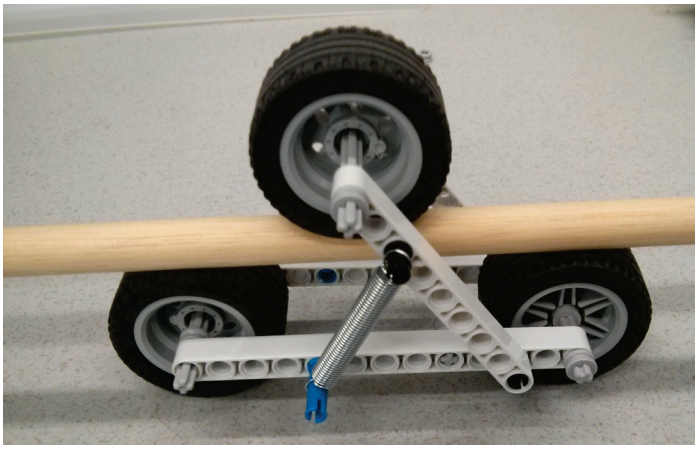
DB212 - Lego Beyond Toys

By Pepijn Fens (M2.1)

The module focused on combining the mechanical, electronical and intelligent aspects of products into something new. Lego offers the perfect platform for this, allowing people tinker with various principles while keeping everything "simple".

Mechanical Explorations

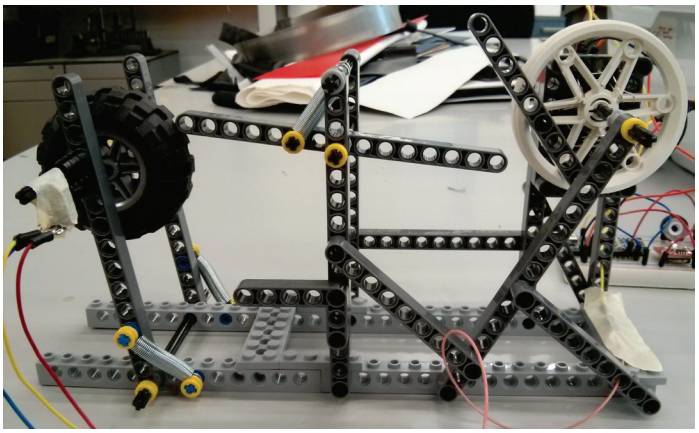
To learn about the basics of mechanics, some explorations where made using Lego Technic. We focused on force closure and transferring energy from one type to another.



Example of "force-closure"

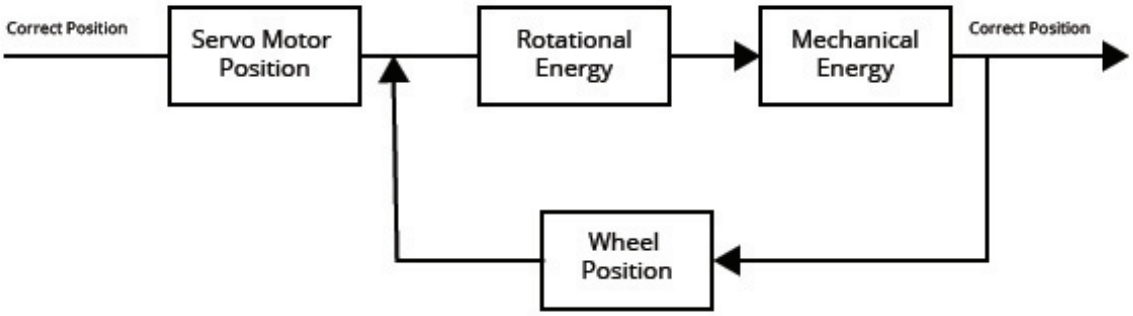
Feedback systems

Prof. Feijs introduced feedback systems, after which we altered the servo motor's feedback loop. This could be done in several ways. We decided to explore energy transfers further, and created an energy transferring machine. This exercise helped understand how feedback systems work, and what is meant by introducing "noise" into a feedback system.

