ROBOTTIOHJELMOINNIN HARJOITUSTYÖ (582326)

Dog Treat Thrower

Documentation for University of Helsinki's robotics course

This document includes specifications about dog threat throwing –robot, both on hardware- and software level.

Jaakko Virtanen, 014579093
jaakko.virtanen@cs.helsinki.fi

Contents

Overview

The purpose of this document is to be a final report and present the DogTreatThrower-robot, which I built for University of Helsinki's robotics-course (a short course during the Christmas time).

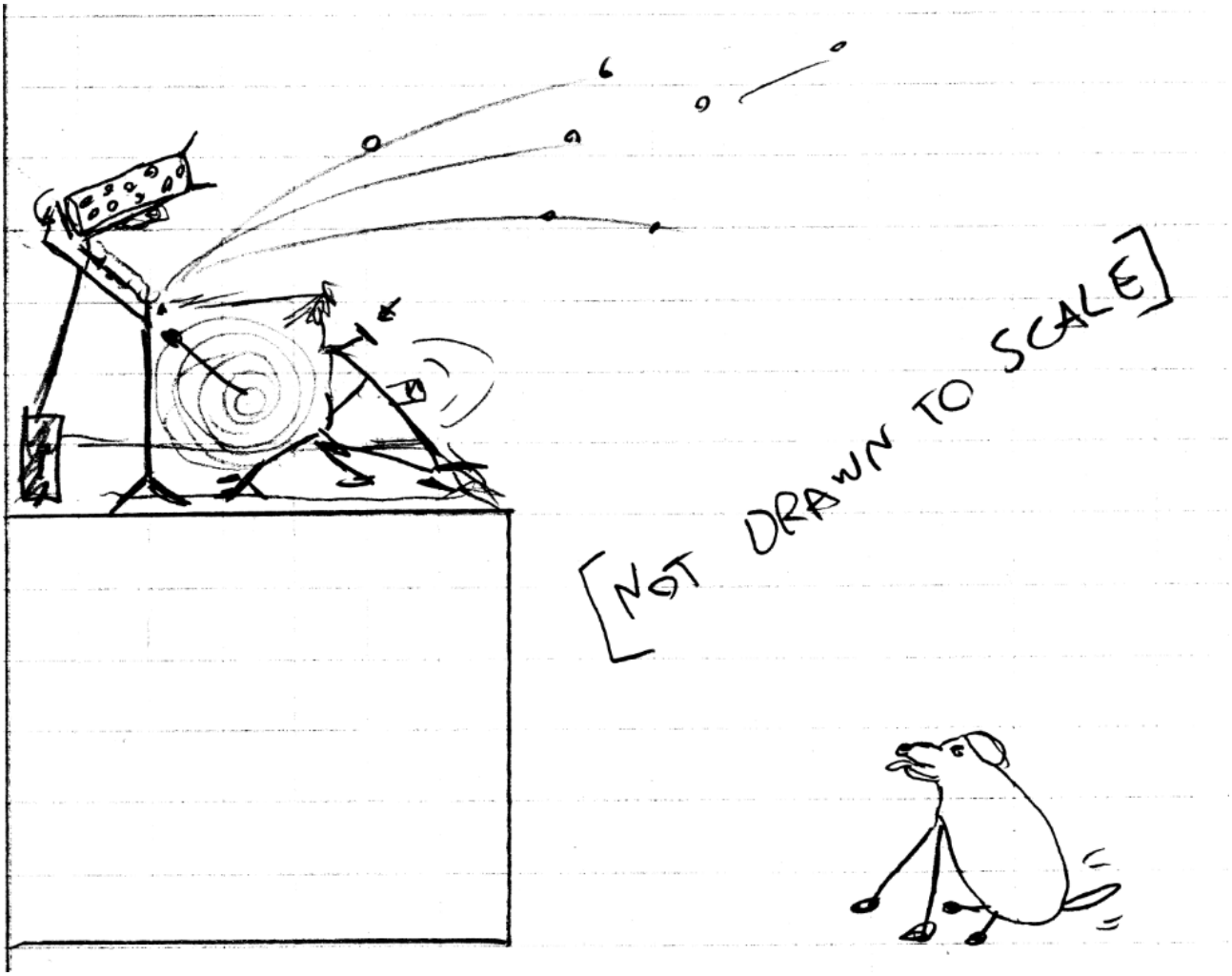
The document contains four main parts:

- Overview, the purpose of the robot
- Hardware specification
- Software overview - structure and design of the code
- User manual

What is the DogTreatThrower-robot?

Purpose of the robot

The main purpose of the robot is to keep the dog happy by throwing dog candies to the dog. The target is to throw ammos randomly at 0.5 - 3 meters away from the launchpad when triggered.



