e-mail account.

6.3 Secondary controller example programs.

In this section programs coded for smart-phones are overviewed. It was decided to use android as a base OS on the device because it is coded in an open source code. To code programs for smart-phones it was decided to use Lejos and App inventor IDEs

Remote controls

This application represents several remote controls for the Inverted pendulum model. Code is presented shown on figures 6.3 – 6.9



AlignHorizontal: Center
AlignVertical: Top
BackgroundColor: None
BackgroundImage: TTU_logo_lyhend_pea
CloseScreenAnimation: Default
Icon: None...
OpenScreenAnimation: SlideHorizontal
ScreenOrientation: Unspecified
Scrollable: ✓
Title: Tittle
VersionCode: 1
VersionName: 1.0

Figure 6.3 An example of the IDE with a title screen of the remote control application.

Discription:

DISCONECTED - indicates the connection status to the NXT controller

CONNECT TO - This button opens a list of all available connections via Bluetooth network

General manual - This button opens a short manual for the program

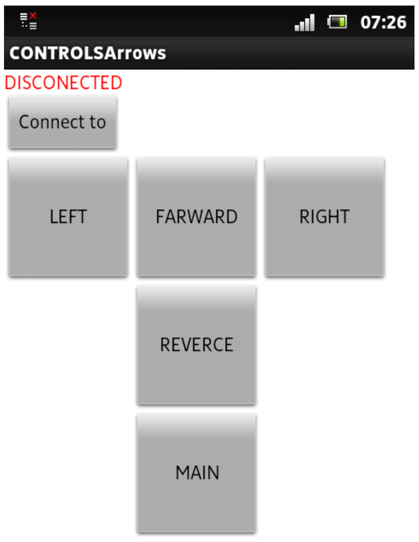
Simple controls - This button opens a simple remote control for the NXT set.

Gyro controls - This button opens a remote control that uses the acceleration sensor in the phone to remotely control the NXT this option also contains fields that allows a user to write an e-mail address and write a phone number. In case if the Inverted pendulum will fall a text (SMS) and an e-mail massage will be send to an email box.

Addvansed Controls - This button opens a remote control with debugging functions to help the developers to determine where the bug (error) is originated. (Note that this function most probably would not be included in the release version of this application).

Light sensor - This button opens a remote control over a light sensor which allows to control led of the sensor by voice or an interphone.

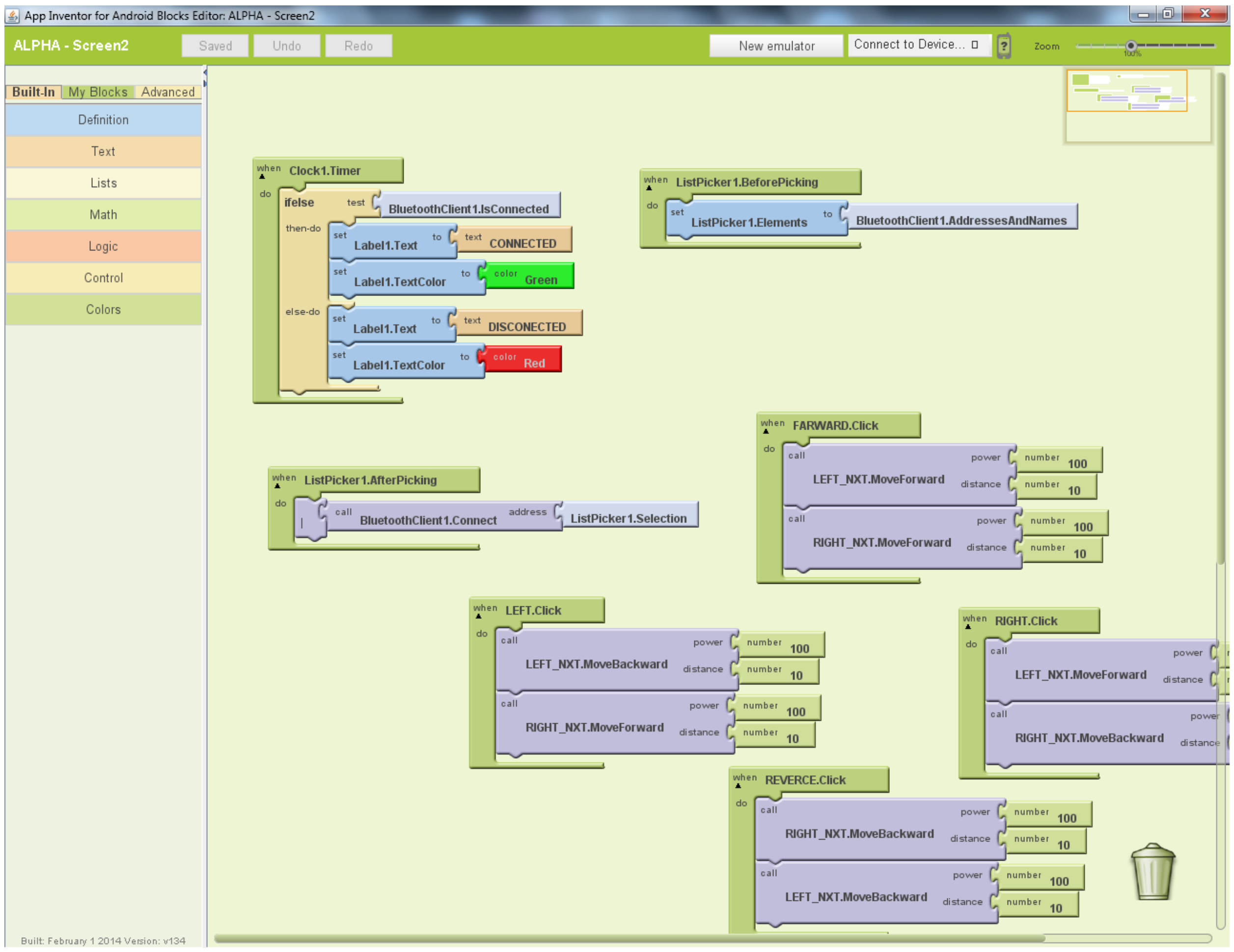
Simple remote control function



Functions:
Disconected (an intentional misprint that will be removed in the relies version) - indicates the connection status to the NXT controller.

- Connect to - opens a list of all available connections via Bluetooth network. If connected changes name to disconnect and disconnects Bluetooth communication if pushed.
- Left, forward, right and reverce – if pushed makes the inverted pendulum model to move or turn in the desired position.
- Main – if pushed returns the user to the title screen.

Figure 6.4 – A simple remote control on an actual smart–phone



```
when Clock1.Timer
do  ifelse  test  BluetoothClient1.IsConnected
      then-do  set Label1.Text to text CONNECTED
               set Label1.TextColor to color Green
      else-do  set Label1.Text to text DISCONECTED
               set Label1.TextColor to color Red

when ListPicker1.BeforePicking
do  set ListPicker1.Elements to BluetoothClient1.AddressesAndNames

when ListPicker1.AfterPicking
do  call BluetoothClient1.Connect address ListPicker1.Selection

when FARWARD.Click
do  call LEFT_NXT.MoveForward   power number 100  distance number 10
    call RIGHT_NXT.MoveForward  power number 100  distance number 10

when LEFT.Click
do  call LEFT_NXT.MoveBackward  power number 100  distance number 10
    call RIGHT_NXT.MoveForward  power number 100  distance number 10

when RIGHT.Click
do  call LEFT_NXT.MoveForward   power  distance
    call RIGHT_NXT.MoveBackward power  distance

when REVERCE.Click
do  call RIGHT_NXT.MoveBackward power number 100  distance number 10
    call LEFT_NXT.MoveBackward  power number 100  distance number 10
```

Figure 6.5 A code of the remote control on a smart–phone.