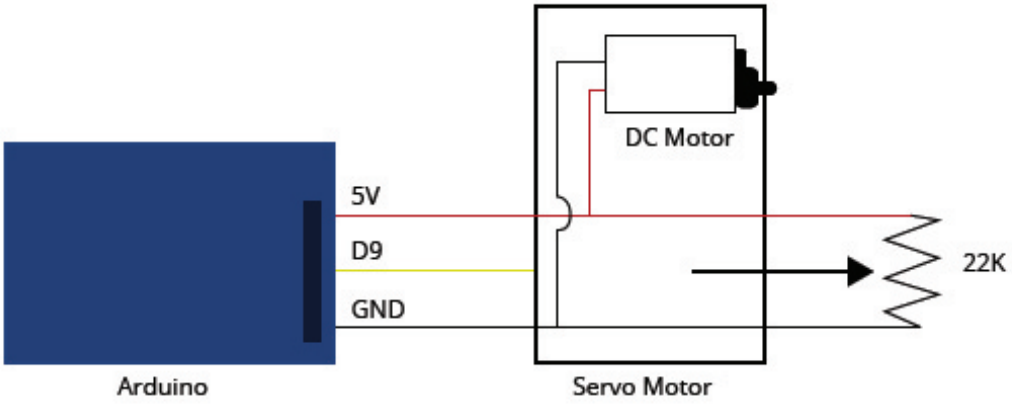


A servo feedback loop machine. The feedback point for the servo is at the end of the machine, while the motor is at the beginning.

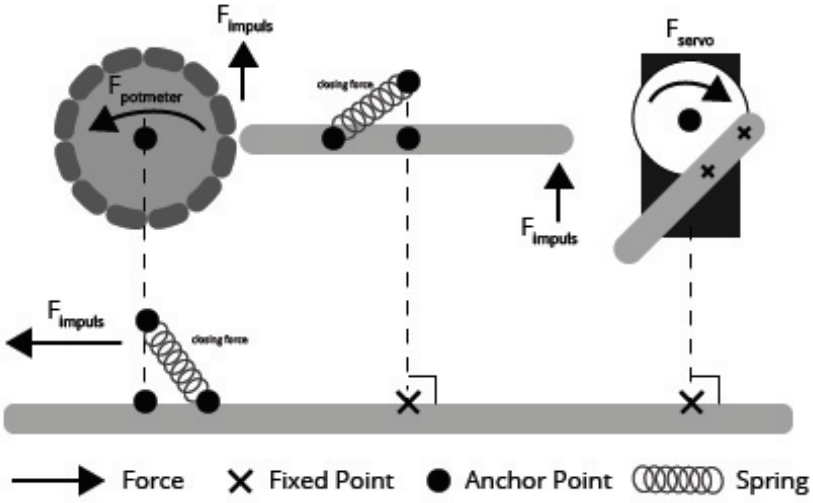
High Level Overview (tentative)



