¡¡¡¡¡¡¡ HEAD ======= ¿¿¿¿¿¿¿ 8cd9d9c4e7856b1a91b6aa08a3306ffeed634cd0
¡¡¡¡¡¡¡ HEAD ======= ¿¿¿¿¿¿¿
8cd9d9c4e7856b1a91b6aa08a3306ffeed634cd0

Design and Control of a Lego Unicycle

Group B: Johan Kellerth Fredlund — Koshin Aliabase
Santiago Castro Chans — Sadik Kenan Sulejmanovic

January 11, 2015

Abstract

Your abstract.

Contents

1 Author's note

2 Introduction

We are designing a unicycle made by lego with an integrated inertia wheel to keep the lateral balence. The project therefore consists of two problems. To stabilize the unicycle in the medial direction by designing a controller for the ground wheel, and to stabilize it in the lateral direction by applying a reaction wheel connected to the top of the unicycle.

3 Construction

3.1 Ev3

The EV3 is the third generation of Lego Mindstorms platforms, the specification of this platform can be seen in the table 1. Furthermore some options for programming platforms are listed in the table 2. Lastly can a picture of the EV3 also be seen in figure 1

3.2 Sensors

3.2.1 HighTechinic Gyro Sensor

The Gyro sensor gives the rate of change in x, y and z orientation in radians per second. More specified details about the sensor can be seen in the table

Processor	ARM9
OS	Linux- based
Sensor ports	4 , analog, digital (up to 460.8 Kbit/sec)
Motor port	4, with encoders
SD- card	Micro SD- Card Reader, (up to 32 GB)

Table 1: Specifications for the EV3 platform found at [3].

Programing platforms
Lejos (extension of JAVA)
RobotC (similar to C)
NCX (similar to C)

Table 2: Some of the optional programing platforms for the EV3.

3. The first row shows how long time it takes for our program to fetch a sample, second row shows how accurate the sensor is given by specification by the manufacturer and the last line shows the update time of the sensor also given by the manufacturer.
In nature all gyroscopes has a drift and offset, both are unwanted. The drift means that the rate of change will increase with time even though the sensor is stationary as can been observed in the figure 3. Secondly the offset is the property that give a constant value although the sensor is stationary. How we dealt with these problems will be explained in later sections. Lastly can we see the gyro sensor in figure 2.

Refresh time in program	10 ms (average)
Accuracy	± 1 degree [5]
Update of value	300 times / second [5]

Table 3: Specifications for the gyro under the name NXT Gyro Sensor (NGY1044).

Figure 1: Shows how the hardware platform EV3 looks like which was used in the project.

Figure 2: Shows the gyro sensor which was used in the project.

3.2.2 HighTechinic Accelerometer Sensor

The accelerometer measure acceleration in x, y and z direction relative to the sensors specified directions given by the manufacturer. The specifications of the accelerometer can be seen in the table 4 which shows the the time for it to give a sample to our program respectively number of times the sensor can update a value. On the other hand this sensor also comes with unwanted properties which is that the value is (very) noise however when it is not noise it gives reasonable good data. We can also see how this sensor looks like in the figure 4 and the refresh time for a longer of time can be seen in the figure 5

Refresh time in program	20 ms (average)
Update of value	100 times / second [4]

Table 4: Specifications for the accelerometer under the name NXT Acceleration (NAC1040).

3.2.3 AbsoluteIMU-ACG

The AbsoluteIMU-ACG can gives gyro, accelerometer and compass data but in this project we are only interested in the gyro and the accelerometer therefor we will only discuss those measurements. As discussed previously gives the gyro part the rate of change in x,y and z. The accelerometer part measures the gravity in the x,y and z direction. The interesting part of this sensor is the refresh time versus the previous sensors and the update time can be seen in the figure 6. With this sensor one can also choose to have a smoothing filter for the