Picture 1. Robot and Dog

Functionality of the robot

The DogTreatThrower's basic user case is:

1. Dog treats are to be loaded manually to the tube (max. ~25 pieces)
2. Starting the program from the NXT-brick
3. Testing the sensors
   - 3.1. Pushing the SensorButton → the NXT-brick prints out text and the value of the SoundSensor
   - 3.2. After the test is OK, pressing NXT.LEFT to start the robot
4. The robot starts and prints short instructions
5. The user interacts with the NXT-brick e.g. to activate the ButtonTrigger
   - 5.1. The user press NXT.RIGHT to access the configuration mode
   - 5.2. The user press NXT.RIGHT again in configuration mode to activate the ButtonTrigger
   - 5.3. The robot announces whether the trigger was added successfully
6. The robot initializes all triggers and starts threads to them → The robot listens to all activated triggers and user commands
7. The user triggers the robot by pressing the SensorButton
   - 7.1. The robot performs the dog feeding routine and throws treats
8. The user wants to turn the VoiceTrigger ON
   - 8.1. The user press NXT.RIGHT to access configuration mode
   - 8.2. The user press NXT.LEFT to activate the VoiceTrigger
   - 8.3. The robot announces whether the trigger was added successfully
9. The robot initializes all triggers and starts threads to them → The robot listens to all activated triggers and user commands
10. The robot hears a sound over the VoiceSensor's limit
    - 10.1. The robot performs the dog feeding routine and throws treats
11. User kills the project by pressing NXT.LEFT

As described above, the robot is launched with different kind of triggers (in version 1.0. only the ButtonSensor and the VoiceSensor are accessible) and after triggered, the robot performs throwing routine and goes back into listening mode. The user can activate triggers at runtime or via modifying the code (multiple triggers can be configured to be initialized during the startup).


Pictures 2 & 3: Mr. End-user, IT-support needs only to turn the Robot ON

Hardware and construction

Physical construction – parts used

The robot is built mainly with Lego Mindstorms NXT-set. Among a NXT-brick, other special parts are:

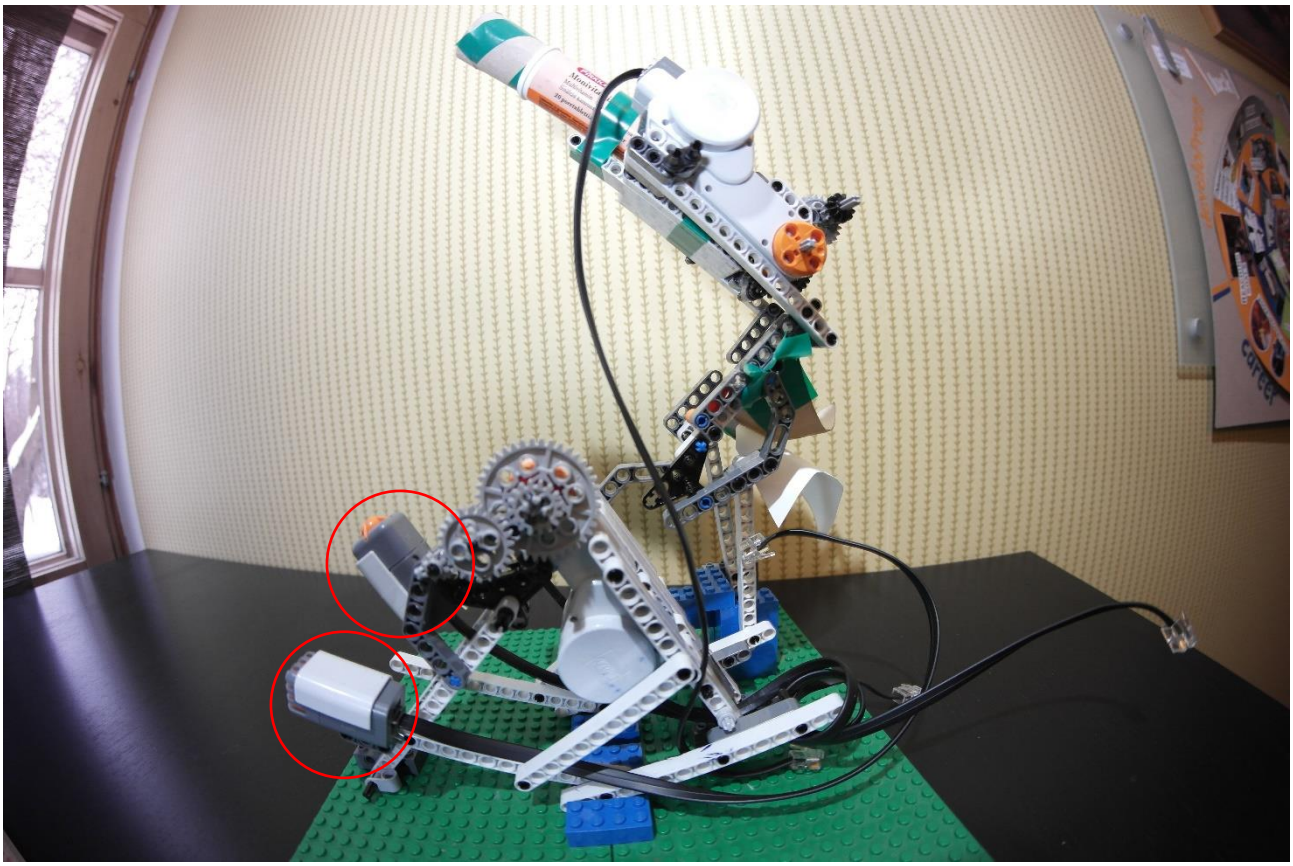
1. Two Servo-motors
2. One ButtonSensor
3. One SoundSensor

The robot stand on old Lego-platforms and blocks to make it stronger and easier to assembly. The vitamin tube is used to keep the robot loaded with dog treats. A little bit of carton and paper can be used to aim thrown ammos.