Simple remote control function

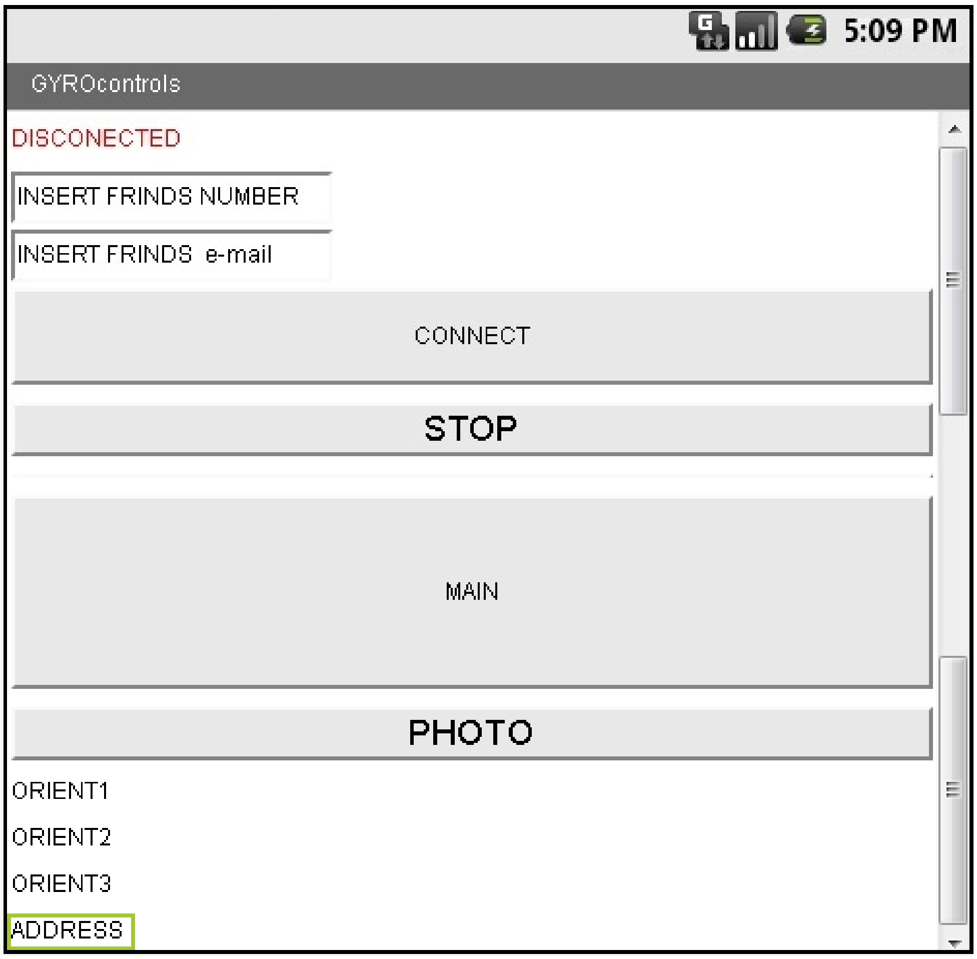


Figure 6.6 A gyroscopic type of a the remote control on a smart–phone

Functions:

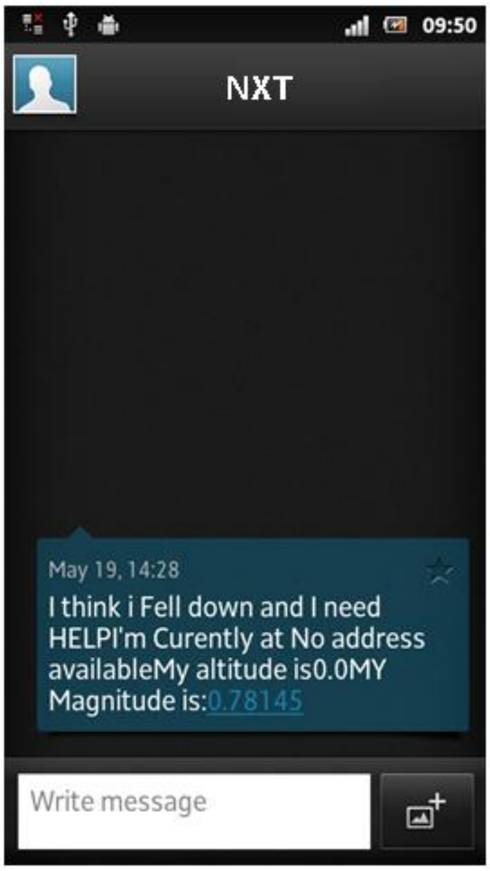
Disconected (an intentional misprint that will be removed in the relies version) - indicates the connection status to the NXT controller.

- Insert Number / e-mail allows the user to insert a number and/or email to which a massage will be send containing all gathered information that may help to determine the fallen models whereabouts. A phone call is also made to the inserted number and a voice massage can be recorded and replayed(using text to speech that is presented in most phones)

Note that it is possible to make an application to insert the phone number of the phone that is currently connected. But this may be considered as an act of illegal information gathering, so it was decided to give the user an opportunity to insert a number by his/her free will. It is also implied that this function will not be misused (example by putting the number of a boss or a local police precinct)

- Connect to - opens a list of all available connections via Bluetooth network. If connected changes name to disconnect and disconnects Bluetooth communication if pushed.
- Stop button an emergency stops signal that will stop the robot and make it fall (strictly for debugging use)
- Main – if pushed returns the user to the title screen.
- Photo button will aider open up a photograph function or (if it is determined that the robot has his main phone physically connected (meaning that the pendulum is caring the phone). Take a picture and send it via MMS to the phone and /or e-mail
- Orient fields will show data gathered from the GPS.

An Example of the text massage is shown on figure 6.7



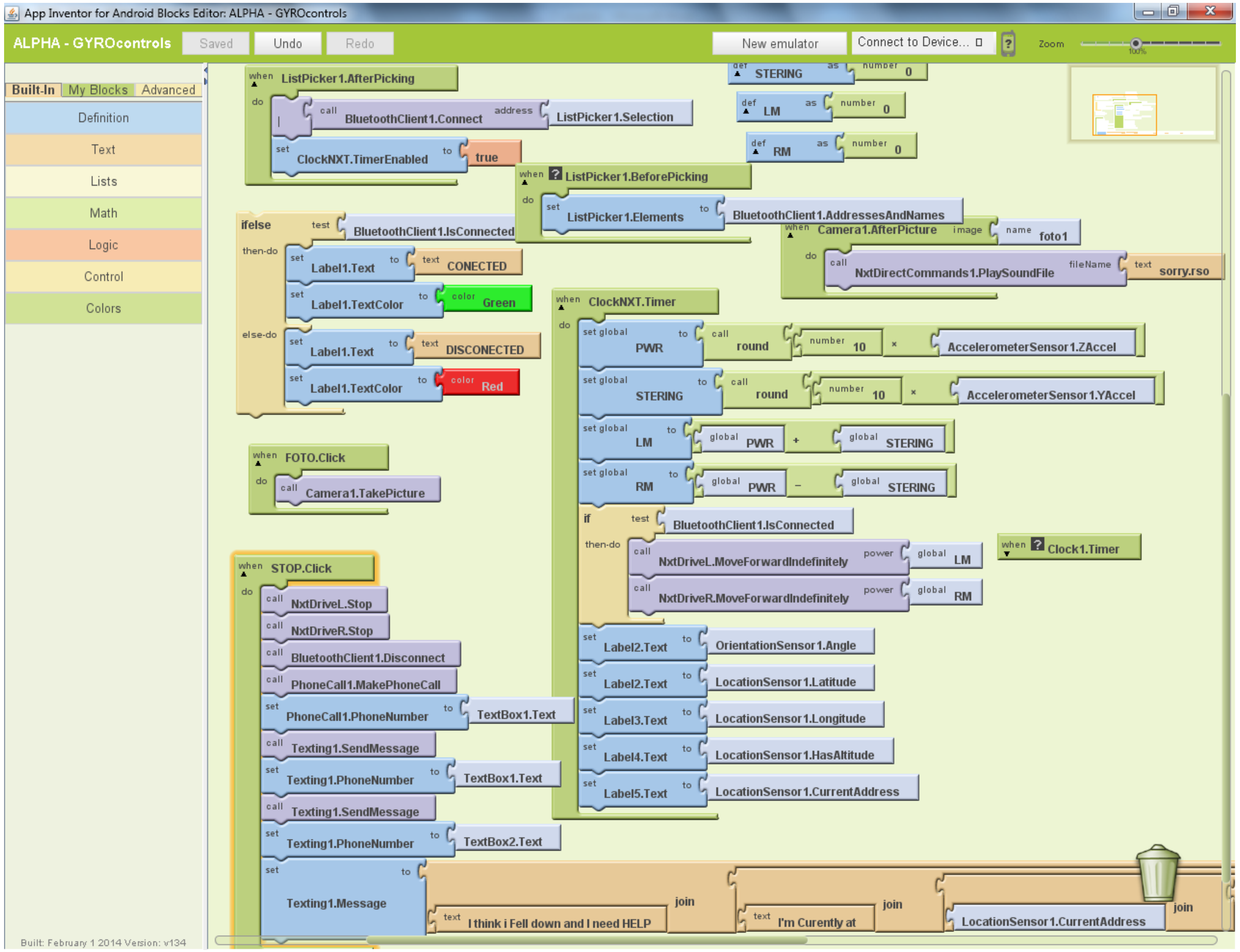
May 19, 14:28

I think i Fell down and I need HELPI'm Curently at No address availableMy altitude is0.0MY Magnitude is:0.78145

As it may be seen from this figure the model is situated at ”No address available” location. This is due to the fact that the model was tested indoors and there were no GPS signal available.
Because of this fact, an above-mentioned analysis and social orientation networks were realized (to this day partly).
It was decided to log (save) information about robots previous whereabouts and if requested send this information as an additional information to help locate the robot.