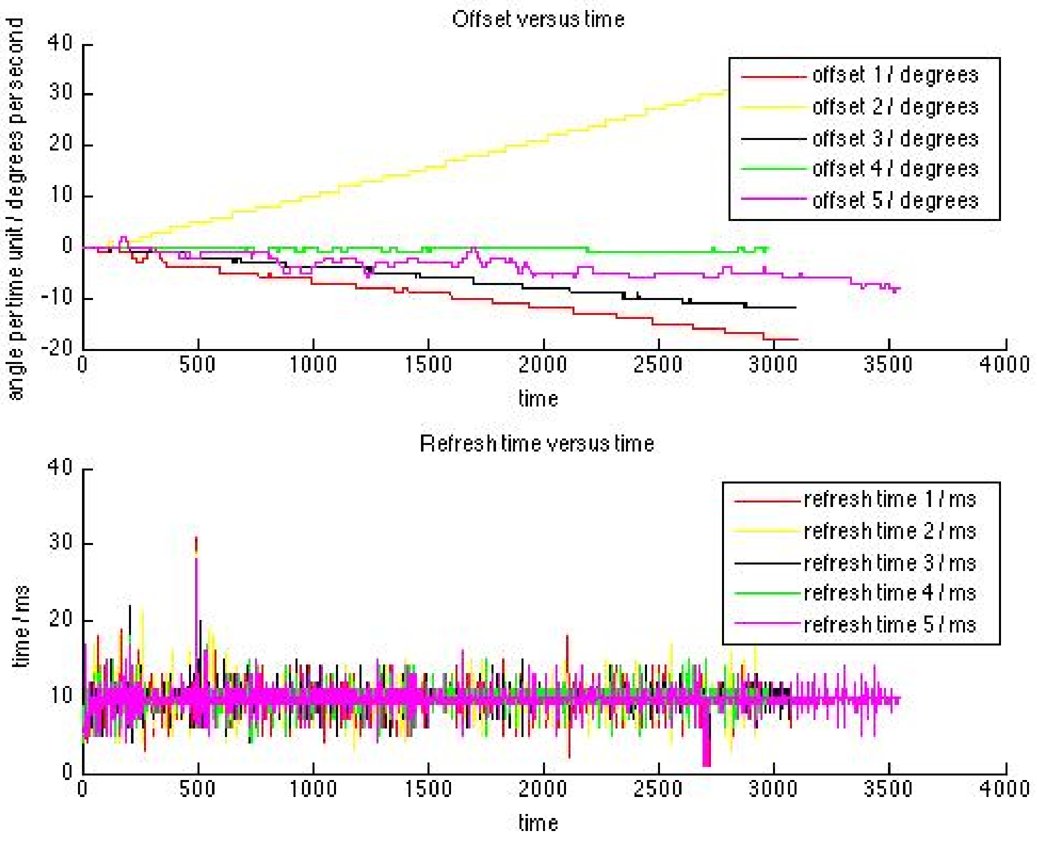

Figure 3: Shows the gyro sensor's offsets and refresh times when beeing stationary.

Figure 4: Shows the accelerometer sensor which was used in the project.

accelerometer part of the IMU, downside is that this will increase the refresh time of the sensor. Therefor we did two measurements, one with sensitivity 0 and one with sensitivity 4 (figure 7).

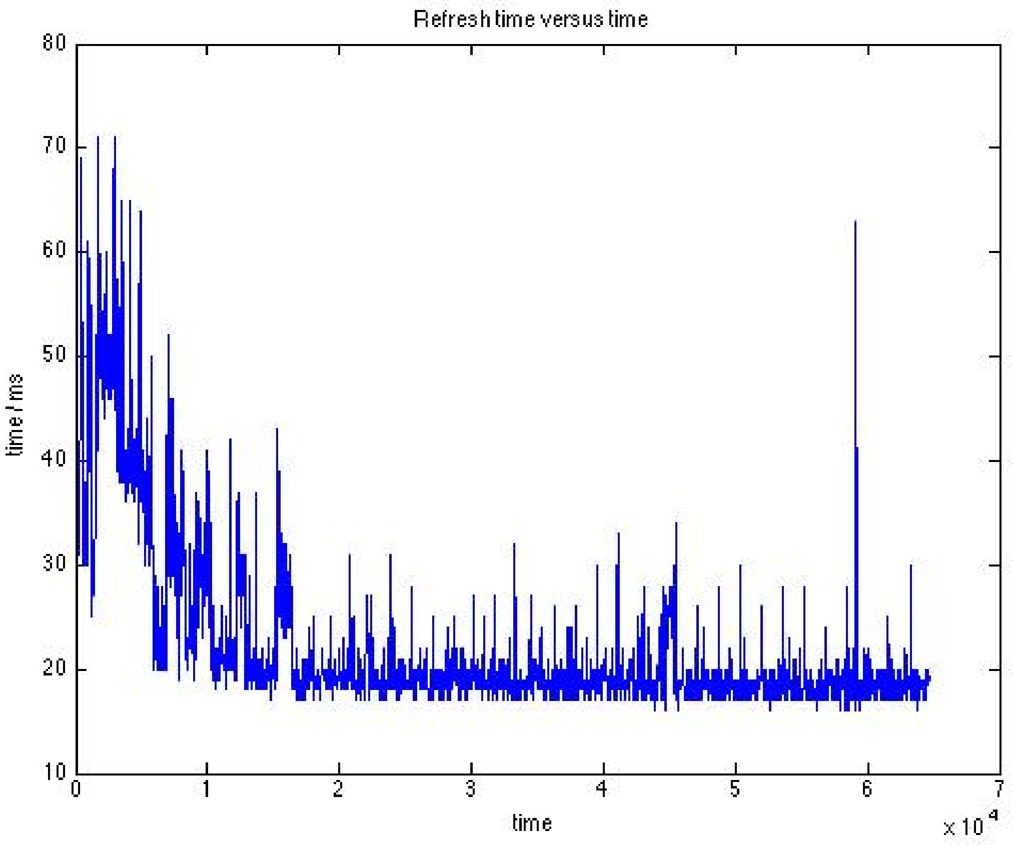

Figure 5: Shows the accelerometer sensor refresh time.

Refresh time in program	83 ms (average)
Update of value	100 times / second [1]

Table 5: Specifications for the IMU-ACG sensor.