Construction demonstration with pictures

The design and vulnerabilities of the construction are explained with pictures below.

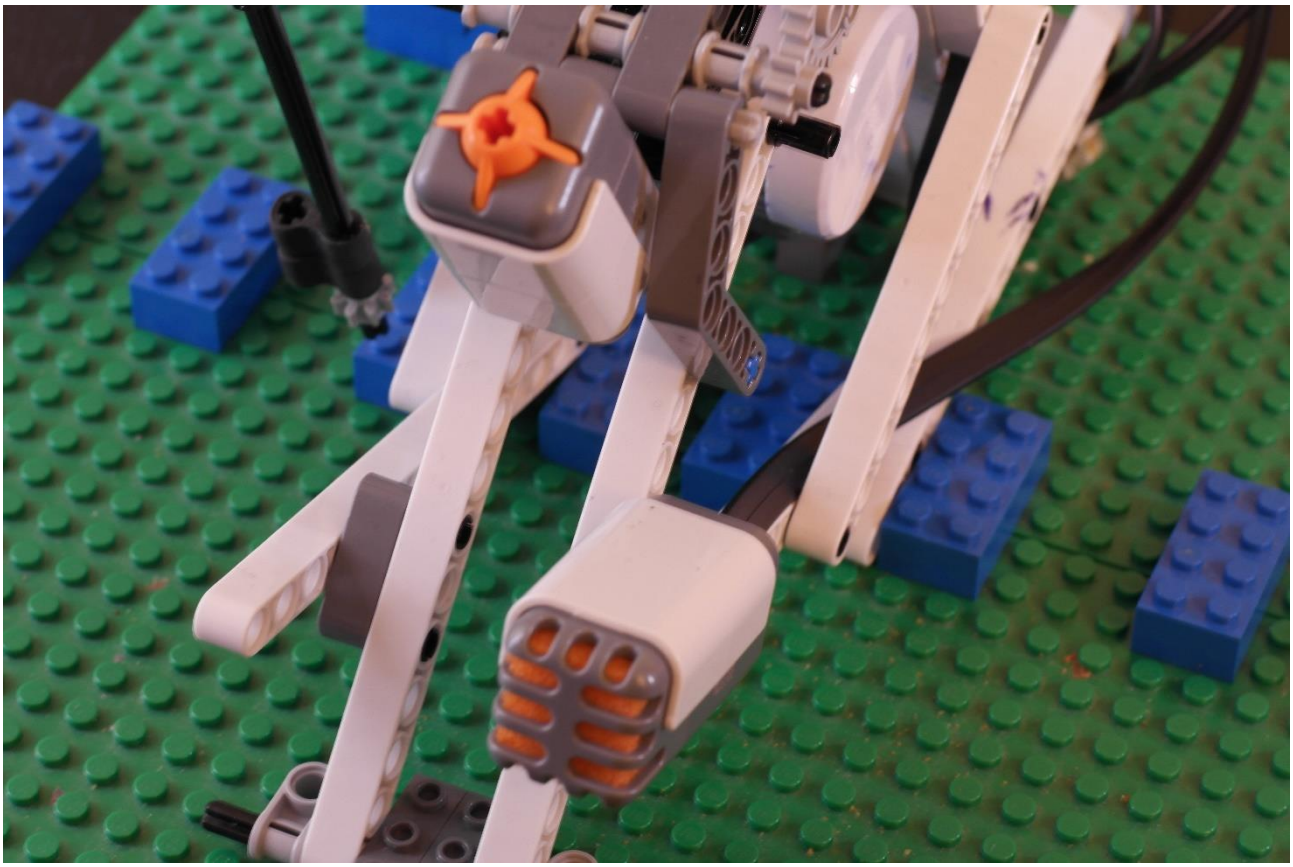
Overview of construction



Picture 4. Fish-eye overall perspective (without NXT-brick)

Picture 4 shows the whole robot which construction can be described "Scorpion-alike". Implemented triggers are marked with red circles. Picture 5 shows the TriggerSensors better; the SoundSensor is assembled low and in front to catch necessary sounds more accurately.