A certain question may be asked by whom ewer will read this thesis. “Why make the robot ask information if a user can ask other people instead of a robot?” Because it is presumed by the author that this system will in time evolve in to fully autonomous system that will resemble co-work and interactions between several autonomous robots. It is presumed that a fallen robot could be rescued by another robot.

Figure 6.7 An example of a text massage that the fallen robot will send.
A code example of the code is shown on figure 6.8



```
when ListPicker1.AfterPicking
do  call BluetoothClient1.Connect address ListPicker1.Selection
    set ClockNXT.TimerEnabled to true

def STERING as number 0
def LM as number 0
def RM as number 0

when ListPicker1.BeforePicking
do  set ListPicker1.Elements to BluetoothClient1.AddressesAndNames

when Camera1.AfterPicture image name foto1
do  call NxtDirectCommands1.PlaySoundFile fileName text sorry.rso

ifelse test BluetoothClient1.IsConnected
then-do set Label1.Text to text CONECTED
        set Label1.TextColor to color Green
else-do set Label1.Text to text DISCONECTED
        set Label1.TextColor to color Red

when ClockNXT.Timer
do  set global PWR to call round number 10 × AccelerometerSensor1.ZAccel
    set global STERING to call round number 10 × AccelerometerSensor1.YAccel
    set global LM to global PWR + global STERING
    set global RM to global PWR – global STERING
    if test BluetoothClient1.IsConnected
    then-do call NxtDriveL.MoveForwardIndefinitely power global LM
            call NxtDriveR.MoveForwardIndefinitely power global RM
    set Label2.Text to OrientationSensor1.Angle
    set Label2.Text to LocationSensor1.Latitude
    set Label3.Text to LocationSensor1.Longitude
    set Label4.Text to LocationSensor1.HasAltitude
    set Label5.Text to LocationSensor1.CurrentAddress

when FOTO.Click
do  call Camera1.TakePicture

when Clock1.Timer

when STOP.Click
do  call NxtDriveL.Stop
    call NxtDriveR.Stop
    call BluetoothClient1.Disconnect
    call PhoneCall1.MakePhoneCall
    set PhoneCall1.PhoneNumber to TextBox1.Text
    call Texting1.SendMessage
    set Texting1.PhoneNumber to TextBox1.Text
    call Texting1.SendMessage
    set Texting1.PhoneNumber to TextBox2.Text
    set Texting1.Message to join text I think i Fell down and I need HELP join text I'm Currently at join LocationSensor1.CurrentAddress
```

Figure 6.8 Block code of the gyro remote screen.

6.4 The social orientation program

There are several ways to communicate with terminals and people:

1. By sending Bluetooth massages to terminals

Advantages:

- There is no need for a secondary controller (smart-phone).
- Almost every Bluetooth connection protocols allow direct massage transmission between devices with no additional software preinstalled.

Problems:

- Not all devices not to mention people have a Bluetooth port.
- All received data needs to be recorded on the NXT controller in case of absence of a secondary controller.
- Bluetooth massages are not very convenient to use.
- Limitation of the massage

2. By an network web based interphone

Advantages:

- Can be accessed anywhere if there is a connection to the Ethernet.
- Easy to use and comfortable for most users.
- Can be stored at the main analyzing terminal and accessed later.

Problems:

- A dedicated server must be set to provide this function.
- Inverted pendulum model needs to be connected to a secondary controller with an Ethernet access.
- A person may forget to send the filled questioner.

3. By using users / helper device to communicate

Advantages:

- Can be used with and without a secondary controller.
- If persons device has a connection to the Ethernet the information can be immediately send to the main analyzing PC and to the NXT block or secondary controller.
- Easy to use and comfortable for most users.
- Can provide additional information like GPS, altimeter and compass data.
- If a persons device has a router option and Ethernet connection available it can provide the inverted pendulum model access to the wider network.
- Can be stored at the main analyzing terminal and accessed later.

Problems:

- A person may need to download an application on his/her smart-phone.

4. By voice (vocal) and visual communication.

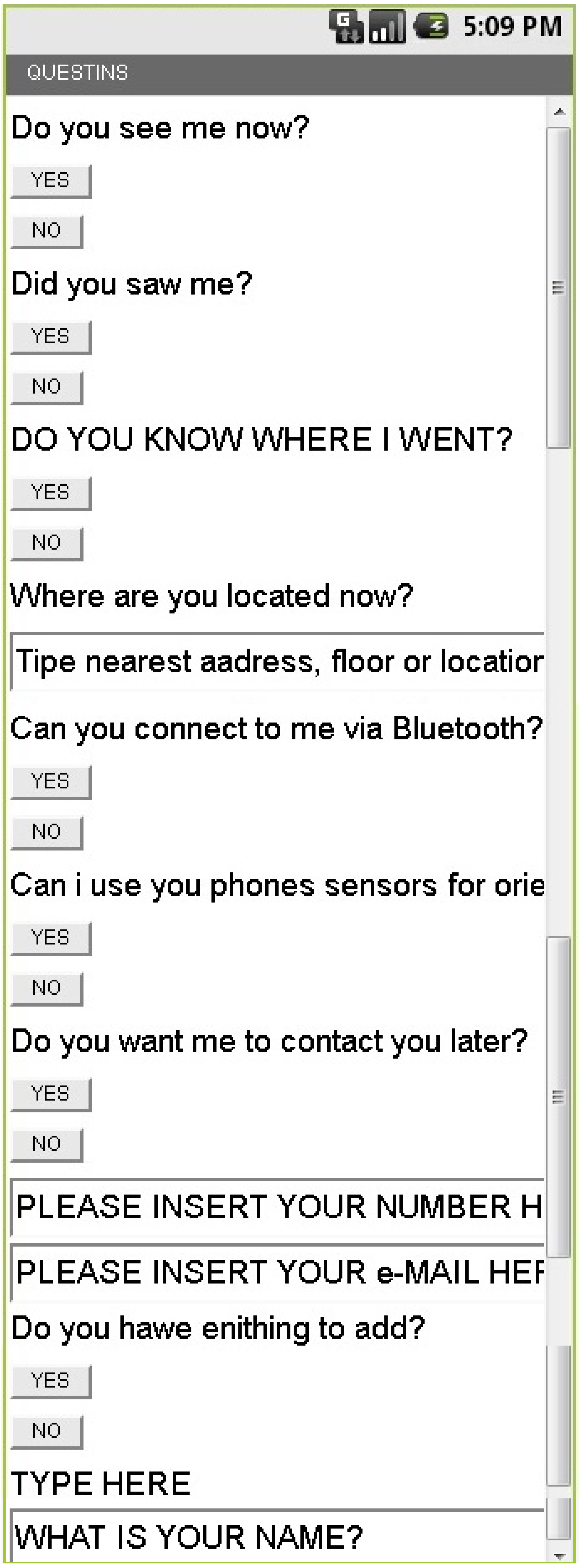
Advantages:

- Does not require the person to have any other mean of communication.

Problems:

- Speech recognition algorithms are not very accurate.
- Not everyone will understand what the robot will say to them.
- Robot may not recognize persons gestures.

An example of a questionary program is presented on figure 6.9



Do you see me now?

Did you saw me?

DO YOU KNOW WHERE I WENT?

Where are you located now?

Tipe nearest aadress, floor or locatior

Can you connect to me via Bluetooth?

Can i use you phones sensors for orie

Do you want me to contact you later?

PLEASE INSERT YOUR NUMBER H

PLEASE INSERT YOUR e-MAIL HEF

Do you hawe enithing to add?