3.3 Reaction wheel

The reaction wheel modeling is explained in the section 4 and some conclusion can be drawn from that. In this section we will bring up what those conclusion became in reality. Because we want as much weight as possible in the outer ring so nuts screwed in the material was the obvious choice. The choice of material was between plastic and wood, because of the nature of time and availability of personal it became wood. Plastic was preferable wood would also make it work. Due to the heavier weight of wood due to its nature it became a problem but was solved with making circles in the circle of wood so the most of the weight was taken out but leave enough for the circle not to crack under pressure of acceleration. The end product of the reaction wheel can be seen in the figure 9 while the dimensions is shown in the table 6. Meanwhile the design of the initial reaction wheel be seen in the figure 8. Thirdly is wort mentioning that

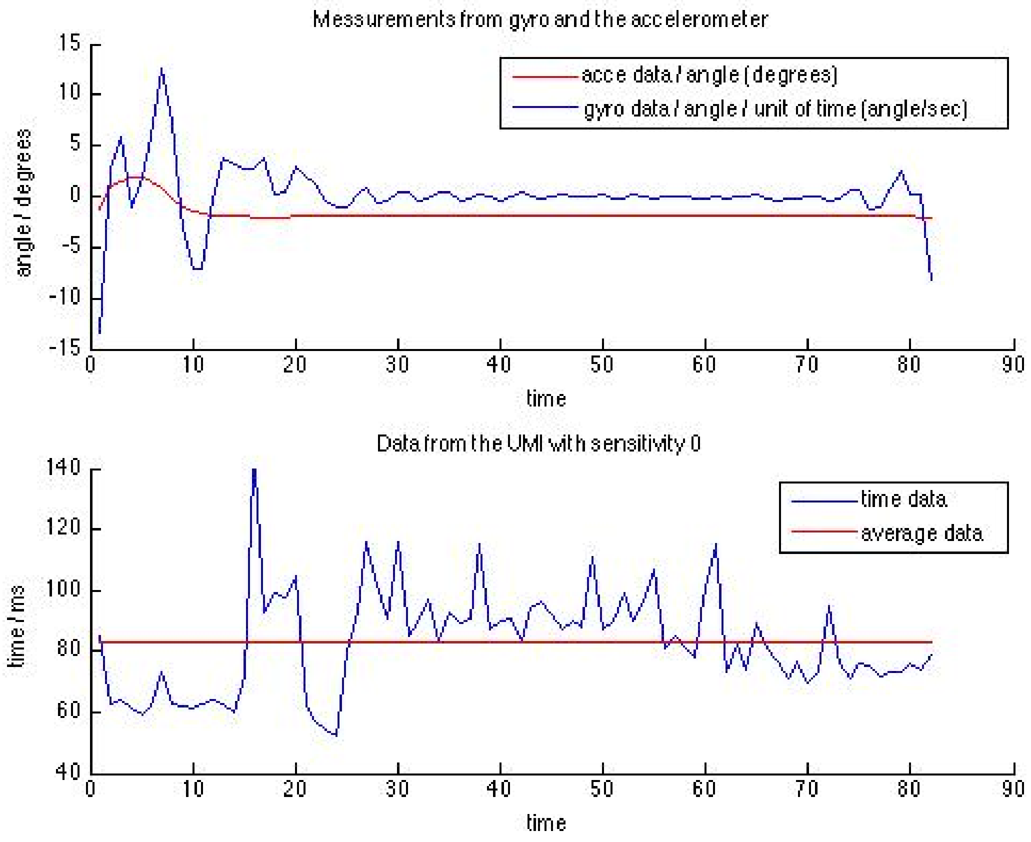

Figure 6: Shows the important properties of the IMU sensor with sensitivity 0.

the wholes at the end of the circle in figure 8 is for the nuts to go in and the cross in the middle is for the attachment to the axis going to the EV3 Large Servo Motor. More on this topic under the section 3.3.1

Diameter	30 cm
Thickness	8 mm

Table 6: Shows the dimensions of the end product of the reaction wheel.