Electronic Overview



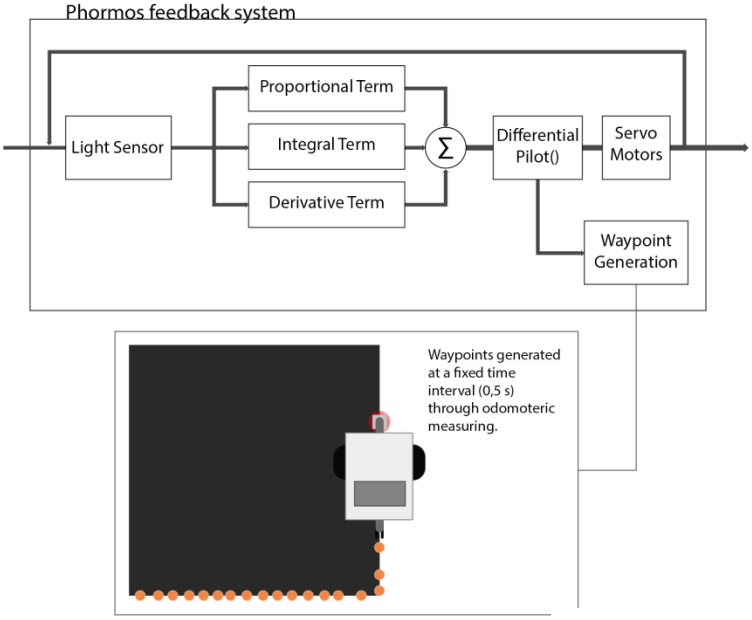
Mechanic Overview

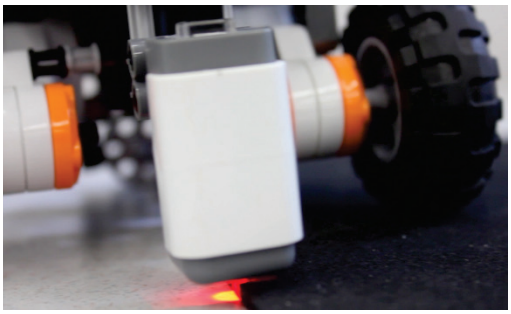


Module goal

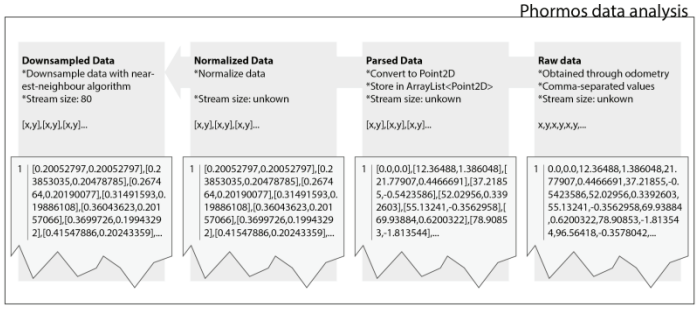
The aim of this module is learning about integrated mechanical and intelligent systems. We have done this by building an intelligent geometric shape sensor. We'll achieve this by having a robot trace a shape using a line following algorithm, while keeping track of his position. Next we want to send the "trace" of the shape to a neural network to classify the various shapes.

Phormos feedback system

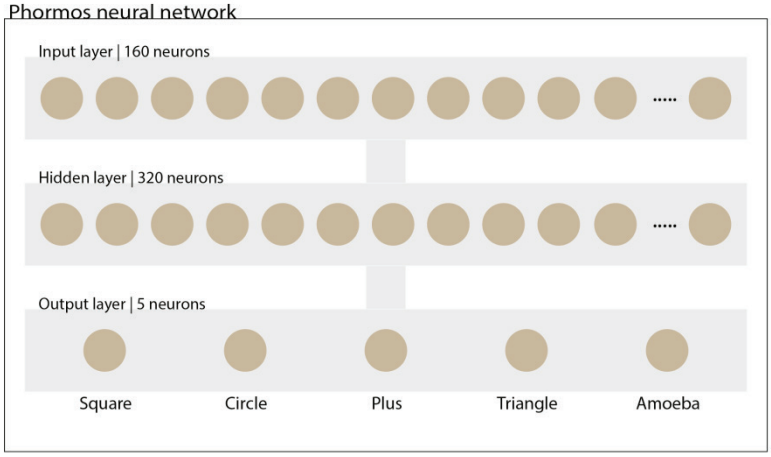




Phormos data analysis



Phormos neural network



Building the robot

Construction

The robot is based on a sturdy construction with the weight centered and a low center of gravity. A rear mounted castor wheel is included to facilitate a tight turning circle. For the line-following purposes, a height-adjustable light sensor is mounted on the front. Tests to determine the ideal position for following a line and recording position of the robot (odometry) pointed out that the sensor should be slightly in front of both wheels.

Sensing & Actuating

The basic movement of the robot is facilitated through the DifferentialPilot class available in the LeJOS NXJ library. This class keeps track of the movements of the robot over time. Using odometry we were able to retrieve an estimation of position (waypoint) each 0.5 seconds. These waypoints are collected in a file on the NXT brick.

The line following has been done using the PID principles to ensure optimal shape following. PID for a line-following robot using Lego NXT has been described online (Sluka, 2009).