TYPE HERE

WHAT IS YOUR NAME?

Figure6.9 Questionary example (in app inventor)

6.5 Live streaming video

To improve the models visibility it was decided to use live streaming video feed.
To do that an VLC application was used both on the PC terminal and several smart phones.
This allowed each device to stream a live video over Ethernet protocol.

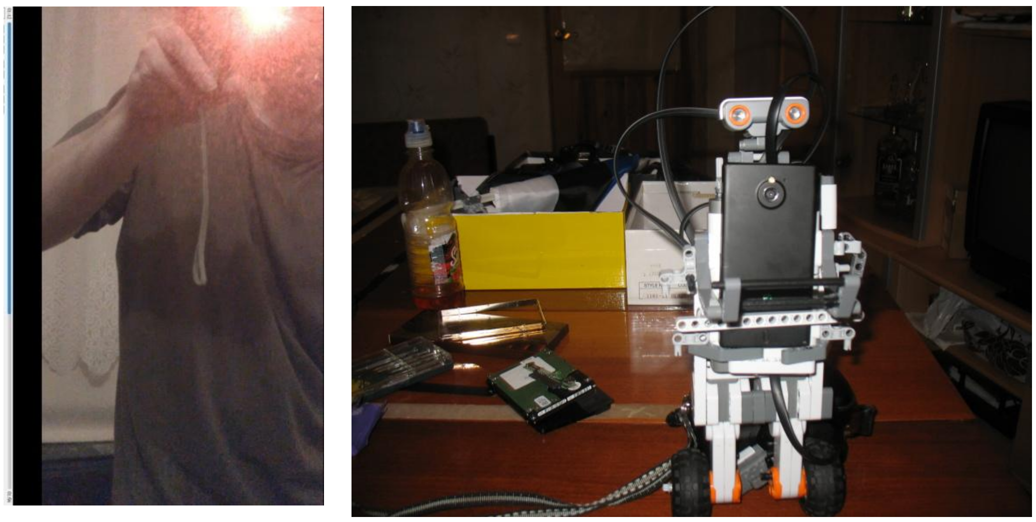


Figure 6.10 On the left: robot vision on the right: actual situation.

The program code was taken from reference link [23]

Programs that are still at testing phase.

1. Program that uses face and picture recognition algorithms to identify users.
2. Program that uses picture recognition to recognize flags (done but was only tested. with one flag) – when a flag is recognized robot / model starts playing an anthem.
3. Topological map drawer (is currently unstable)

6.6 Final prize and functionality

The project is consists of:

- One NXT 1.0 set
- 3 additional sensors
- Notebook
- Smart-phone

Although only three additional sensors were actually bought and their cost was 108.27 dollars (including customs tax and shipping price), it will be presumed that all the components were have to be bought. All components presumed to be new:

Name	Minimal price	Average price	Alternative price
NXT 1.0 set	250.00 dollars	349.95 dollars	400 dollars
3 additional sensors	108.27 dollars	108.27 dollars	108.27 dollars
Notebook	100 euro	120 euro	150 euro
Smart-phone	40 euro	60 euro	80 euro

Table 6.1 – Final price of the entire project

Minimal price 262.59 + 100 + 40 = 402.59 euro

Average 335.94 + 120+60 =515.94 euro

Maximal (by means of common sense) 372.62 +150+80 =602.62 euro

Software price

1. NXT-G, App inventor, lejos and LabVIEW toolkits are totally free
2. LabVIEW is preinstalled at many institutions so it could be used for free
3. LabVIEW has a trial period in which similar project can be written.
4. LabVIEW K13 lab supports students and educational facilities and can compensate the price of the software if a notable project was presented.
5. RobotC has a trial period in which similar project can be written.
6. If a user is able to strongly participate in the testing of this IDE a slightly different version of this product can be presented under sustain NDA conditions.

Although there are many alternative IDE that are free and suitable for NXT controller.

This set of IDEs was chosen as a most suitable set of IDEs for academical research and studies All the advantages are listed in chapter 4. Furthermore those IDE are supported by many enterprises and similar IDEs are used worldwide for coding. (MatLab was excluded due to problems of cross-platforming programming.)

Prices for single uses.

Name	Annual	Perpetual
NXT-G	Free	Free
LabVIEW	$1000	
RobotC	$24.50	$139
RobotC virtual world. with robotC	$89	

Table 6.2 – Software price of the entire project

Prices for academic uses.

Name	Annual	Perpetual
NXT-G	Free	Free
LabVIEW	$1000	
RobotC	$299.50	$599
RobotC virtual world with robot C	$499	$999

Table 6.3 – Software price of the academic uses

Final conclusion of the thesis

As a result of the work presented in this thesis, an initial introduction to the topic was presented to help understand the reasons and potential value of this thesis. Then Aims of the entire thesis were clarified. Most important of them here organized in a list – they can be summarized in one ambitious objective of building a cheap, portable, mobile, re-modifiable, re-programmable, easy to build, multi-tasking inverted pendulum model using NXT 1.0 set. Than all needed equations for construction and tuning of the inverted pendulum model were derived.

To obtain a most suitable component for the model a brief market research was conducted and several possible solutions were compared to find out, which set of components posses better qualities. As a result of those actions an NXT 1.0 set was obtained and thoroughly analyzed. Some notable but not critical problems meaning: a faulty display, low memory and battery power were found and their possible solutions were presented.

To test capability of the initial set a test model of the inverted pendulum was constructed and put to the test. Those tests had shown that it was possible to build, code and tune the inverted pendulum model with (only) one light sensor. The model in question could maintain balance and counteract some applied forces and disturbances, but had a limited working area and was very sensitive to lighting. And therefore was unsuitable for the initial objective.

To overcome those limitations the entire NXT 1.0 set and Programming IDEs were heavily modified and customized to wider the capability of an entire system. Additionally, a wider market research was conducted to find more suitable sensor(s) for both balancing and allocation problems. As a result of those actions, additional sensors were acquired and IDEs were re-customized to work simultaneously and gather data more efficiently. The design of the inverted pendulum model was modified to handle several tasks simultaneously,

A final inverted pendulum model was build and tested to handle multiple tasks, including object avoidance and floor (falling) detection, while balancing and working autonomously in a real time environment. To enhance functionality of the model in question it was implemented into a specific system, that consisted of one to several remote terminals and now possessed variety abilities. A smart-phone was set as a secondary controller for the inverted pendulum, thus granting ability of remote access, control and data transfer over Bluetooth and Ethernet protocols. Making the model able to communicate not only with other devices (terminals), but with people as well ask directions and guidance to obtain vital information and then determine its current location in a real time, and even call for assistance in case of an emergency. Additionally, it allowed for some parts of the programmed code to be changed in a real time without the need of interrupting current tasks and / or executed algorithms.

Useful skills were obtained, an understanding of a real time stabilization algorithms and localization problems.

The author plants do develop ideas presented in this thesis in the future. It is presumed that multiple similar robots could be included in the above mentioned system to work as a single unit.

Reference list

0. IDEs help documentation.
1. "Inverted pendulum on a cart" süsteemi juhtimine Nikolai Vidjajev 2010 – includes many theoretical material that was used in this work.
2. http://www.pt.ntu.edu.tw/hmchai/BM03/BMClinic/Walk.htm#locomotion – walking
3. Бернштейн Н. А. Очерки по физиологии движений и физиологии активности. — М., 1966. Can be found on a link below pages 338, 356.

- http://publ.lib.ru/ARCHIVES/B/BERNSHTEYN_Nikolay_Aleksandrovich/_Bernshteyn_N.A..html – the library where this book could be obtained.

4. http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%8F%D1%82%D0%BD%D0%B8%D0%BA#cite_note-2 –uses of the Inverted pendulum system
5. http://earthquake.usgs.gov/learn/topics/seismology/history/part12.php - information about seismograph.
6. Ракеты на твёрдом топливе в России Авторы: В.Н. Сокольский page 75-90 – confirmation of rocket constructions and the inverted pendulum usage for this since 1846 year.