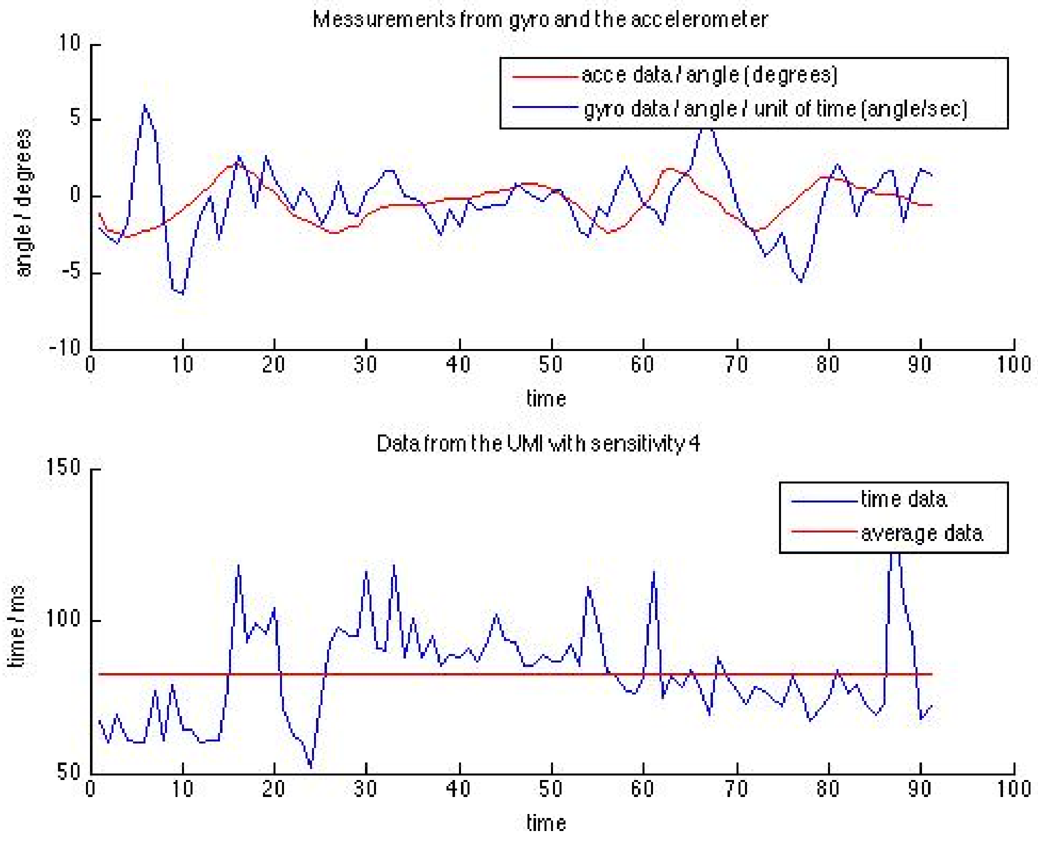

Figure 7: Shows the important properties of the IMU sensor with sensitivity 4.

3.3.1 Attachment between the reaction wheel and EV3 Large Motor

Due to the attachment of the sensors on the current prototype we build in one small circular block to the reaction wheel which also had wholes matching the attachment directly to the EV3 Motor attachment. The specific attachment pattern is shown in figure 10. By using that specific pattern we achieved more stability of the reaction wheel but still there were problems with leaning of the reaction wheel relative to the full scale prototype. We also experienced wobbling of the reaction wheel when it was set to use. Therefore we rebuild the reaction wheel once more relative to the initial design with one more circular block on the other side of the reaction wheel which meant that we now had two circular blocks coming out from the reaction wheel at both ends. The prototype holding up the reaction wheel was also rebuild due to rebuilding of the reaction wheel, more on rebuilding of the full prototype can be seen in section 3.5. Thus in the end leading to a full stable reaction wheel relative to the full scale prototype. The end attachment of the reaction wheel is shown in figure 11

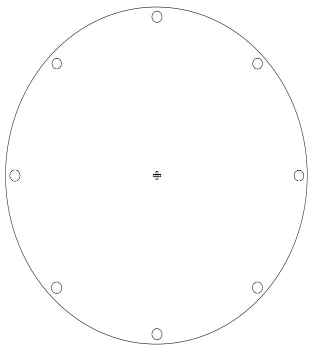

Figure 8: Shows the initial design for the reaction wheel.

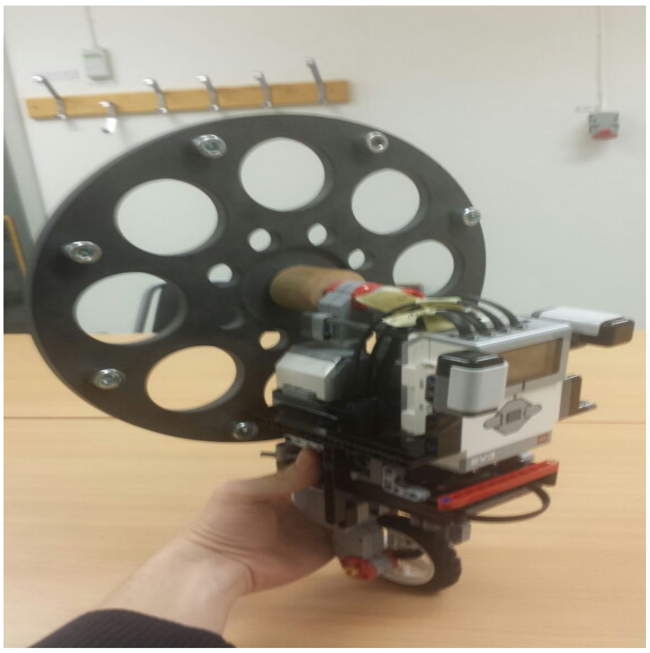

Figure 9: Shows the end product of the reaction wheel with the whole prototype at one stage.

3.4 Motors

3.4.1 EV3 Large Servo Motor

This is one of the strongest motors there is for LEGO and the specifications for this motor can be seen in the table 7. One experiment was done to get the connection between the torque and the power which is the parameter that control the PWM that goes to the motor. The power goes from -100 to 100. The result can be seen in the figure 12.