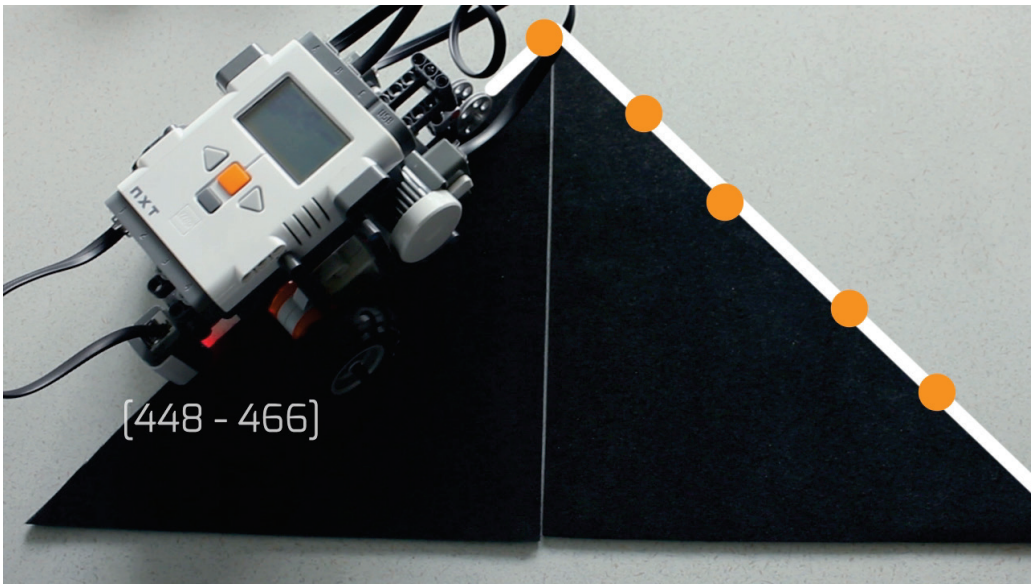
The following java code was uploaded to the NXT brick to facilitate the PID process.

```
//PID variables
double Kp = 40; //Error multiplier
double integral = 0; //integral, helps stabilizing the system over time
double Ki = 0.05;   //integral multiplier
double lastError = 0;
double Kd = 0.01; //derivative multiplier, helps smoothing out movement

int l = light.getLightValue(); // value from 0 to 100
double error = l-offset;
integral += error;
double derivative = error - lastError;

//PID
double turn = (error*Kp) + (Ki * integral) + (Kd * derivative);
pilot.steer(turn); //Send the value to the wheels

lastError = error;
```



The robot follows the shape and creates "waypoints" in periodic intervals

Data Communication

Tests with real-time Bluetooth communication were unfruitful because of the unreliable connection. The alternative was a locally generated and stored coordinate file, which is downloaded to the laptop through Bluetooth. This log file could then be used in two different ways. It could serve as training data, by adding the correct output after each node (supervised learning). Also, it could be used as validation data, by using the data from the file as input to the trained neural network.

Data preprocessing

Before the data can be put into the trained network, some preprocessing is required. This preprocessing consists of multiple steps.

Waypoints that are generated each 0.5 seconds introduce a problem. The amount of waypoints grows with the size of the object. Therefore the data needs to be downsampled. Downsampling means reducing the amount of points in a set while remaining representative for that set. We have used the simple Nearest Neighbor algorithm to reduce the points to 80 (Spoike, 2009).

Next to this, the data needs to be normalized. Normalization ensures that variations in size of followed shapes do not skew training and/or testing results. Specifically, we have mapped all shapes to fit into a coordinate space of 0-1, rather than the arbitrary coordinate space used by the odometry class in the robot.

Network Training

Integrating the network with our dataset has been done in Java. We've used the Neuroph (Neuroph, 2013) Java libraries to create the network and interpret the training files. The network has 5 output nodes, which means it has been trained to distinguish between 5 different shapes. These shapes were squares, circles, crosses, triangles, and complex "amoeba" shapes. The robot is trained by following clear cut examples of the shapes multiple times. For each of the 5 classes, we had more than 3 training sets available.

We have tried several different training configurations and found that the following configuration served our purposes best:

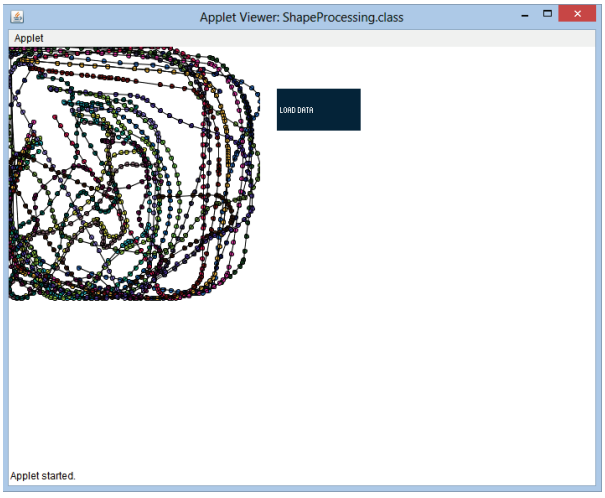
Input nodes	160
Hidden Layer	320
Output nodes	5
Training method	Resilient back propagation
Transfer function	Sigmoid
Learning Rate	0.5
Max Error	0.01