- http://exploration.grc.nasa.gov/education/rocket/rktstab.html - alternative material

7. Yamamoto Y, (2008) NXTway-GS Model-Based Design - Control of self balancing two-wheeled robot built with LEGO Mindstorms NXT
8. http://en.wikipedia.org/wiki/PID_controller - description of the PID controller
9. http://www.mstarlabs.com/control/znrule.html - tuning methods
10. Extreme NXT: Extending the LEGO MINDSTORMS. NXT to the Next Level Second Edition. Authors: Michael Gasperi and Philippe "Philo" Hurbai 2009. Pages 7-20 and 135 - 140
11. Mindstorms Software Developer Kit – manual of mindstorms NXT 1.0 set
12. Mindstorms Hardware Developer Kit – manual of mindstorms NXT 1.0 set
13. Mindstorms Bluetooth Developer Kit – manual of mindstorms NXT 1.0 set
14. http://www.legomindstormsrobots.com/lego-minstorms/nxt-faulty-display - faulty-display
15. http://blogs.wsd1.org/etr/?p=495 - faulty-display repair
16. http://www.nxtprograms.com/segway/ - original model and code used in chapter 2
17. www.robotc.net – description of RobotC IDE
18. http://www.robotvirtualworlds.com/download/ description of robotC virtual world
19. http://www.ni.com/academic/mindstorms/ - description of LabView
20. https://decibel.ni.com/content/docs/DOC-17997 description of LabView FTC
21. http://www.appinventor.org/ description of App Inventor
22. http://www.robokits.co.nz/MS240 DIST-Nx-Medium-v3 sensor
23. http://www.dexterindustries.com/dIMU.html - dIMU sensor
24. http://www.hitechnic.com/cgi-bin/commerce.cgi?preadd=action&key=NGY1044 - HiTechnic NXT Gyro Sensor and code
25. http://robotsquare.com/2012/02/12/tutorial-building-segway/ - design and instruction that were modified to construct the model
26. http://www.videolan.org/vlc/ - VLC homepage

Appendix 1 - Test of NXT 1.0 motors

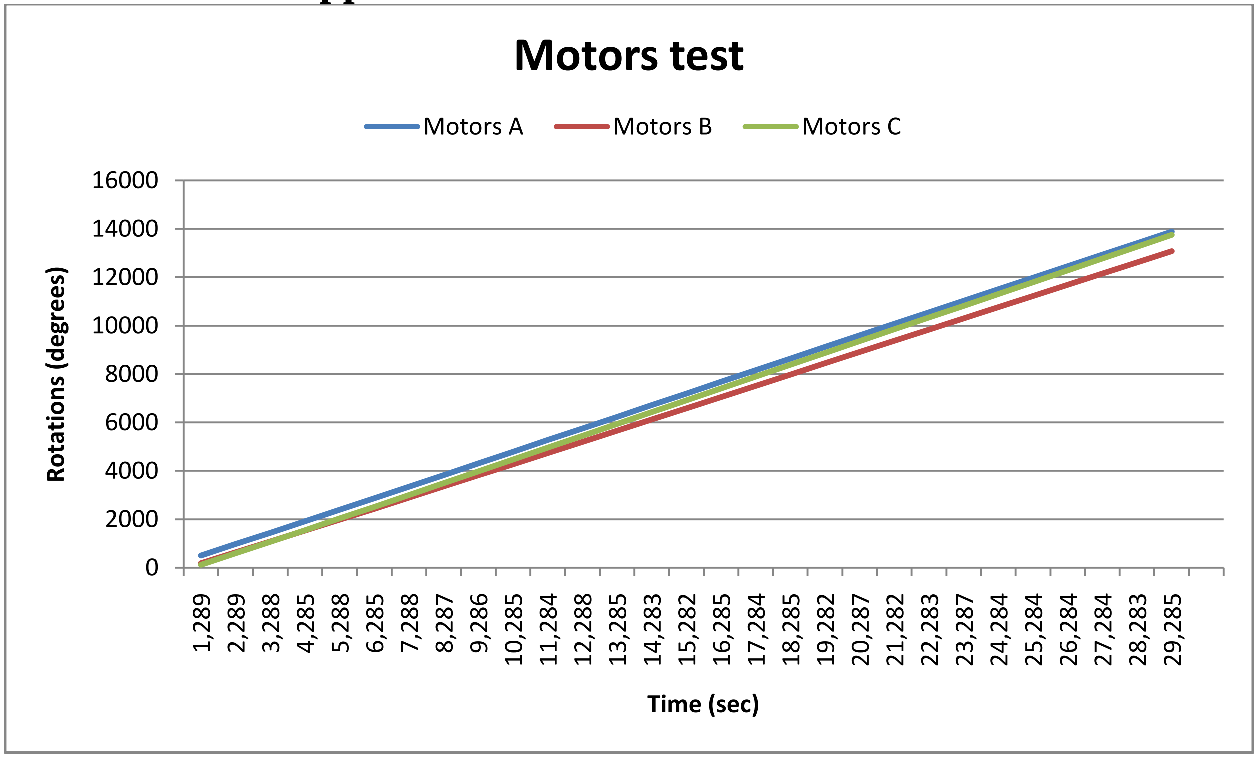
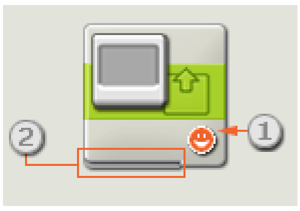


Figure 01 Motors test graph.

To determent a most suitable pair of motors a series of tests were done. All motors were connected to the NXT controller and their rotations were recorded as they run at full speed.

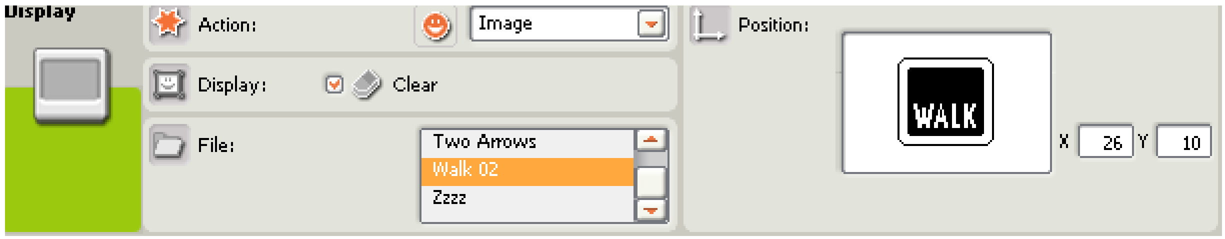
As it may be seen from the graph most suitable pair of motors are motor A and Motor B (note that those are numbers of the motors and not ports.

Appendix 2 - Functions of every block that is used in the NXT-G program [0]



Display

1. This icon shows whether the block is set to display an image, some text, or a drawing; or whether it will just reset the display to the default icon.
2. You can change values dynamically by connecting data wires to this block's data hub.

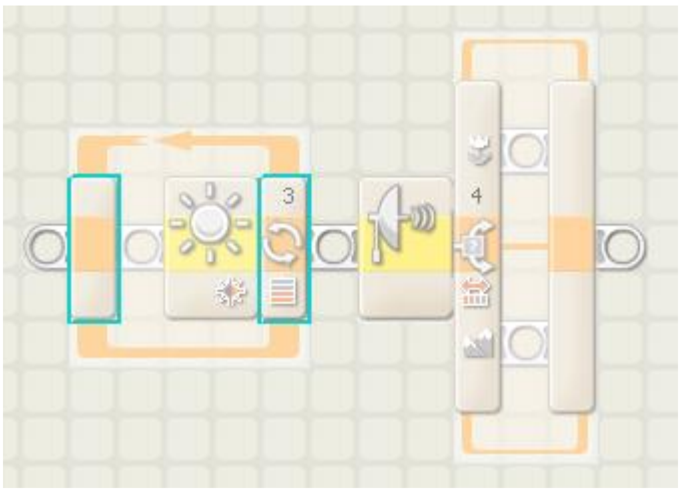


Display
Action: Image
Display: ✓ Clear
File: Two Arrows / Walk 02 / Zzzz
Position: X 26 Y 10

Figure 0.21 NXT-G display block and settings

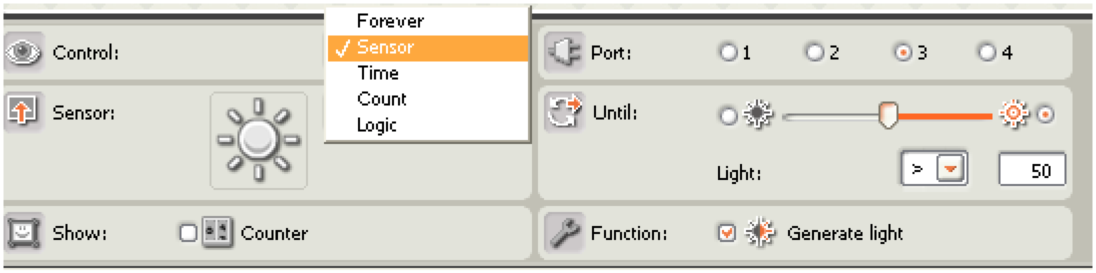
Display block allows to program information that is displayed on the NXT monitor.