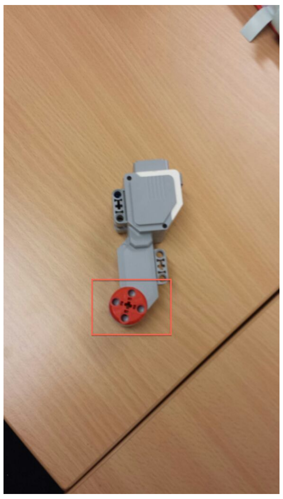

Figure 10: The red rectangle specifies the attachment pattern on the EV3 motor.

RPM	160 - 170 [2]
Running torque	20 N/cm [2]
Stall torque	40 N/cm [2]

Table 7: Specifications for the EV3 large motor server.

3.5 Prototypes

In this section we will bring up the limitations of the physical construction of the lego segway and bring up some key limitations of the model as well.
The most critical limitations is the building blocks of Lego because of some key concepts. The first being that the attachment pieces which holds the lego blocks together are not the most stable one which means that is you put two blocks together you can not expect that the blocks are held together while have some stress on them. They will fold not all the way by enough for the prototype to experience wobbles when the reaction wheel starts to accelerate and move around. This problem was experienced with great signification in the three first prototypes but the last one held it out pretty well considering that the system was made in lego.

The second limitation is availability of the connection points on the EV3, there are some but could be more of them. From a building perspective this set great boundaries on how you might build your prototype hence the four prototypes. Each of them had strengths and weaknesses which is brought up in the table 8. The possible options that the different categories can be are shown in the table 9, these options are taken up to give a hint of how the different prototypes act when used and are therefore not scientific in a since that you can measure them.

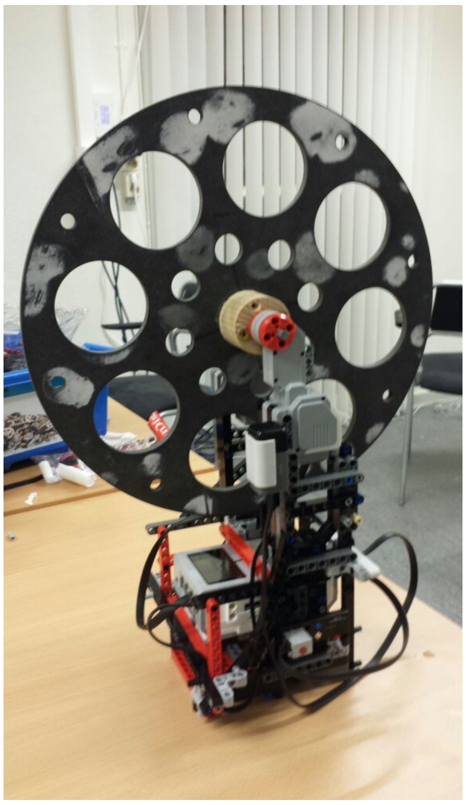

Figure 11: Shown the end product of the attaching of the reaction wheel with the two engines.

Categories	p1	p2	p3	p4
Handle stress due to reaction wheel	-	falling apart	bad	good
Stationary movement	-	forward	weak forward	stationary
Gravidation centre	middle	high	high	low
Reaction wheel torn out	-	yes	yes	no

Table 8: The prototype properties for different categories, where pN is as follows; the p for prototype and the N for which one (e.g. p1 stands for prototype 1).”-” means that is was never tested.

Categories	The possible options
Handle stress due to reaction wheel	falling apart-bad-magageable-good
Stationary movement	falling forward-falling backward-stationary
Gravidation centre	low-middle-high
Reaction wheel torn out	yes-no

Table 9: The different options that the categories can take in the table 8.