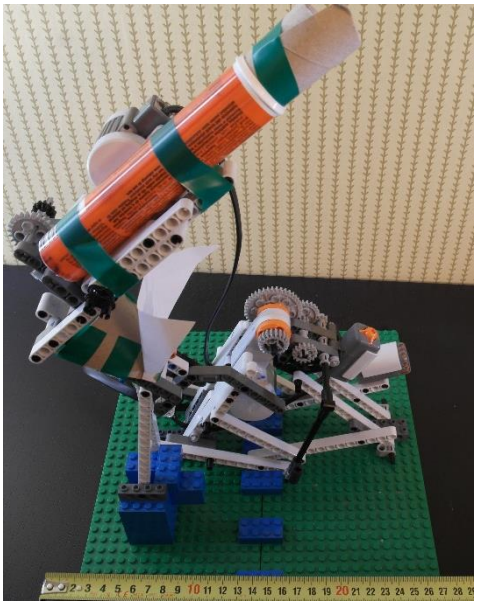
Close up of the triggers



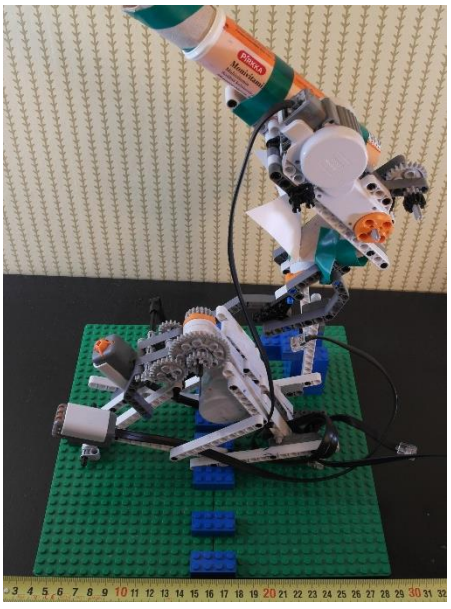
Picture 5. Button- and SoundSensors are assembled in front of the robot

The robot in different angles

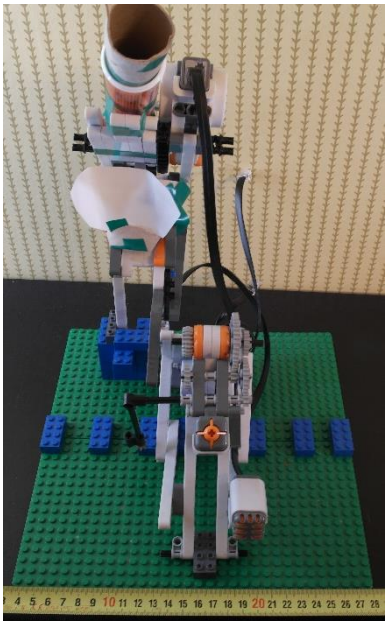
The next pictures 6-9 shows the robot in different angles. Blue bricks are used to unite the two green Lego platforms. These platforms are meant to make the light scorpion-construction stronger and easier to assembly. E.g. books can be used as weight to keep the operating unit steady and in place.