Loop / switch blocks



The block allows to create loops and switches for the program. They can process:

1. values
2. sensors
3. logic
4. time
5. can run forever
6. can run for set number cycles



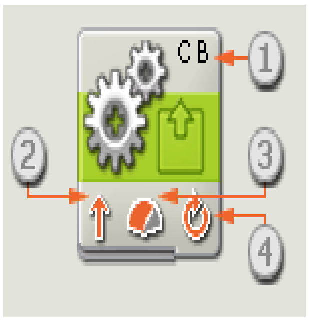
Control: Forever / ✓ Sensor / Time / Count / Logic
Sensor:
Show: ☐ Counter
Port: ○1 ○2 ●3 ○4
Until: ○ ●
Light: > 50
Function: ✓ Generate light

Figure 0.22 NXT-G Loop / switch block and settings

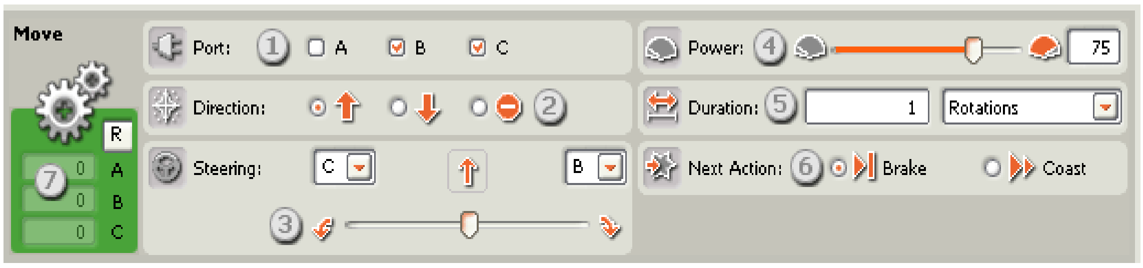
This particular loop runs until or when the light sensor value is more then 50

Appendix 2

Motor block



1. The letters at the top right corner of the block show which of your NXT ports will be controlled.
2. This icon shows which direction your robot will go.
3. This icon shows the power level. Your robot's speed may also be affected by other conditions, like the surface it is moving over or whether it is moving up or down.
4. This icon shows whether you have set the Duration property to unlimited, degrees, rotations, or seconds.



Move

Port: 1 ☐ A ☑ B ☑ C

Direction: ● ○ ○ 2

Steering: C B 3

Power: 4 75

Duration: 5 1 Rotations

Next Action: 6 ● Brake ○ Coast

7 0 A 0 B 0 C R

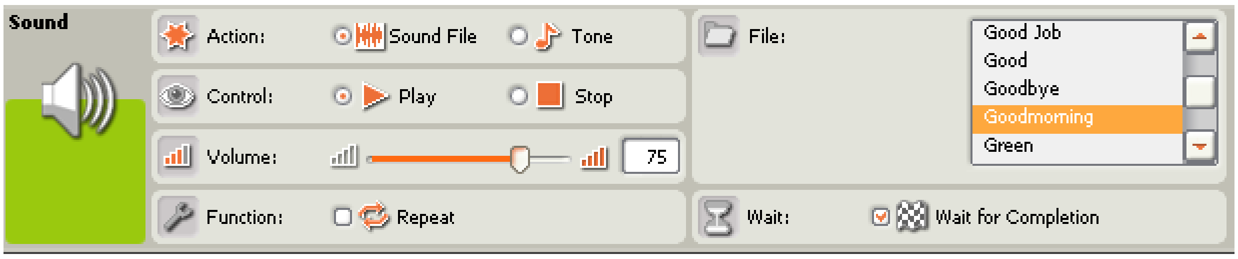
Figure 0.23 NXT-G motor block and settings

This block allows to program motor/rotation sensor input and outputs.

Sound block

This block allows to program output sounds that will be playd by the NXT controller. The volume level is controlled by this block.

Sounds can be prerecorded by any recording program in .pno or .wav formats



Sound

Action: ● Sound File ○ Tone

Control: ● Play ○ Stop

Volume: 75

Function: ☐ Repeat

File: Good Job, Good, Goodbye, Goodmorning, Green

Wait: ☑ Wait for Completion

Figure 0.24 NXT-G Loop / sound block and settings

Sensors

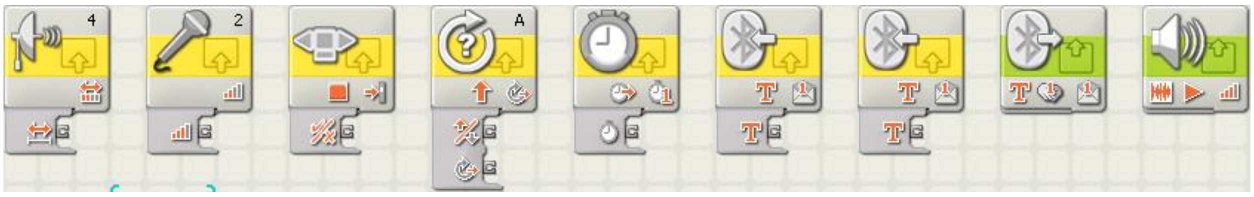


Figure 0.25 NXT-G sensor blocks and settings

1. Those blocks controls the information from every sensor connected to the NXT,
2. The info can be send (green) or received (yellow).

Appendix 2

Data blocks

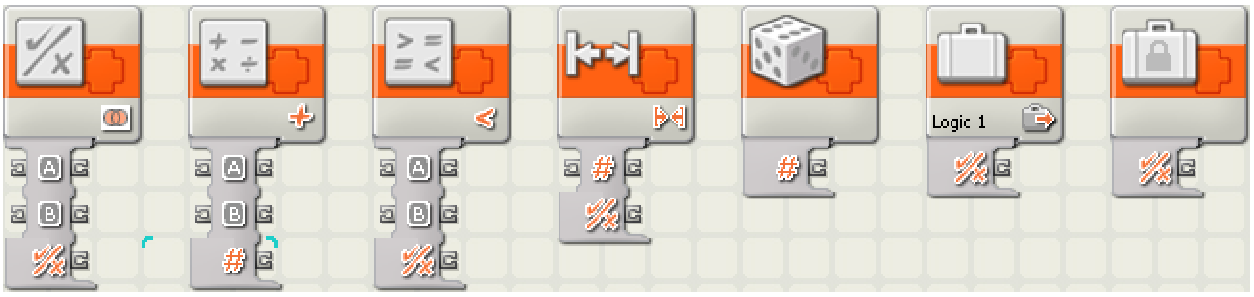


Figure 0.26 NXT-G data blocks

Those blocks contain variables and mathematical functions.

Advansed blocks

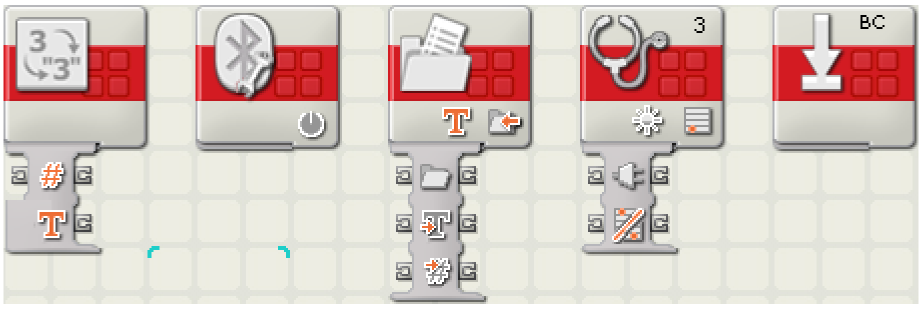
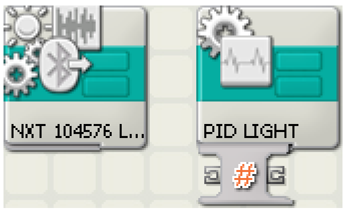


Figure 0.27 NXT-G Advanced blocks

Are blocks that work with files, data formats, offset values and connections to USB or Bluetooth