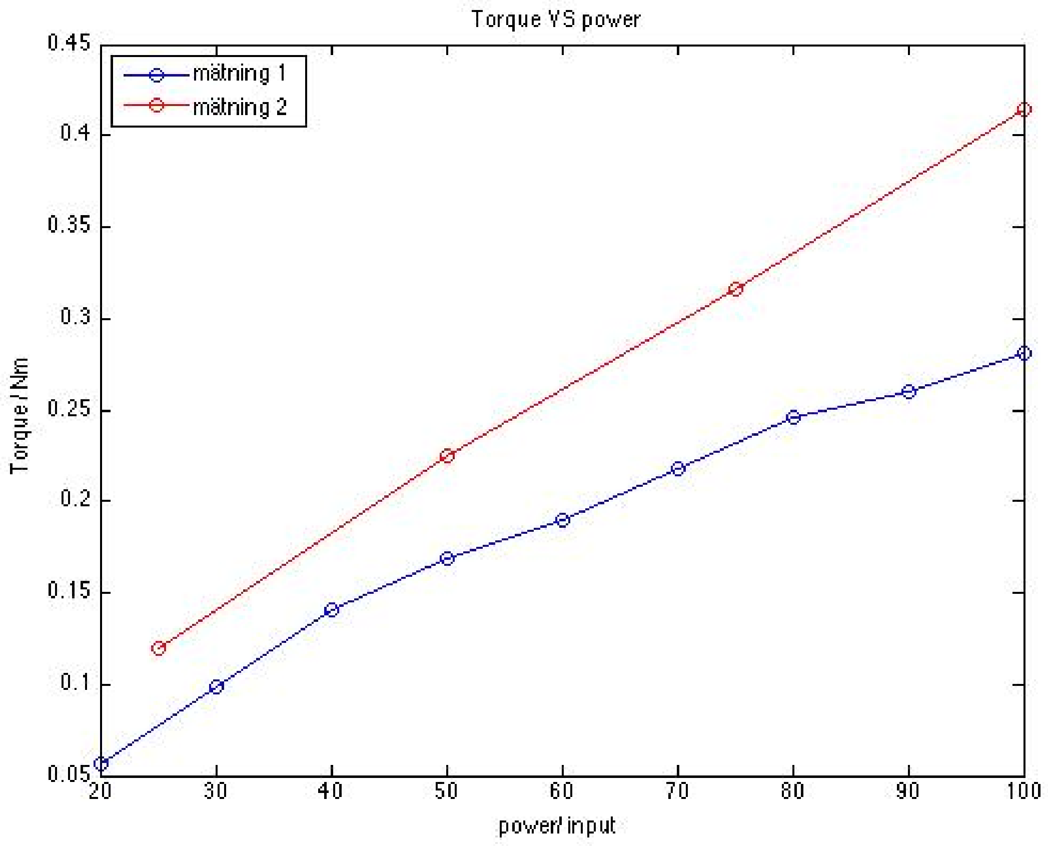

Figure 12: Shows the relation between the power and the torque of the motor.

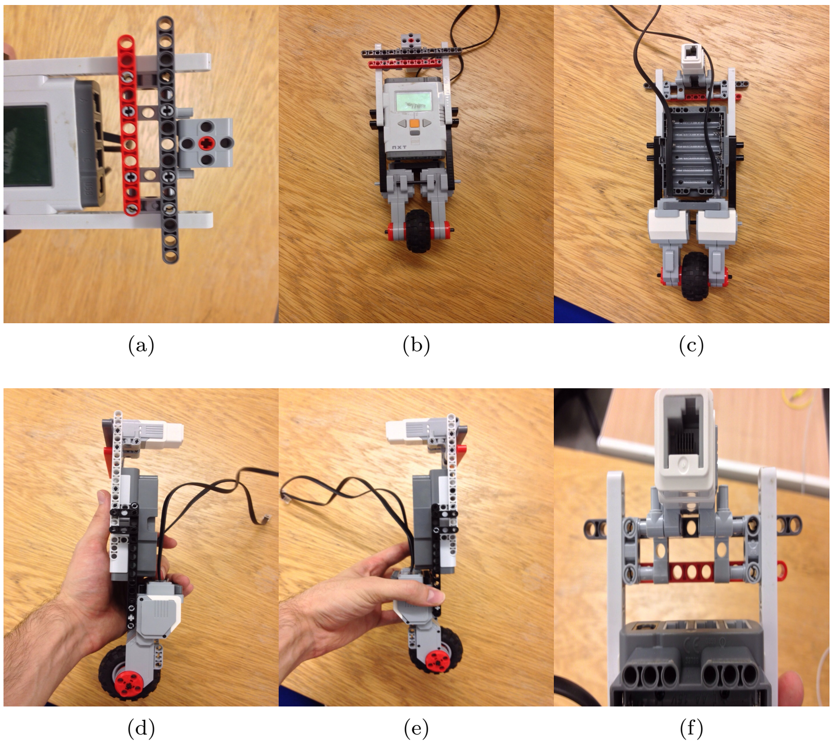
(a) (b) (c)
(d) (e) (f)

Figure 13: Shows the first prototype from different angles.

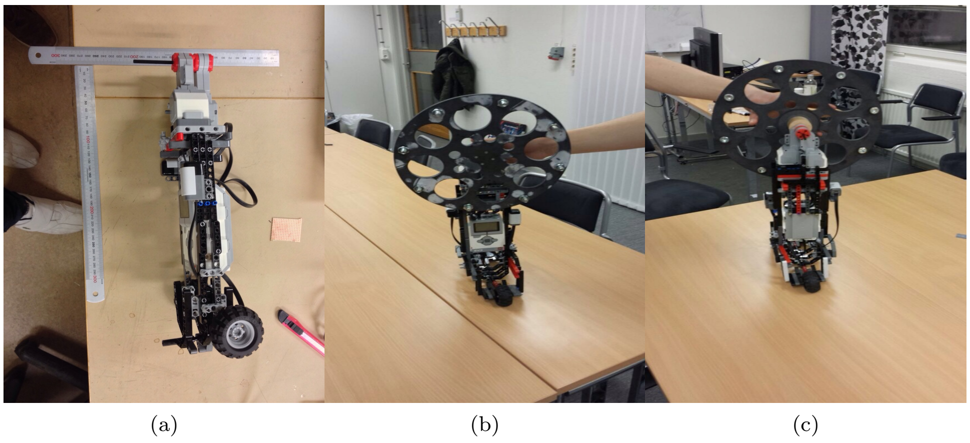
(a) (b) (c)

Figure 14: Shows the second prototype from different angles with the first prototype of the reaction wheel.