Network Testing

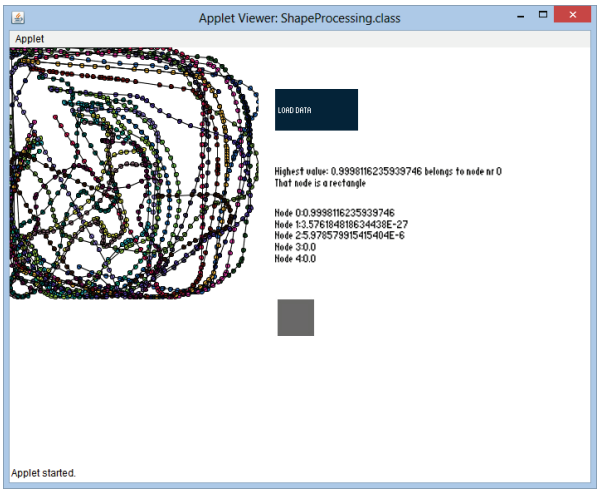
Testing the network is done by using waypoint data from the robot which isn't included in the test set and sending this through the neural network. The Java applet is written specifically to accommodate the testing of input files against the trained network. After training the network was able to classify newly recorded test data with accuracy up to 0.99.

Communicating Results

The results of calculating input data sets can be seen in a simple java program. This program plots the various test sets, and allows for selecting an input file to be calculated by the network. Next, it will select the highest output node and tell the user which shape corresponds with this node.



Interface with training paths displayed



Interface with new data loaded, and the result of the calculation (a square).

Reflection Pepijn

Basic Mechanic systems

The simple and clear explanation of mechanic systems helped me to understand the basics of mechanic systems. No other assignment or module introduced me to these systems, so this explanation occurred right on time. I knew about specific mechanics to use when to achieve movement, but I never wondered why. The theory provides a guideline in defining the correct constrains when designing a mechanical system. Working with Lego, these constraints were easy to construct. Lego formed a good connection to the theory.

Feedback Systems

The principles of feedback systems are at the heart of intelligent systems. During the module we received an in-depth explanation of feedback theory, showing me the power of feedback theory. We have integrated feedback systems in the final deliverable, for example, the Morphos robot utilizes a PID controller to follow the line. Without the introduction to these principles the robot wouldn't have these advanced line-following capabilities.

Together with the Basics of Mechanics, the feedback systems provide a powerful way to create interactive "moving" prototypes.

Communicating using block diagrams

Feedback systems work on the border between the physical and the digital world. On both the physical and digital side there are mechanisms (mechanical and software) to make the overall system work correctly. Block diagrams are a very useful way of visualizing this process. Using these simple cross sections of a system, people can quickly understand how a complex system works. Also, it is possible to make these diagrams layered, so that even more complex "blocks" can be hidden in a next layer.

Intelligent Systems

Building on the theory of mechanics and feedback systems, intelligent systems were introduced. I have worked with "neural networks" before, but remain difficult to wrap my head around. Because of the potential we decided to work with these "neural networks". This proved to be difficult since the tools available weren't specifically tailored for our purposes. We ended up writing specific software for training and calculating our custom neural network. This took by far the most time, but in the end delivered accurate results. Writing a program to read, write, send and receive data was just one of the challenges we encountered when working with neural networks.

Lego as a prototyping tool

Lego was the obvious prototyping platform, hence the name of the module. The famous plastic blocks and rods were no secret to me; I was already familiar with the capabilities of Lego. But I never worked with the NXT bricks before. I did work with Java before, which is used to program the bricks. The Lejos OS made it easy to write complex behaviors, but contrasts with the "fool-proof" and easy-to-use plastic parts Lego parts.

References

Spoike. (2009, October 20). How to write a downsampling function in Java. Retrieved September 23, 2013, from http://stackoverflow.com/questions/1594543/how-to-write-a-downsampling-function-in-java

Neuroph. (2013). Java Neural Network Framework Neuroph. Retrieved September 23, 2013, from http://neuroph.sourceforge.net/

Sluka, J. (2009). PID Controller For Lego Mindstorms Robots. *inPharmix*. Retrieved September 23, 2013, from http://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html