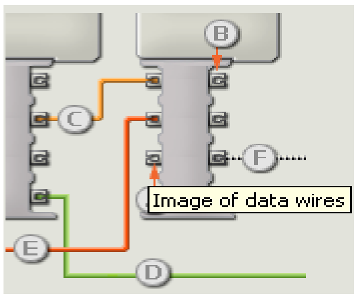
My blocks



My block are several of the above described blocks or parts of the program packed in one block

Figure 0.29 – NXT-G my blocks

Wires and connections



[A]	Input plug
[B]	Output plug
[C]	Number data wire (yellow)
[D]	Logic data wire (green)
[E]	Text data wire (orange)
[F]	Broken data wire (gray)

Figure 0.210 – NXT-G data transferring wires / connections

Appendix 3 segway C.code

```
    const tSensors LightSensor          = (tSensors) S3;

//Initialize variables
int lightValue = 0;               //Raw light sensor value (units)
int midiValue = 0;                //Vertical light sensor value
int err = 0;              //start Error
int oldErr = 0;           //Old error used for comparison
int diffErr = 0;          //Differential (derivative) error
int intErr = 0;           //Integral error
int pid = 0;              // start PID Value

//Declare constants
const int kp = 60;                //Constant proportional multiplier P
const int ki = 3;                 //Constant integral multiplier I
const int kd = 8;                 //Constant differential (derivative) multiplier D

const int scale = 55;             //Scales the PID value for motor input
const float damp = 0.66;                  //Damping value for controlling integral error can be
changed
const float corrector = 0;                //weight compensater / corrector can be different

task main()
{
        nSyncedMotors = synchBC;              //Synch Motors

        while (nNxtButtonPressed == -1)                //Wait for a button press
        {
        }
        midiValue = SensorRaw[LightSensor];                   //Read vertical light value

        wait1Msec(5000);                  //Wait 5 seconds before balancing

        while(true)               //always on
        {
                lightValue = SensorRaw[LightSensor];                  //Read light sensor raw data

                err = lightValue - midiVal;               //Calculate error

                //Compensate if not 0
                if (err > 0)
                {
                        err = err * corrector;
                }
```

```
		diffErr = err - oldErr;		//Calculate differential error

		intErr = intErr + err;		//Calculate integral error
		intErr = intErr * dmp;		// damping

		pid = kp * err;		//Compute proportional component and add it to PID value

		pid = pid + (ki * intErr);		//Compute integral component and add it to 
PID value

		pid = pid + (kd * diffErr);		//Compute differential component and add it 
to PID value

		pid = pid / scale;		//Scale the PID value for motor input no more then 
100% of output

		// PID values into the motor scale even after being scaled
		if (pid > 100)
		{
			pid = 100;
		}

		// PID values into the motor scale even after being scaled
		if (pid < -100)
		{
			pid = -100;
		}

		motor[motorB] = pid;		//Set PID value to the motors
	}
}
```

Appendix 4 – data gatherer in LabVIEW

A overview of the program written in LabVIEW to gather info from the light-Segway. (figure)

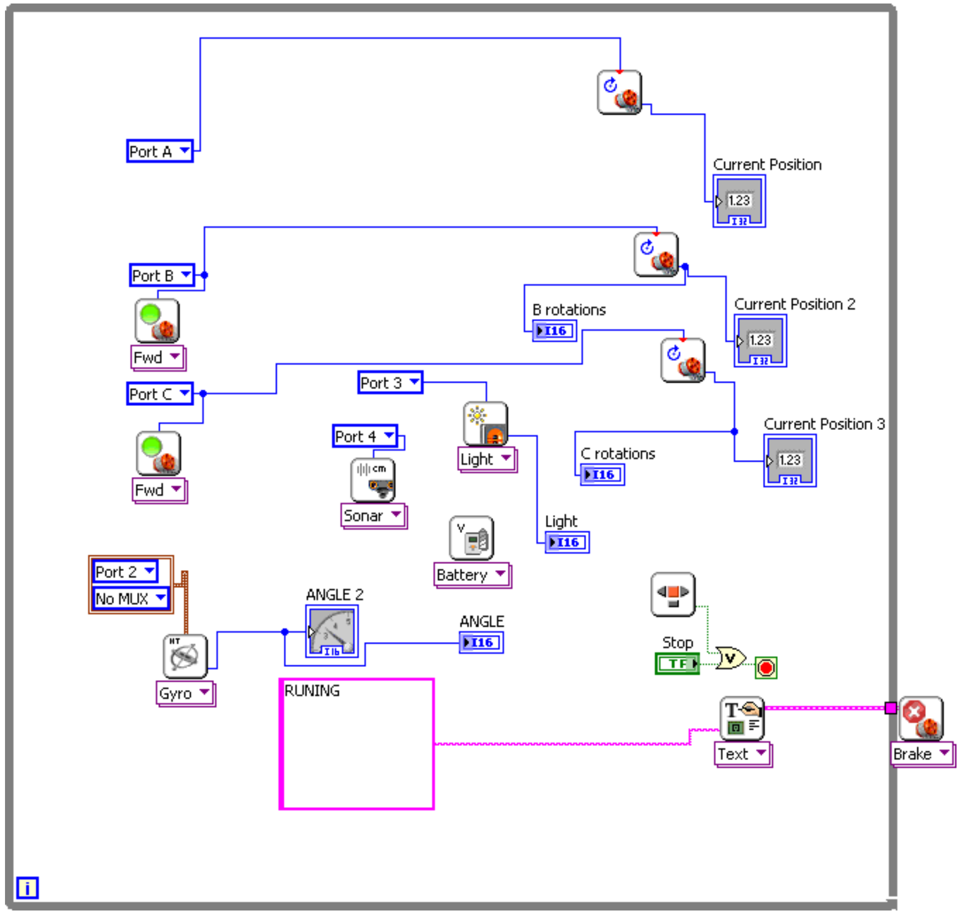


Figure 04 block diagram of graph4light.VI

The program records all of the information from sensors (light sensor and rotation sensors) and presents them both in graphical and numerical forms in a real time.

Appendix 5 - Additional testing

the program with Bluetooth controlled driver (Motor A). It worked fine but the model itself handled the balancing task in a limited area, and therefore was unsuited for completing stated aims. The function is added by putting the block (shown on figure 05) inside of the controlled algorithm.

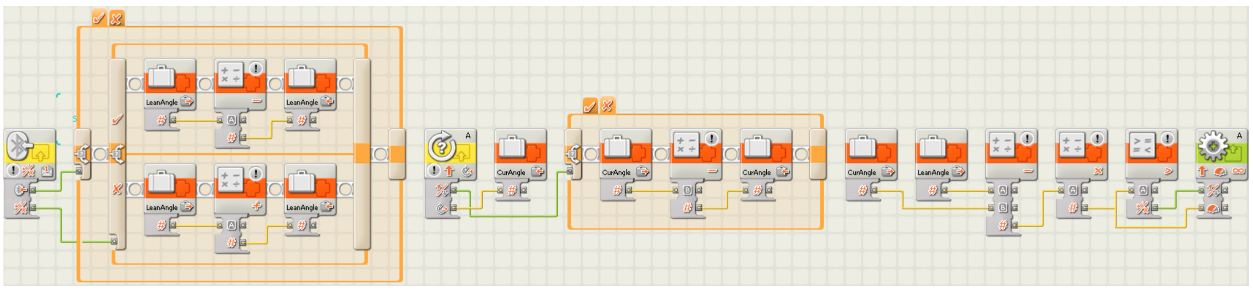


Figure 05– Light-Segway rider control blocks via Bluetooth massages.

Appendix 6 K-way Block by block description of the program.

Calibration Block

1. Calibrates the gyroscope offset by reading values for 0.5-1 seconds and calculating average value of those readings. The average of the used sensor is approximately 605.5 it can change in a range from 600-606 but usually that is a bad sign. And it is better to recalibrate the sensor (the robot might moved a little during this second). (Full information of the process is shown to the user via display)Calibration algorithm is modified from the LabVIEW of the gyro offset program provided by HiTechnic at their page [24]

Figure 06.1 NXT-G Calibration Block

Start Block

2. Reads the calibration value and displays some information about the model. (particularly it says that this is a K-WAY v104576)
3. Gives time to set the robot straight, but later it was modified to work instantly on activation (It is needed for multi-program use and it is easier to put the robot straight and start a program.)