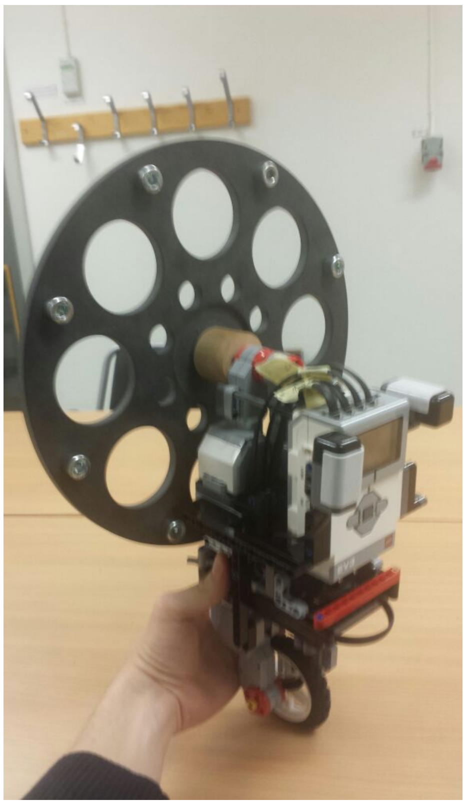

Figure 15: Shows the third prototype with the first prototype of the reaction wheel.

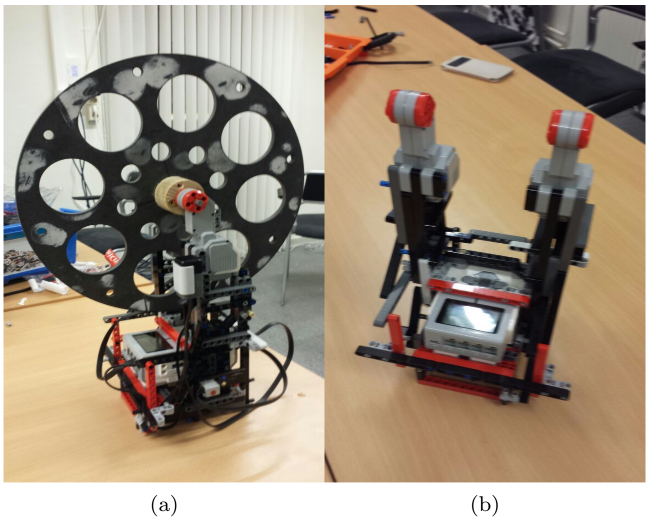

Figure 16: Shows the fourth prototype from different angles with the first prototype of the reaction wheel.

4 Modeling

4.1 Lateral Balance

We realized early in the project that controlling the inertia wheel is the most complicated part of the project. The complexity comes from several areas. The sensors have to be very accurate (which they are not), the simulation in simulink had to take account of the relationship between torque and the speed of the wheel among other things, we had to design a working controller.

The unicycle is similar to an inverted pendulum with a point mass in the center of mass, with an additional mass that is the inertia wheel, seen in figure 17. We modelled the pendulum as can be seen in (1). The relevant states from the model are the angle of the pendulum, the the angular velocity of the pendulum and the angular velocity of the inertia wheel. The angular position of the inertia wheel is not important as only the change in angular position affects the torque. Torque is the momentum you get when a vector force is acted on the wheel with a certain radius. The torque acting on the pendulum due to gravity (which makes it fall) must be less that the torque from the inertia wheel (which we can control) for it to be stable. The reaction torque is therefore dependent on the radius of the wheel. The maximum torque of the engine is also limited, so some angles are impossible for the reaction wheel to stabilize.

To optimize the controller we implemented a linear quadratic regulator which minimizes the cost function, seen in (2). The Q matrix determines how much each state will be penalized. Since we want to keep cost function (J) small a large Q will require that the states are small. Q=[q1 0 0; 0 q2 0; 0 0 q3].

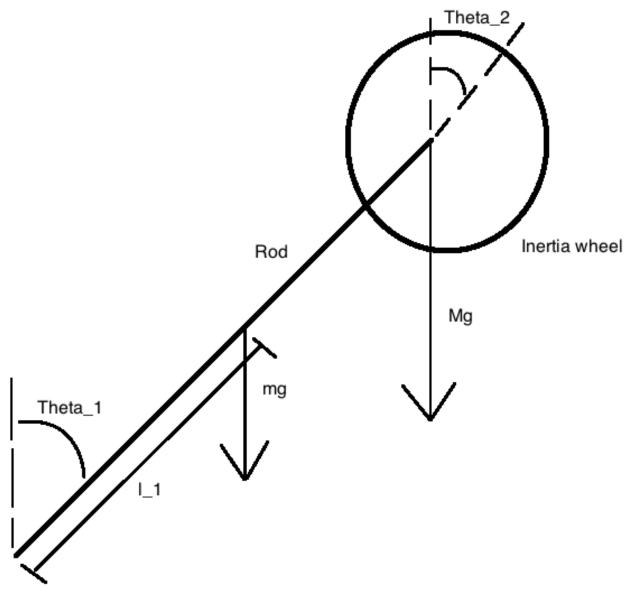

Figure 17: Shows the sketch of the inverted pendulum

Mathematical model

The principle idea of this control problem is to receive a reaction torque from the inertia wheel that is larger than the torque exerted on the unicycle from gravity. Looking at figure 17 we see the forces acting on the unicycle as it falls. If we assume that all the mass of the body exists in one point, the center of mass, the model of the process can be calculated using Newtons laws. The use of Newtons laws was chosen instead of the Lagrangian equations because it would give an adequately accurate model while still being easier to determine.

