# Reinforcement Learning with LEGO Mindstorms

Project Report

*Per Steffen Czolbe (wjq874)*

Handed in: December 30, 2018

## 1 Abstract

## Contents

## 2 Introduction

scientific goal, educational goals, agenda by lego

### 2.1 The ev3 robot

#### Hardware

central component: the ”brick” cpu: 32bit, 300MHz ARM9 Processor Memory: 64MB DDR RAM Storage: 16MB Flash up to 32gb SD-card

Communication: USB Bluetooth wifi

Periferals: up to 4 motors and 4 sensors.

src: LEGO MINDSTORMS EV3 Hardware Developer Kit (pdf) ©2013 The LEGO Group

#### Default Software

The ev3 is shipped with LEGOs own graphical programming software, based on the visual programming language LabVIEW.

Many community driven software packages exist. These support languages such as matlab, python, java, javascript, C++, C, GO, Ruby, Perl, R, Lue and may more (src: https://www.ev3dev.org/docs/programming-languages/). Direct control of the robot via command line is also possible.

## 3 Development Setup

- tutorial style description of how to get things to run with jupyter notebook.

### 3.1 New software: ev3dev

”ev3dev is a Debian Linux-based operating system that runs on several LEGO® MINDSTORMS compatible platforms including the LEGO® MINDSTORMS EV3 and Raspberry Pi-powered BrickPi.” src: https://www.ev3dev.org/

Because of the limited storage available on the Brick, this 2GB linux image has to be added to the ev3 via an SD-Card.

### 3.2 External Computer for computations

While the brick runs a fully fletched linux system that can execute programms on it’s own, the processing and memory contraints of the brick limit development of programms severely. Installing a new python package with pip often takes over 2 minutes. Installing deep learning frameworks like tensorflow, or performing image processing on the brick is expected to lead to severe sortages of processing power. Thus we want to run these more computationally expensive parts of the program on a seperate computer.

The setup of a main programm running on a computer and only interacting with the brick when nessesary to move motors or read sensor data has been employed by multiple community projects, such as the LEGO rubics cube solver. (src: https://www.ev3dev.org/projects/2014/05/09/Python-Rubiks-Cube-Solver/) We follow

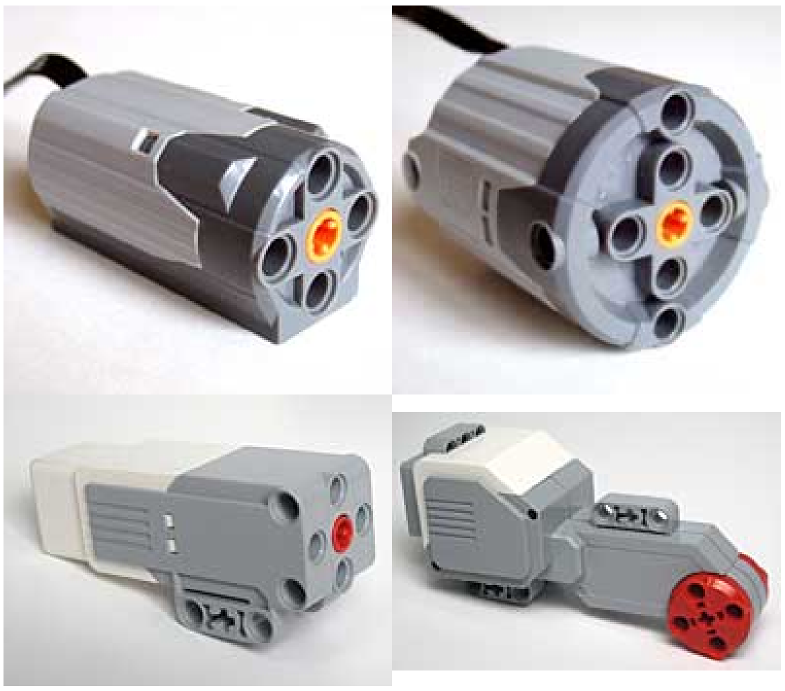

Figure 1: Overview of current (2018) LEGO motors. Upper row: Power Functions large and X-large motor. Bottom row: EV3 medium and large motor

the same design approach implemented in these projects. The brick runs a remote procedure call server, implemented by the python3 libary ‘rpyc‘. This allows other computers to connect to the brick as clients and remotely interact with the python libary controlling the robot functions. All program logic is implemented on the computer, the brick is merely listening to commands. Next to the performance gains, another benefit is that the programm can be executed in a jupyter notebook on the computer, which allows better integration of documentation and visualizations within the programm.