Figure 06.2 NXT-G Start Block

Position Block

4. Reads the positions and speed then finds speed determinant (v dt) and sends it to next blocks

Figure 06.3 NXT-G Position Block

Encoders Block

5. Reads data about the robot (similar to matrixes described in **section**) checks desired “theta” and y position and sends it further.
6. The data is not exactly the same as in tables, because of the counting method and small defect of the gyroscope

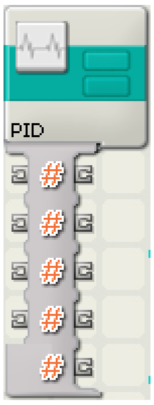
Figure 06.4 NXT-G Encoders Block

Gyro Block

7. reads the gyroscope value (current “theta”) and counts expected one
8. Then sends them to the PID controller

Figure 06.5 NXT-G Gyro Block

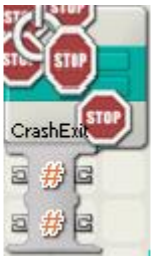
PID Block



9. All the data from previous blocks are gathered in a PID controller
10. then P -Proportional gain,
11. then I - Integral gain
12. Then D -derivative gain
13. the Error is counted
14. Then the sum of all inputs is counted U(t) according to the PID theory
15. Note that motor blocks of NXT are controlled by PID controllers, with low pass filters, that are included in some IDE (RobotC and LabVIEW)
16. the output is send to two blocks

Figure 06.6 NXT-G PID Block

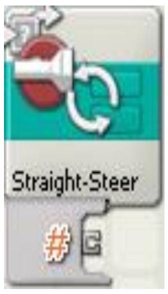
Error detection Block / emergency stop



17. Checks the maximum allowed acceleration (or “theta”), PID and controller output
18. If one of those settings is more then the allowed one: That means that the pendulum had felled down; or will soon be in an “unstabilazed” mode
19. Then the robot alerts the user by sound and stops the program.
20. If everything is OK, then the system will continue to stabilize the pendulum.

Figure 06.7 NXT-G emergency stop

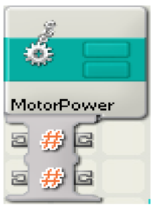
Straight Steer



21. The rotation of robot is set to zero and it is balanced on a linear line
22. This block counts the rotation of motors and remembers them to help synchronize motors B and C
23. the motors are driven by variables “Speed” that controller forward/backward motions of the motors and “Steering” that controls the rotating of our robot
24. The current steering number is compared to the desired one (desired for now and near future)
25. it also contains low pass filter (in some cases)

Figure 06.8 – NXT-G Straight Steer

Motor Power Block



26. The final filtered PID value is combined with the “Steer” value and motors B and C are synchronized for the final time.
27. finely the robot / inverted pendulum is starting / continuing stabilizing the entire system.

Figure 06.9 NXT-G Motor Power Block

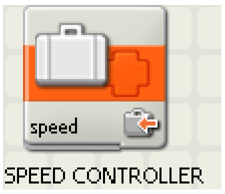
Wait Block

At the end of the loop:
28. The time is checked and d(t) is filtered (if needed).

Figure 06.10 NXT-G Motor wait Block

Appendix 6

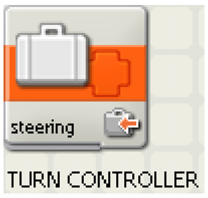
Speed controlling Block



- This is the main variable to control direction and speed of linear movements (back and forth)
- Changing those variables makes an inverted pendulum on to move in a two directions with out loosing balance (if it is not more then 80 or less then -75)

Figure 06.11 NXT-G speed controlling block

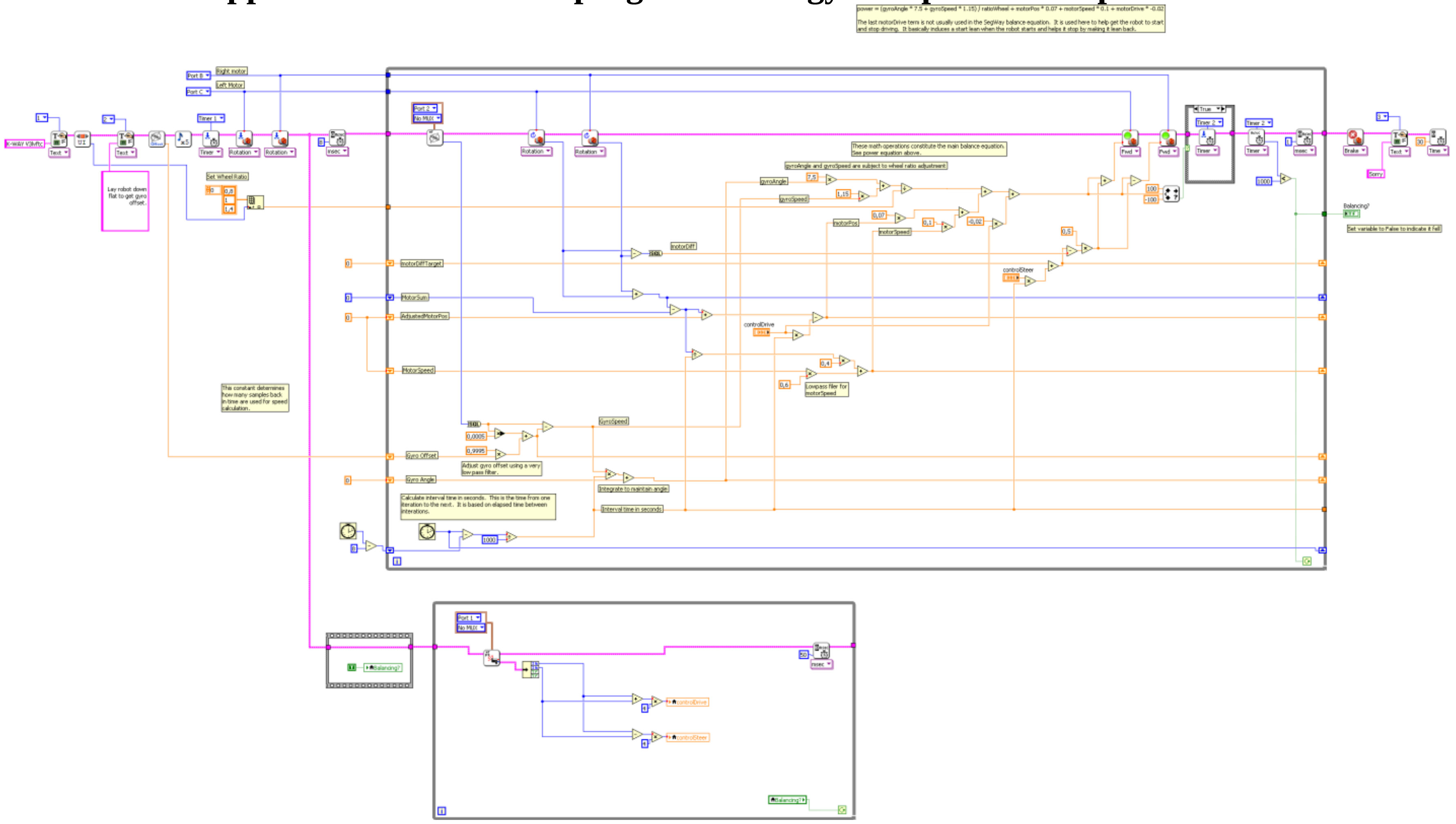
Steering controlling Block



- This is the main variable to control turns and rotation speed (Left or right)
- Changing those variables makes an inverted on a to turn to any direction with out loosing balance

Figure 06.12 NXT-G steering controlling block

Appendix 7 – LabVIEW program of the gyroscope inverted pendulum


power = (gyroAngle * 7.5 + gyroSpeed * 1.15) / ratioWheel + motorPos * 0.07 + motorSpeed * 0.1 + motorDrive * -0.02

The last motorDrive term is not usually used in the SegWay balance equation. It is used here to help get the robot to start and stop driving. It basically induces a start lean when the robot starts and helps it stop by making it lean back.

These math operations constitute the main balance equation. See power equation above.

gyroAngle and gyroSpeed are subject to wheel ratio adjustment

Lay robot down flat to get gyro offset.

This constant determines how many samples back in time are used for speed calculation.

Adjust gyro offset using a very low pass filter.

Calculate interval time in seconds. This is the time from one iteration to the next. It is based on elapsed time between interations.

Set variable to False to indicate it fell

Figure 0.7 – K-Way for LabVIEW