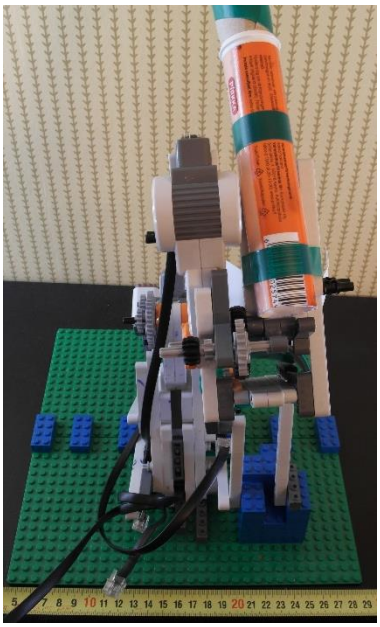
Picture 6. Robot from right



Picture 7. Robot from left



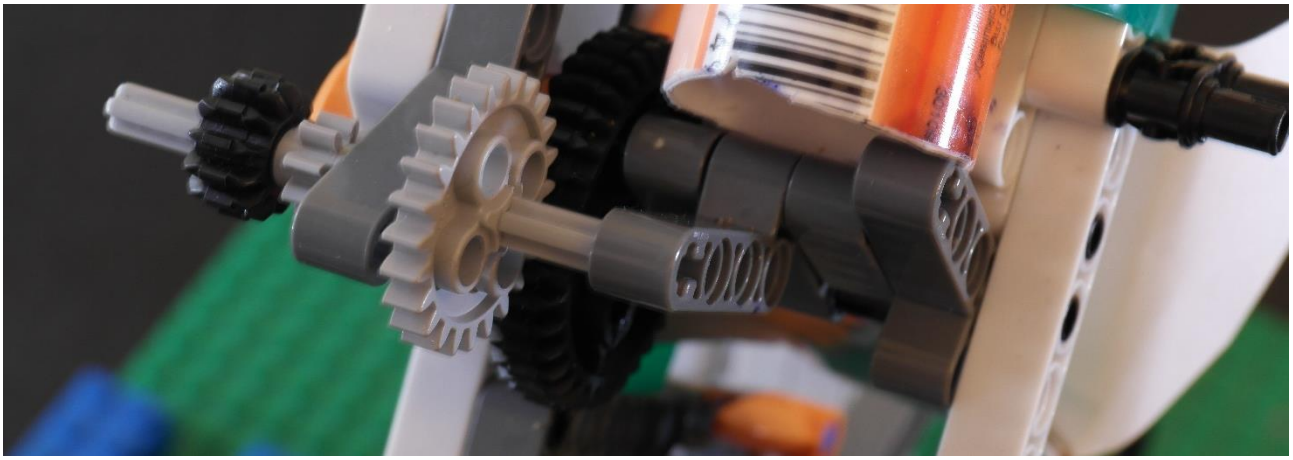
Picture 8. Robot front



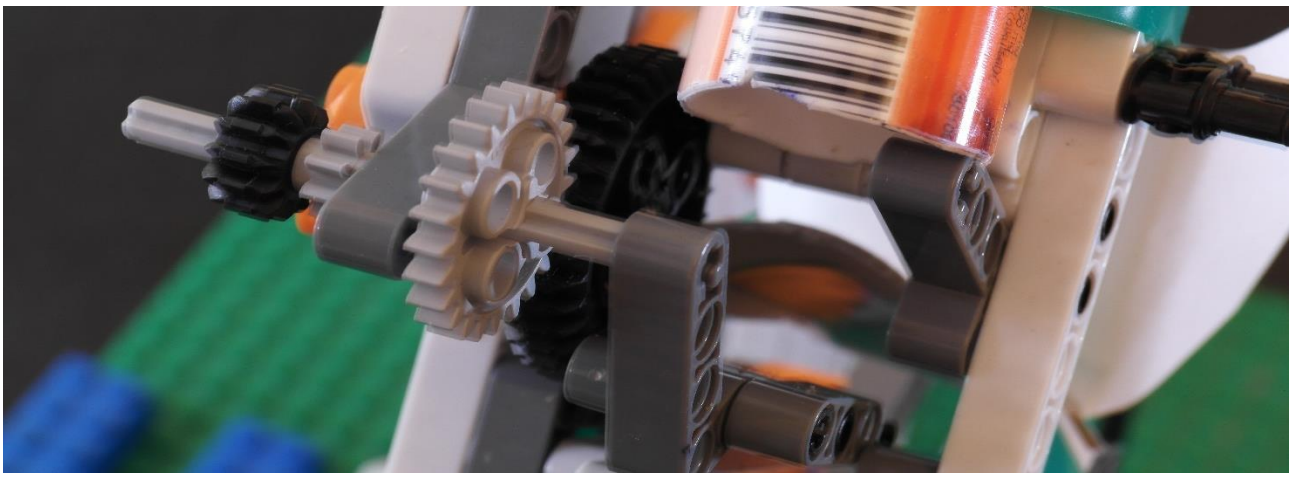
Picture 9. Robot back

Ammo container and it's releasing gate (built with servo)

The pictures above displays blue bricks in the back of the platform, which are built to strengthen the heavy ammo container/releasing structure. Picture 10 & 11 below shows the opening of the ammo container.