The angular acceleration of the body will equal the net torque acted on the center of mass and the center of the inertia wheel due to gravity, and the reaction torque opposing the other two. The reaction torque is caused by the angular acceleration of the inertia wheel. Controlling the reaction torque in such a way that will keep $\frac{\partial^2\theta_b}{\partial t^2} = 0$ and $\theta_b = 0$ will make the system stable. It was estimated that the friction from the EV3 motor would we small and it was neglected from the equations.

$$\begin{aligned}(J_w + ML^2 + J_b)\frac{\partial^2\theta_b}{\partial t^2} &= l_1 \sin\theta_b mg + L\sin\theta_b Mg - \tau \\ J_w\frac{\partial^2\theta_w}{\partial t^2} &= \tau\end{aligned} \tag{1}$$

Where θ_b is the angle of the body, θ_w is the angle of the inertia wheel, $J_(b)$ and $J_(w)$ is the moment of inertia of the body and the wheel, m is the mass of the body without the wheel, M is the mass of the wheel, g is the gravitational constant, l_1 is the length from the pivot point to the center of mass, L is the length of the pivot point to the center of the wheel and τ is the reaction torque.

As stated before the interesting states are the angle of the body, the angular speed of the body and the angular speed of the wheel. The model in (1) can be linearised around the origin where $\theta_b = 0$. When the angle of the body is close to zero $sin(\theta_b) \approx \theta_b$. Since the angle of the body will mainly be close to zero with the right controller the approximation will serve for the model. The new model and state space representation linearised around the origin will be:

$$\begin{aligned}(J_w + ML^2 + J_b)\frac{\partial^2\theta_b}{\partial t^2} &= l_1\theta_b mg + L\theta_b Mg - \tau \\ J_w\frac{\partial^2\theta_w}{\partial t^2} &= \tau\end{aligned} \tag{2}$$

x_1: Angle of the body
x_2: Angular velocity of the body
x_2: Angular velocity of the inertia wheel

$$\begin{aligned}\dot{x_1} &= x_2 \\ \dot{x_2} = \frac{g}{(J_w + ML^2 + J_b)}(l_1 m + LM)x_1 - \frac{1}{(J_w + ML^2 + J_b)}\tau \\ \dot{x_1} &= \frac{1}{J_w}\tau\end{aligned} \tag{3}$$

$$A = \begin{bmatrix} 0 & 1 & 0 \\ \frac{g}{(J_w+ML^2+J_b)}(l_1 m + LM) & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ -\frac{1}{J_w} \\ \frac{1}{J_w} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$D = 0$$

Simulink model

4.2 Medial Balance

4.2.1 Mathematical model

4.2.2 Simulink model

5 Implementation

5.1 Parameters

5.2 Programs

In this section we will consider looking at the different programs and why the specific formats and programs where chosen as can bee seen in table 10. The OS (operating system) windows 7 was chosen due to the easiest setup guides on the internet for Lejos (the main programing language). Lejos for main programing language was mainly used due to the familiarity of java, Lejos is a extension of java. It also has a class for every unit which was used in the project in other words made it easy to work with. Lejos had also support for parallel threads which other programs had diffucuilities

OS	Programing language	GUI
Windows 7	Lejos [5]	Eclipse [2]
Communication GUI	Communication format	Communication medium
Winscp [8]	txt -format	Bluetooth
Data- caption	-	-
DataLogger [9]	-	-

Table 10: Shows the programs and important formats which was used in the project.

5.2.1 Lejos

5.2.2 NXC

5.3 Sensors

6 Results

7 Discussion and Conclusion

A Lejos Source Code

A.1 The main source code

A.2 The sensor source test codes

References

[1] Absolute imu sensor. `http://www.mindsensors.com/index.php?module=pagemaster&PAGE_user_op=view_page&PAGE_id=158`.

[2] Eclipse luna. `https://eclipse.org/home/index.php`.

[3] Ev3 large servo motor. `http://shop.lego.com/en-US/EV3-Large-Servo-Motor-45502?fromListing=listing`.

[4] Ev3 platform specifications. `http://botbench.com/blog/2013/01/08/comparing-the-nxt-and-ev3-bricks/`.

[5] Lejos ev3, java for lego mindstorms. `http://www.lejos.org/ev3.php`.

[6] Nxt acce. `https://www.hitechnic.com/cgi-bin/commerce.cgi?preadd=action&key=NAC1040`.

[7] Nxt gyro sensor. `https://www.hitechnic.com/cgi-bin/commerce.cgi?preadd=action&key=NGY1044`.

[8] Winscp free sftp and ftp client for windows. `http://winscp.net/eng/index.php`.

[9] Ole Caprani. Datalogger version 4.02.13. `http://legolab.cs.au.dk/DigitalControl.dir/NXT/src/DataLogger.java`.