## 4 Sensors and Motors

experiments, description of how to interact with modules

### 4.1 Time Delays

Asger

### 4.2 Motors

The LEGO product line includes a variety of 9V DC electrical motors, which can be used to power a wide range of LEGO models, such as cars, trains and robots. The current product offering consists of two main families: “Power Functions” motors and “EV3” motors. The most common sizes of these product lines are shown in figure 1.

#### Power Functions Motors

The “Power Functions” line of motors is designed to work with a battery pack and is intended to be controlled either with a physical switch on the battery pack or a remote control. Adapters for the power cables are required to use them in combination with the EV3 platform. [2]

By changing the current and polarity of the DC power supply, these motors can spin in both directions and different speeds. Because ‘Power Functions” motors do not measure rotation or angle, robots utilizing them often require other means of measuring mechanical movements. Compared to the EV3 motors, they offer similar torque and mechanical power, have faster reaction speeds but offer no feedback. [4]

### EV3 Servo Motors

The EV3 robot kit contains a medium and two large servo motors. The EV3 motors use tacho feedback to measure their alignment and rotation, which allows for more precise control compared to “Power Functions” motors. [3]. In strength and speed they are comparable to their “Power Functions” counterparts. The large EV3 motor is roughly twice as powerful as the medium one, and comes in a bigger case with different mounting points. Detailed comparisons of the mechanical and electrical properties of all LEGO motors is available at [4].

The ev3dev programming environment offers functions for low level interactions with the EV3 motors. Available are methods to start and stop the motor, reading the angle, setting the speed and 3 different stopping patterns (coast, brake and hold). The medium and large EV3 Motor offer identical programming interfaces. For angle measurements, the motor saves the rotation at system start as 0 degrees. During operation, the current rotation angle offset to the starting point can be read by the motor. [1]

## Motor Accuracy

Training reinforcement learning algorithms requires repetition of the same motions many times. To assume a non-adversarial environment and ensure reproduce-ability of the results, actions taken by the motors need to be accurate and consistent. We design an experiment to measure the accuracy of the motors after a series of repeated actions.

### Experimental Setup

The experiment consists of an arm attached to a motor. We rotate the arm in a 90 degree angle back and forth. After every 25 iterations we measure the offset from the starting position externally with a set square. To measure the influence of weight on the results we use two independent arms, one without a weight and the other one weighted by three heavy metals balls. The experimental set-up is shown in figure 2.

The ev3dev python bindings offer two different movement patterns: `'on_for_degrees()'`, which rotates the motor relative to the current position, and `'move_to_pos()'`, which rotates the motors to an absolute value. Further, three stopping patterns exists: `'brake'`, `'hold'` and `'coast'`. In the experiment we will test different combinations of movement and holding patterns, to determine the configuration with the highest amount of accuracy and reproducibility. Since we want high accuracy, we do not test the coasting setting.

| Move Pattern | Stop Pattern | Weight | offset after 25 iterations | after 50it | after 75it | after 100it |
|---|---|---|---|---|---|---|
| relational | hold | no | 1 | 1 | 2 | 2 |
| relational | hold | yes | 9 | 11 | 35 | 45 |
| relational | brake | no | 1 | 1 | 1 | 2 |
| relational | brake | yes | 35 | 50 | >50 | >50 |
| absolute | hold | no | 0 | 1 | 2 | 1 |
| absolute | hold | yes | 2 | 1 | 4 | 3 |
| absolute | brake | no | 1 | 2 | 2 | 3 |
| absolute | brake | yes | 4 | 6 | 5 | 4 |

Table 1: Experimental result. Angle offset in degrees after 25, 50, 75 and 100 iterations for different configurations.