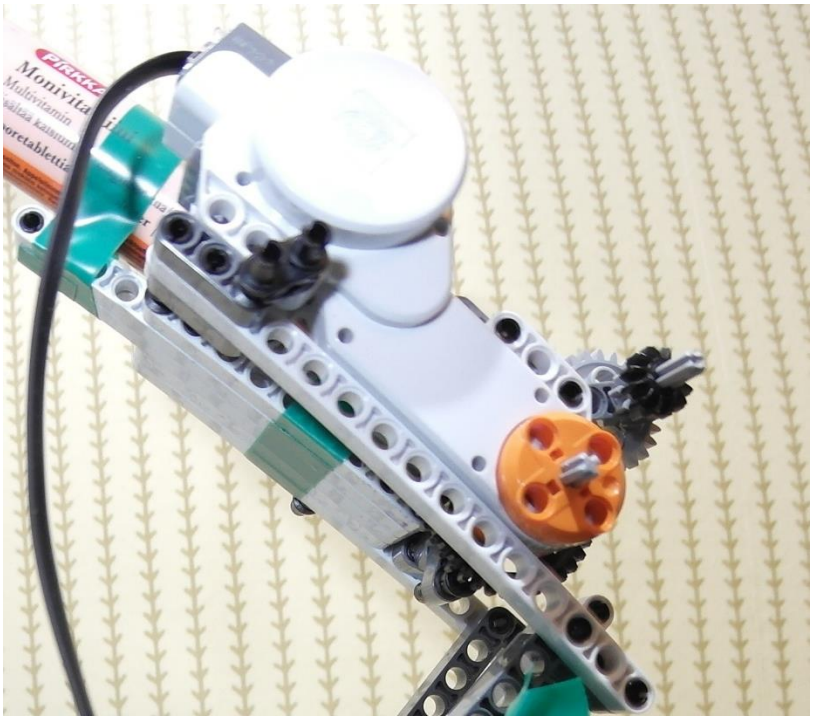


Picture 10. Ammo container closed



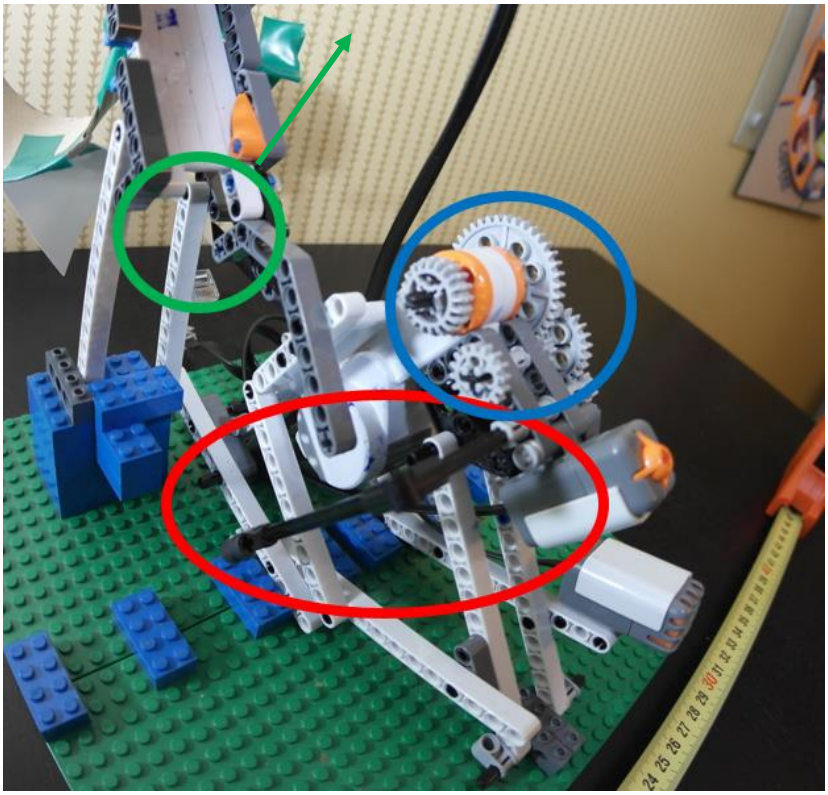
Picture 11. Ammo container opened

The opening of the ammo container is controlled by one of the two servos used in the robot. As seen in the picture above, the gear is built to make the gate open and close at a faster frequency (this is needed when releasing only a few ammos at a time). Picture 12 shows that the gate is secured with the servo near the ammo container. With limited Lego pieces it is necessary that the servo operating the opening gate, is built close to the gate.



Picture 12.

Firing unit more closely

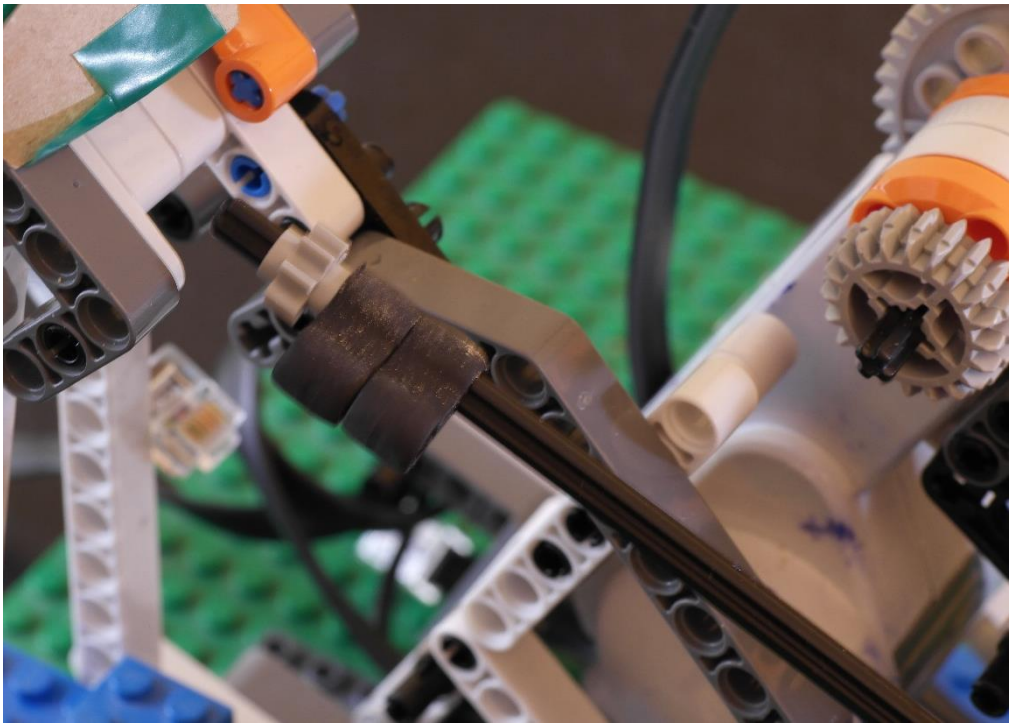


Picture 13. The firing unit

Green Circle: A rapidly swinging stick is meant to hit dog treats and throw them in the air.

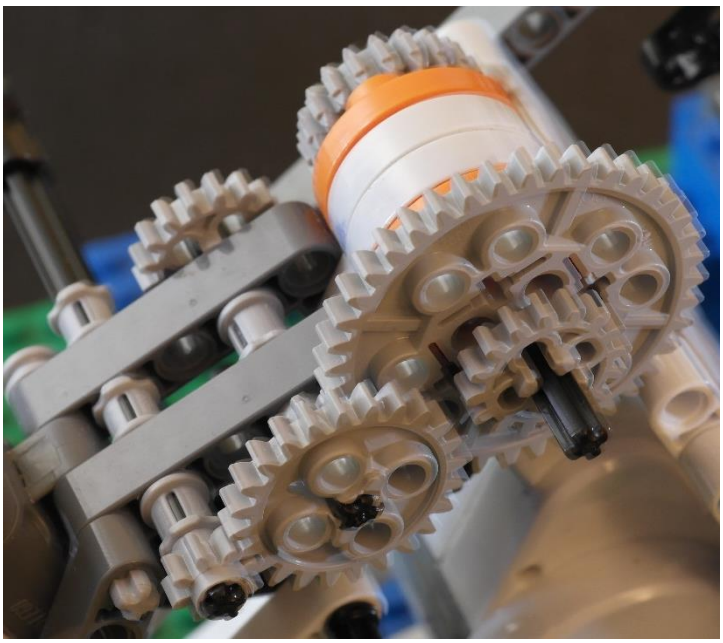
Red Circle: This is the stick which end is weighted by a Lego cogwheel and rubber. It achieves high speed thanks to the gear.

Blue Circle: The gear. This makes the stick spin much faster than the Lego servo provides as default.



Picture 14. The Stick and ammo ramp more closely.

Below is picture of the gear. The big cogwheel spins the little one, which in turn spins the bigger one, which spins the little one and so on. This results in a high RPM.



Picture 15.

Software overview – Code structure and Design

This chapter describes the code's structure, design and explains some of the choices I have made.

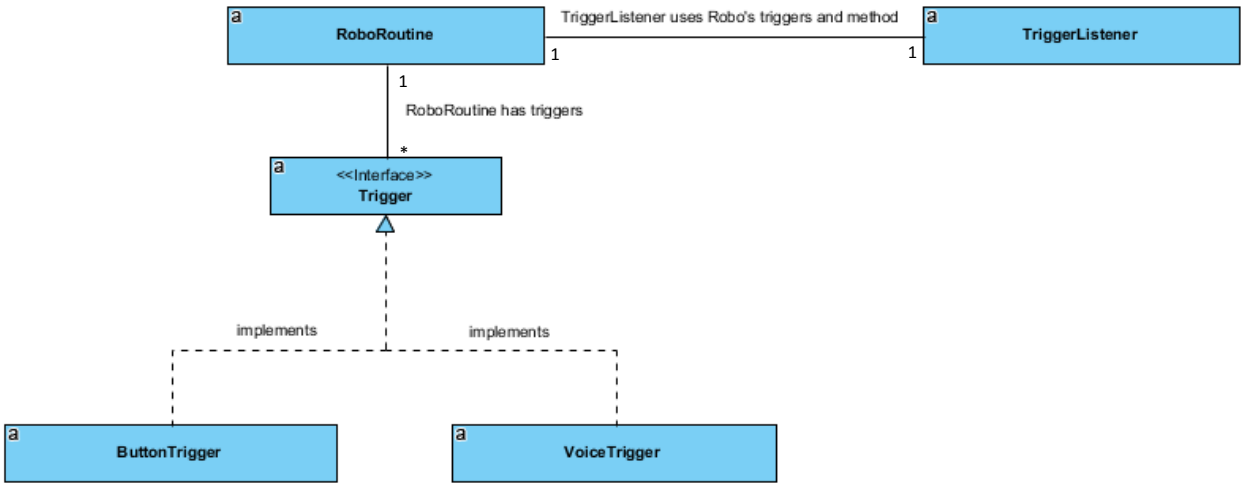
Packages and classes listed

main	basicBehaviour	triggers
Main.java	RoboRoutine.java	Trigger.java <interface>
	TriggerListener.java	ButtonTrigger.java
		VoiceTrigger.java

UML-Diagrams

Here are presented few (warning! not too strictly UML-) diagrams.