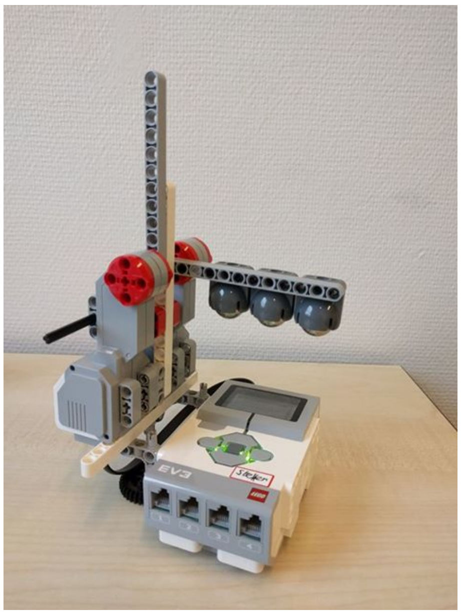

Figure 2: Experiment to measure motor accuracy. The two arms (gray) swing back and forth in a 90 degree angle. One arm is weighted down by three heavy metal balls.

**Results**

The angle offsets measured after 25, 50, 75 and 100 iterations for different motor control settings and configurations are shown in table 2. As we can see, the non-weighted arm is very accurate in all scenarios. The results for the weighted arm differ drastically between configurations. We can observe two trends:

1. The `'brake'` stopping pattern creates more offset than the `'hold'` pattern. Further observations of the experiment let us conclude that `'brake'` stops the motion of the motor, but does not block the motor for further movement after the initial momentum has been eliminated. This leads to the weighted arm pulling slightly further down between the breaking phase and the next swing of the arm. The `'hold'` stop motion holds the arm in place even after the initial momentum is eliminated. It leads to more accurate results under load.

2. Movements ot absolute positions are more precise than relative movements. While inaccuracies of relative movements accumulate over time, movements to absolute positions only carry the inaccuracy of the last movement.

During the experiment we also measured the range of motion that is possible without receiving any push-back from the motors. This amount of slack is consistently measured at about 12 degrees. No configuration lowered this inaccuracy. If it causes a problem later on, mechanical solutions such as a worm gear could be used to reduce the slack.

In conclusion, to attain the highest level of accuracy and reproducibility, motors should be moved to absolute positions and the stop pattern `'hold'` should be used. We note however that motors still allow for free movement of about 12 degrees without providing any breaking or pushback.

## 4.3 Gyroscope accuracy

## 4.4 The Ultrasound Sensor

Steffen

### 4.5 The Colour Sensor

Asger

### 4.6 Need for calibration

- most sensors no calibration - motors and gyroscope have their 0-rotation initialized at system start. This makes reclibration nessesary every time the system is restarted. The robots should be designed with this in mind. For motors, automatic calibration can be achieved by having them hit against a push sensor first

### 4.7 General building techniques

- as simple and robust as possible -

## 5 Reinforcement Learning with LEGO Mindstorms

general issues arising when working with actual robots vs simulation

## 6 Colour Detection

Asger

### 6.1 Problem

what task solved

### 6.2 The Robot

description of the robot build

### 6.3 The Learning

implementation of learning, choices made in picking and implementing algorithm

### 6.4 Results

final results

### 6.5 Discussion

includes issues / future work

## 7 Crawl-Robot

Steffen

### 7.1 Problem

what task solved

### 7.2 The Robot

description of the robot build

### 7.3 The Learning

implementation of learning, choices made in picking and implementing algorithm

### 7.4 Results

final results

### 7.5 Discussion

includes issues / future work

## 8 Swing-Robot

### 8.1 Problem

what task solved

### 8.2 The Robot

description of the robot build

### 8.3 The Learning

implementation of learning, choices made in picking and implementing algorithm

### 8.4 Results

final results

### 8.5 Discussion

includes issues / future work

## 9 Future Work

Shared/independant future robots/ multi actor learning

## 10 Individual contributions

## References

[1] EV3 python bindings. *Github Repository*, 2018.

[2] LEGO corporation. Products and Sets - LEGO® Power Functions. *LEGO.com*, 2018.

[3] LEGO education. EV3 Large Servo Motor, shop descitption. *LEGO shop*, 2018.

[4] Philippe Hurbain. LEGO® 9V Technic Motors compared characteristics. *philohome*, 2018.