Class diagram with relations

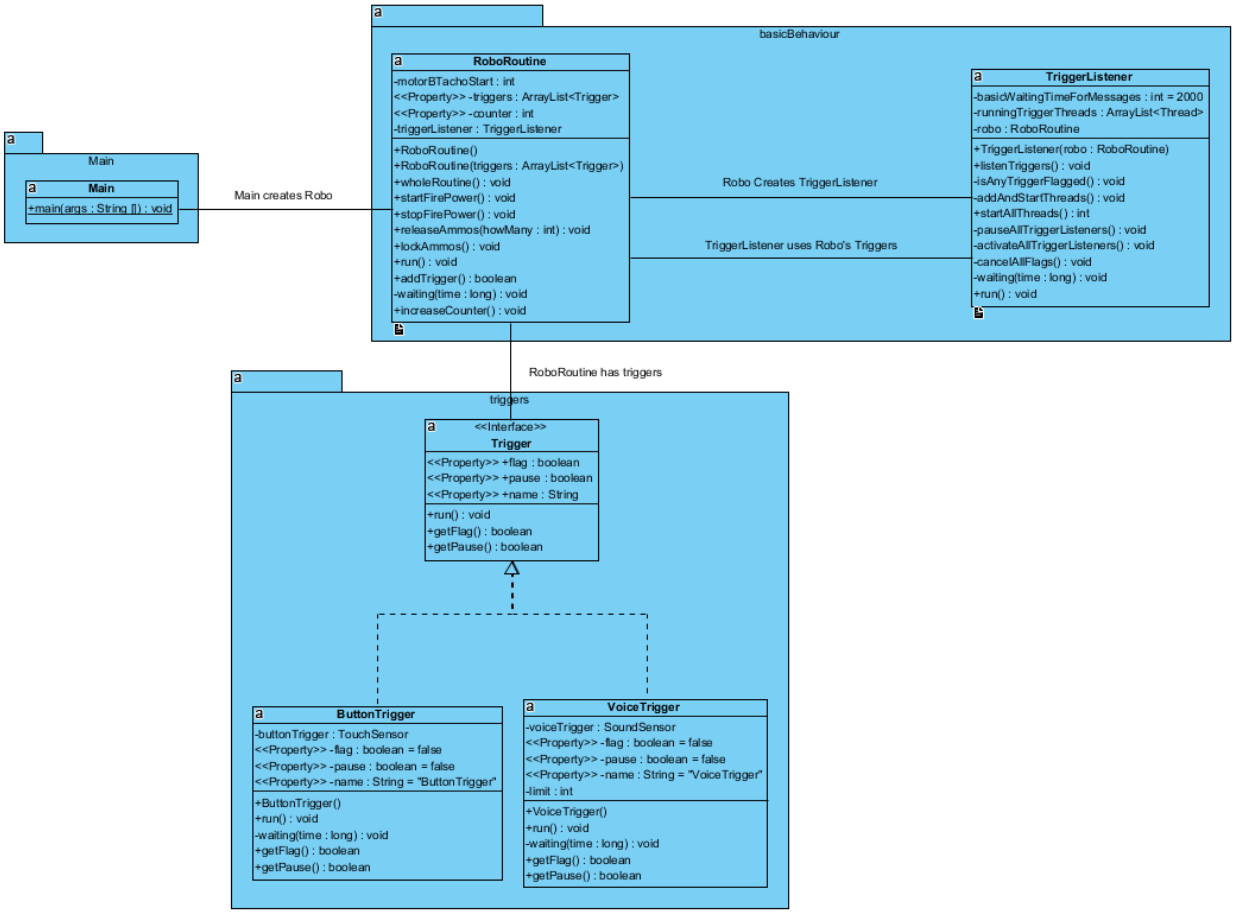


Picture 16. Class diagram

The code's high level overview:

1. RoboRoutine-class:
   1.1. Knows how to control NXT-servos to perform the dog feeding
   1.2. Knows which trigger(s) should start the dog feeding
   1.3. Creates the TriggerListener to listen necessary triggers
2. TriggerListener-class:
   2.1. Is linked to RoboRoutine-class, which created it
   2.2. Can ask triggers from RoboRoutine's trigger-list
   2.3. Can ask RoboRoutine to add a new trigger to list
   2.4. Communicates with triggers (flagged, pause, start)
   2.5. Can ask RoboRoutine to feed the dog
3. Trigger-interface:
   3.1. "Blueprint for the real instances of triggers"
   3.2. Extends Runnable, can be threaded
4. ButtonTrigger-class:
   4.1. Implements Trigger
   4.2. Initializes new TouchSensor and listens it
   4.3. Marks flag if necessary
5. ButtonTrigger-class:
   5.1. Implements Trigger
   5.2. Initializes new SoundSensor and listens it
   5.3. Marks flag if necessary

Class diagram with attributes and methods



Picture 17. Another class diagram

Future development and restriction

The Ambition from the start was to write the code, which could be easily expanded especially with new kind of triggers. Also being able to change triggers at the runtime was one of my goals.

The first idea was to create trigger, which could be launched via web browser. Unfortunately that task was too difficult to finalize within this schedule (and because of developer's limited knowledge). I started to develop the wwwTrigger at the first place. Main idea was to use old Android-phone to run nanoServer, send command to it, establish Bluetooth-connection and forward the command to NXT-brick. My wwwTrigger development crashed quite soon to problems with Android Studio and pairing problems with NXT-brick and the phone. This could be something to be developed in the future.

Ideas for future development

The future's development could aim to implement e.g.

- New behavioral triggers with the existing sensors
  - Correct barking sequence
  - Correct TouchSensor pressing sequence
  - Combination of preceding

- New triggers with new sensors
    - Ultrasonic sensor to trigger if dog performs correct tricks
        - based on movement, for example jump to trigger sensor
    - Web trigger
        - Throw all precious candies to the dog if owner's gambling account skyrockets

Every idea above could be implemented without a need to make any big changes to the existing source code.

Known bugs and test cases

Not known bugs. All user scenarios are well tested. RoboRoutine-class would not need to have TriggerListener as a class attribute though (I just noticed it while making class diagram).

I tried to comment the code clearly while writing. Also many print outs helped to debug and clean up the code. During the development, biggest difficulties were caused by my own inaccurate usage of the sensors. I tried to write and debug the code only with the NXT-brick → I learned that if multithreading the voice- and button triggers at the same time without sensors plugged in, the running threads suffocate the project → difficult case to debug.

Possible suffocations or unwanted trigger flags are taken away by pausing the trigger-threads while feeding the dog (especially needed with the VoiceSensor, because it could be triggered by a noise that comes from the operating robot).

Closer look to the code

Because the code is quite heavily commented, I kindly ask the reader to review the code further in the /src/-folder.

Picture 18. Bored dog. "The document is way too heavy, just run the code"

User manual

In the repository is loaded video for the learning purposes, it demonstrates the usage of the DogTreatThrower and is highly recommended to be watched before running the machinery. Because of the high compression and low resolution of the PDF-file, no screenshots with a text are added in this user manual.

High level instructions

1. User adds manually dog treats to the tube
2. User starts the default project from NXT-brick
3. User enters configuration mode by pressing NXT.RIGHT
    a. User makes decision which trigger to activate
        i. RIGHT again → ButtonTrigger
        ii. LEFT → VoiceTrigger
    b. If user doesn't press neither one, no trigger is added
4. User can make step 3. again at any time
5. The Robot listens given triggers and feeds the dog when conditions are met

Picture XXX. Now I have one of those *Pavlov's dogs*